NASA Technical Memorandum 78812

NASA-TM-78812 19790012597

PRELIMINARY ISIS USERS MANUAL

CAROLYN GRANTHAM

FOR REFERENCE

NOT TO BE TAKEN FROM THIS ROOM

FEBRUARY 1979

LIBRARY COPY

LANGLEY RESEARCH CENTER
HAMPTON, VIRGINIA

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665
The Interactive Software Invocation (ISIS) is an interactive data management system. ISIS is being developed to act as a buffer between the user and host computing system. ISIS provides the user with a powerful system for developing software or systems in an interactive environment. ISIS protects the user from the idiosyncrasies of the host computing system by providing such a complete range of capabilities that the user should have no need for direct access to the host computing system. These capabilities can be divided into four areas: desk top calculator, data editor, file manager, and tool invoker.
# Table of Contents

## I. Introduction ........................................... 5

## II. Interactive Software Invocation System (ISIS) Concept ........ 6
   A. Available Files ........................................ 6
   B. IPL/PASCAL Differences .............................. 7
   C. System Variables ...................................... 8

## III. Interactive Programming Language (IPL) Syntax ................ 10
   A. Abbreviations .......................................... 11
   B. Statement Summary Sheet ............................ 12
   C. Break Key ............................................... 13
   D. Operators and Functions ............................. 14
   E. Calculator Programming Statements ................... 15
      1. ABBREV .............................................. 15
      2. TYPE ................................................ 16
      3. VAR ................................................... 17
      4. ERASE .............................................. 18
      5. EXITIF ............................................... 19
      6. IF .................................................... 20
      7. FOR .................................................. 21
      8. LOOP .................................................. 22
      9. WHILE ............................................... 23
     10. REPEAT ................................................ 23
     11. XEQ .................................................. 24
     12. SET TRACE .......................................... 25
     13. CLEAR TRACE ....................................... 25
     14. ASK .................................................. 26
     15. PRINT,PRINTLN .................................... 28
F. Library Statements

1. SET NAME .................................. 30
2. USE ........................................ 31
3. USING ....................................... 32
4. SAVE ......................................... 33
5. PURGE ....................................... 34
6. DISPOSE ..................................... 35

G. Text Editing Statements

1. Used with ACTIVE PAGE only
   a. INSERT .................................... 36
   b. DELETE .................................... 37
   c. REPLACE ................................... 38
   d. CHANGE .................................... 39
   e. ADD ........................................ 40

2. Used with ACTIVE page and read only pages
   A1, A2, and A3 (Read only statement commands)
   a. LIST ....................................... 41
   b. COUNT ...................................... 42
   c. EXEC ........................................ 43

H. Tool Invocation Statements

1. RUN .......................................... 45
2. SEND ........................................ 47

I. Interrogation Statements

1. SHOW SHOWS ................................ 48
2. SHOW RESERVED .............................. 49
3. SHOW STATEMENTS ............................. 49
INTRODUCTION

This documentation covers the latest version of the Interactive Software Invocation System (ISIS)(I-15-79) discussing the syntax and operation of the Interactive Programming Language, IPL. IPL is based on the higher order language PASCAL and anyone wishing to use ISIS should have a working knowledge of PASCAL or have access to a PASCAL Manual (see ref. 1). PASCAL was chosen as the base language because of the wide range of the capabilities of the language. IPL and PASCAL differences are discussed on page 6. The IPL language contains statements for performing desk top calculations, data and text editing, file management, and tool invocation. IPL contains most of the arithmetic operations, functions, and control statements of a traditional programming language which allows powerful desk top calculations and the programming of all operations. The editor manipulates pages of text or data. The IPL statements allow insertion, deletion, replacement, and modification of lines of text or data. The file manager allows the user to save, access, and purge pages within a 5-level hierarchical file system. The tool invoker allows the user to communicate with the host computer system.

ISIS is being developed as a part of the MUST program at LaRC and is written for the MUST software developers. Anyone else having an application for ISIS is welcome to use it with the understanding that ISIS is not a production system, but a developing system.

ISIS is stored on a direct access permanent file under the user number 961300N. The control statements to access and use the file are:

ATTACH,ISIS/UN=961300N. - Get ISIS

ISIS. - Execute ISIS

ISIS may be terminated by typing STOP.
A "page" is a numbered sequence of lines of text. The text editor allows the user to edit a page of data at a time. When not being edited, pages are stored within a "library" maintained by the file manager.

A library is a direct access file containing a 5-level hierarchical directory plus the contents of the pages saved within this hierarchy. The user has access to 4 "working pages" at a time. A working page is a copy of a library page and may be designated ACTIVE, A1, A2, or A3. However, only one of these, the ACTIVE page, can be edited. A1, A2, and A3 are read-only pages and cannot be modified. All code modification takes place in the ACTIVE page. The user selects a page from the library, puts a copy of that page in the ACTIVE page, and then can modify the code using any of the edit statement verbs. A1, A2, and A3 are read-only pages which means that A1, A2, and A3 can be used for temporary storage of code for examination or for use with ready-only statement verbs such as LIST, RUN, EXEC and COUNT. A1, A2, and A3 files are especially useful when the user is creating an INPUT file (using RUN statement verb) for submission to the host computer. For example, the control cards might be stored on A1, the program source on A2 and the data on the ACTIVE page. This would permit the user to easily change the data for the different runs. It should be noted that text cannot be moved back and forth between the working pages. The text can only be brought from the library to a working page.
IPL/PASCAL Differences

IPL contains the PASCAL variable types, REAL, INTEGER, and BOOLEAN in abbreviated forms; REAL, INT, BOOL. ISIS has two data types not in PASCAL; STRING and KEY. ISIS deviates from PASCAL in not allowing CHARS and ALFA's but instead includes a type called STRING. This STRING type is similar to the PASCAL ALFA except that there is no set length on the STRING. A STRING contains alphanumeric information enclosed by quotation marks and may be assigned to a variable. It should be noted that if an odd number of characters are contained in the string, a blank will be added to the end of the string. This is due to the characteristics of the CDC 6600 computer.

In IPL the semicolon (;) at the end of a statement is optional if it is the only statement on that line. If there is more than one statement on a line, then the semicolon (;) must be used to separate the two statements.

Another IPL/PASCAL difference is in the assignment statement. IPL requires only an = for assignments whereas PASCAL requires a :=.

Another difference appears in the IF and loop statements. IPL does not require the BEGIN and END's around the THEN and ELSE parts of the statement. All that is required is an END to terminate the IF or loop statement.
System Variables

There are several ISIS system variables which have been made available to the user. These variables are contained in a record (ref. 1, p. 42) named SYSTEM. The SYSTEM record field identifiers are as follows:

**.VERBOSE**
- Not used at present (default value = T.)
  - .VERBOSE is a BOOLEAN variable.

**.DELTA**
- System variable name of the range increment (inc) on the Text Editing Statement, INSERT (default value is 1)
  - .DELTA is a KEY variable

**.ALARM**
- Twenty-four hour clock alarm - when (ALARM - clock time) is 0, then a message is printed to the user of the CRT - ***ALARM***
  - This may be useful to the person with a very tight schedule and a poor memory.
  - .ALARM is an INTEGER variable

**.CLOCK**
- Is the number of elapsed milliseconds in the current terminal session (read only variable)
  - .CLOCK is an INTEGER variable

**.TIME**
- Current day time (read only variable)
  - .TIME is a STRING variable

**.DATE**
- Current date (read only variable)
  - .DATE is a STRING variable

To obtain the information in this record, the user may inquire with SHOW ID SYSTEM, or PRINT SYSTEM. The SHOW ID SYSTEM will print out the field identifier names and types. The PRINT SYSTEM will print out the current values of these field identifiers.

**EXAMPLE:**

```
08.53.10? ALARM = 0856  - set the alarm to go off at 8:56 a.m.
08.53.29?PRINT .CLOCK
26900  - Print the clock time (SYSTEM.CLOCK)
08.55.20?PRINT .TIME
08.55.43.  - Print the day time
08.55.44?PRINT .DATE
78/11/09.  - Print the date
08.55.52?VAR A:ARRAY [1..5] OF RECORD F1:STRING; F2:INT END - Continue inserting program statements
***** ALARM *****  - Message printed when the alarm goes off.
```

8
System Variables cont'd

08.52.51?SHOW ID SYSTEM VARIABLE 'SYSTEM ': RECORD
  VERBOSE: BOOL;
  DELTA: KEY;
  ALARM: INT;
  CLOCK: READONLY (INT);
  DATE: READONLY (STRING);
  TIME: READONLY (STRING);
END;

08.52.57?PRINT SYSTEM
VERBOSE = TRUE
DELTA = 1.
ALARM = 0
CLOCK = READONLY
DATE = READONLY
TIME = READONLY

- Displays type information for SYSTEM record.
- Print the values of the SYSTEM record variables.
INTERACTIVE PROGRAMMING LANGUAGE (IPL) SYNTAX

The IPL programming statement write-ups will consist of the statement format at the top of the page. The words appearing in caps must be typed as shown whereas the words in small letters will be replaced with appropriate information. A discussion of the statement, what it does, and how to use it follows the format. Following the discussion is an example illustrating how the statement can be used.

The statements are discussed in groups according to capabilities. The first group, Calculator Programming Statements, contains statements of a traditional programming language which will allow the programming of all operations and powerful desk top calculations. The second group contains File Management Statements. The user is able to save, replace, access, and purge information which is contained within a 5-level hierarchical file system. The third group, Text Editing Statements, contains statements which allows editing (Insertion, Deletion, Replacing, and Modification) of lines of text or data which is contained in the file system. The fourth group, Tool Invocation Statements, allows the user to communicate with the host computer system. The fifth, and last group, Interrogation Statements, allows the user to make inquiries of ISIS, relating to statements in any of the above groups.
Abbreviations

These abbreviations will be used in the IPL statements or in the discussions of the statements.

?    ISIS prompt - informs the user that ISIS is ready.
>
ISIS indication that the current statement is not yet complete and more input is required.

id   Identifier

ln   Line number

n\#   A specific number of lines

an   Any number

lim  Limited number of lines

[ ]   Optional information for command

{ / } Command choices are enclosed in brackets separated by slashes (/).

col  Column number

inc  Line increment

...  ISIS acknowledgement to the [BREAK] key

.    Separation in DATA BASE statements
### Showing Equivalence Between Statement Verbs and Interrogation Statements

<table>
<thead>
<tr>
<th>PROGRAMMING STATEMENTS</th>
<th>INTERROGATION STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREV abbreviation(s): statement-verb</td>
<td>SHOW SHOWS</td>
</tr>
<tr>
<td>TYPE type-id: Type specification</td>
<td>SHOW RESERVED</td>
</tr>
<tr>
<td>VAR variable-id: type-id</td>
<td>SHOW STATEMENTS</td>
</tr>
<tr>
<td>ERASE type-id or variable-id (s)</td>
<td>SHOW AVAIL</td>
</tr>
<tr>
<td>EXITIF conditional</td>
<td></td>
</tr>
<tr>
<td>IF condition THEN statement(s) [ELSE statement] END</td>
<td></td>
</tr>
<tr>
<td>FOR var-id = initial value {TO</td>
<td>DOWNTO} final-value DO statement(s) END</td>
</tr>
<tr>
<td>LOOP statement(s) EXITIF condition statements(s) END</td>
<td></td>
</tr>
<tr>
<td>WHILE condition DO statement(s) END</td>
<td></td>
</tr>
<tr>
<td>REPEAT statement(s) UNTIL condition</td>
<td></td>
</tr>
<tr>
<td>XEQ string-expression</td>
<td></td>
</tr>
<tr>
<td>SET TRACE variable-id</td>
<td></td>
</tr>
<tr>
<td>CLEAR TRACE variable-id</td>
<td></td>
</tr>
<tr>
<td>PRINT, PRINTLN exp [:Format1[:Format 2]]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LIBRARY STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET NAME [LIB].[SHELF].[BOOK].[CHAPTER].[PAGE]</td>
</tr>
<tr>
<td>USE [LIB].[SHELF].[BOOK].[CHAPTER].[PAGE][:A1</td>
</tr>
<tr>
<td>USING {A1</td>
</tr>
<tr>
<td>SAVE [*]</td>
</tr>
<tr>
<td>PURGE [LIB].[SHELF].[BOOK].[CHAPTER].[PAGE]</td>
</tr>
<tr>
<td>DISPOSE [OP] {ACTIVE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEXT EDITING STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT range</td>
</tr>
<tr>
<td>DELETE range [:NL</td>
</tr>
<tr>
<td>REPLACE range [:NL</td>
</tr>
<tr>
<td>CHANGE string-id TO string-id IN range [:NL</td>
</tr>
<tr>
<td>ADD string-id [AT column] IN range [:NL</td>
</tr>
<tr>
<td>LIST [range][:NI</td>
</tr>
<tr>
<td>COUNT [range]</td>
</tr>
<tr>
<td>EXEC [range] [:ECHO]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOOL INVOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN [range][:ECHO]</td>
</tr>
<tr>
<td>SEND</td>
</tr>
</tbody>
</table>

**NOTE:** ( ) - no corresponding statement verb
Break Key

The BREAK key enables the user to discard the line that he is in the middle of typing and reenter it. This line may be a command statement (statement verb) or a line in the ACTIVE page the user is in the process of editing. A good example would be if he made a typographical error in the text being inserted. If the user hits the BREAK key, ISIS will reprompt him for reentry of that line at which time he may type the correct information. ISIS acknowledges the BREAK key by printing out 3 dots (...).

EXAMPLE:

13.31.56?INSERT 4//2
  4. =THIS IS
  6. = AN
  8. =EXAMPLE
 10. =PROGRAM...
12. =FOR THE
14. =BROCK...
16. = BREAK KEY

INSERT TERMINATED.
13.32.54?LIST
  4. =THIS IS
  6. = AN
  8. =EXAMPLE
 10. =PROGRAM
12. =FOR THE
14. =BREAK KEY

- Mistake here in spelling. The BREAK key was depressed and ISIS responded with 3 dots and prompted the user for reentry of line 10.

- Another mistake in spelling -- hit the BREAK key and ISIS responds with dots (...) and prompts the user for line 14.

- List the ACTIVE page to check correctness.
Operators and Functions

Shown below are the operators and functions available to the user in programming desk top type calculations.

RESULT TYPE:

STRING:

Functions
  CAT(X,Y): Concatenation of X and Y strings
  SUB(X,Y,Z): Substring of X (string) starting at character number Y (Integer) for Z (Integer) number of characters. Y and Z can be variables, constants, or expressions.

INT:

Operators +,-,DIV,MOD

Functions
  ABS(x): Absolute value of integer x
  LEN(x): Length of string x
  ORD(x): Character set dependent ordinal of the first character of the string x (ORD('C') = 3)
  ROUND(x): Rounded value of real x
  SQR(x): Square of integer x
  TRUNC(x): Truncated value of real x
  LOC(S1,S2): Is S2 a substring of S1?
    If S2 is an undefined string variable (zero length string) then LOC(S1,S2) = -1
    If S2 is not a substring of S1 then LOC(S1,S2) = 0
    If S2 is a substring of S1 then LOC(S1,S2) = the character position (index) within S1 at which the 1st occurrence of S2 begins

REAL:

Operators +,-,/,*

Functions
  ABS(x): Absolute value of real x
  ARCTAN(x): Arc-tangent of x radians
  COS(x): Cosine of x radians
  EXP(x): e raised to the power x
  LN(x): Natural logarithm of x
  SIN(x): Sine of x radians
  SQR(x): Square of real x
  SQRT(x): Square root of real or integer x

BOOL:

Operators <,<=,>,>=,=,AND,OR

Functions
  ODD(x): x must be integer. The result is BOOLEAN.

   If x is odd then ODD(x) = TRUE
   IF x is even then ODD(x) = FALSE
ABBREV abbreviation(s) : statement - verb

The ABBREV statement allows the user to abbreviate ISIS statement verbs. More than one abbreviation may be given to a single statement verb. See SHOW STATEMENTS for list of verbs that may be abbreviated.

EXAMPLE:

09.47.01?ABBREV P,PR;PRINT
09.47.22?SHOW ABBREVS
'PR ; PRINT
'P ; PRINT
09.47.50?P X
 15
09.48.30?PR X
 15
09.48.37?PRINT X
 15

- Set P & PR as abbreviation for PRINT
- Use abbreviations P & PR in place of PRINT
Calculator Programming Statements

TYPE type-id I=J:l type-specification

The TYPE statement is similar to the TYPE section of a PASCAL program. It allows the user to specify a new TYPE for subsequent use in variable declaration statements. The type specification consists of combining any of the system-provided types (INT, BOOL, REAL, STRING, ARRAY, RECORD) into RECORDS and ARRAYS, etc. to obtain a user-defined type. Subsequent type specifications may involve previous user-defined types in addition to system-provided types.

EXAMPLE:

09.43.49?SHOW TYPES
'BOOL ' : BOOL;
'INT ' : INT;
'KEY ' : KEY;
'REAL ' : REAL;
'STRING' : STRING;
09.45.12?TYPE PERSONS:REAL
09.46.10?TYPE MESS : ARRAY[1:5] OF BOOL
09.51.04?TYPE REC1 : RECORD NUM:INT; FLAG:BOOL; NAM:STRING; END;
09.52.57?TYPE RECM:ARRAY[1:3] OF REC1;
09.53.42?

SHOW TYPES
'BOOL ' : BOOL;
'INT ' : INT;
'KEY ' : KEY;
'MESS ' : ARRAY [1:5] OF BOOL;
'MISS ' : ARRAY [1:2] OF
    ARRAY [1:5] OF BOOL;
'PERSONS ' : REAL;
'REAL ' : REAL;
'RECM ' : ARRAY [1:3] OF
    RECORD
        NUM: INT;
        FLAG: BOOL;
        NAM: STRING;
    END;
'REC1 ' : RECORD
        NUM: INT;
        FLAG: BOOL;
        NAM: STRING;
    END;
'STRING ' : STRING;
09.54.18?

- Initial type-id table provided by ISIS system (Do not erase any of these!)

Declare new types

- New declared TYPES have been added to the type-id table
VAR variable-id : type-id

The VAR statement is similar to the VAR section of a PASCAL program. It allows the user to assign a prespecified type (REAL, INTEGER, BOOLEAN, STRING, and user defined types) to program variables. All program variables must be declared in this manner. If the user fails to declare all variables being used, the ISIS system interrogates the user for the type of the undeclared variable instead of aborting the command.

Declared variables are assigned default values by the ISIS system. Integers and real numbers are set equal to zero, Booleans are set FALSE and strings are empty (zero length).

EXAMPLE:

14.00.08?SHOW VARS
** NONE **
14.00.20?VAR X,Y,Z:REAL;
14.00.43?VAR I,J:INT
14.00.59?VAR B,C:BOOL
14.09.10?VAR HAM: STRING
14.09.53?VAR INPT: RECM;
14.10.09?TYPE LOCK:RECORD UNL:BOOL; SHUT:STRING; END;
14.10.42?VAR BTHE: LOCK
14.11.01?

  SHOW VARS
  'BTHE  " RECORD
    UNL: BOOL;
    SHUT: STRING;
  END;
  'B  " BOOL;
  'C  " BOOL;
  'HAM  " STRING;
  'INPT  " ARRAY [1..3] OF BOOL;
  'I  " INT;
  'J  " INT;
  'X  " REAL;
  'Y  " REAL;
  'Z  " REAL;

14.09.48.45?Z=A+Y;

- Declare variables
- Define new types
- Display variables in symbol table to show the new declared Variables have been included in the symbol table
- Equation to be calculated contains the undefined variable A. System interrogates user for type. User responds with type and execution continues.
The ERASE statement removes the specified types or variable-ids from the identifier tables. More than one id may be erased at one time with ids separated by a comma. Caution should be exercised when using ERASE. Erasure of a TYPE will not affect already defined variables of that type, but it will prevent the user from defining new variables of that type.

**EXAMPLE:**

```
13.01.42?SHOW TYPES
'bBOOL', 'bINT', 'bKEY', 'bPERSONS', 'bREAL', 'bSTRING
13.01.51?SHOW VARS
'a', 'b', 'c', 'prom', 'resp', 'x', 'y', 'z
13.05.31?
13.06.28?

- Display existing types

- Display existing variables

- Erase types and variables from tables

- Display types and variables again to show that the erased variables and types were removed from the identifier table
```
The EXITIF statement allows the user to exit from the middle of a loop statement (IF, WHILE, REPEAT, LOOP and FOR) when a specified condition becomes true. The EXITIF may appear anywhere in the loop.

```
11.37.47\%LIST
  1.  \tex\textcolor{red}{=}VAR R,T,X:REAL\}
  3.  \textcolor{red}{=}R=10.
  5.  \textcolor{red}{=}T=-1.745329f
  7.  \textcolor{red}{=}WHILE T < 3.14 DO
  9.  \textcolor{red}{=} X=R*COS(T)\}
11.  \textcolor{red}{=} PRINTLN T=57 295780, T=DEG, \textcolor{red}{=}X\}
13.  \textcolor{red}{=} T=T+.1745329f
15.  \textcolor{red}{=} EXITIF X <= 0\}
17.  \textcolor{red}{=}END
```

- **A WHILE loop**
  - containing an EXITIF statement
  - (residing in ACTIVE page)

- **Execute ACTIVE page**

The output shows the WHILE loop was exited via the EXITIF (x becomes < 0)
Calculator Programming Statements

IF condition THEN statement(s) [ELSE statement(s)] END

The IF statement allows for conditional execution of other statements. The condition must evaluate to a BOOLEAN value. If the condition is TRUE, the statements following the THEN are executed and those following the ELSE (if present) are skipped. If the condition is FALSE, the statements following the THEN are skipped and those following the ELSE (if present) are executed.

It should be noted that ISIS deviates from PASCAL by not requiring BEGIN . . . END's around the THEN and ELSE sections of the IF statement when multiple statements are contained in them.

EXAMPLE:

15.08.59?LOOP
15.09.10> IF B<A THEN PRINTLN B; B=B+1;
15.09.39> ELSE IF B=A THEN PRINTLN 'ONE MORE STEP';
15.10.06> ELSE PRINTLN 'READY TO STOP';
15.10.29> END
15.10.36> B=B+1
15.10.46> END
15.10.53> EXITIF B>16
15.11.05>END
1.00000000000000E+01
1.10000000000000E+01
1.20000000000000E+01
1.30000000000000E+01
1.40000000000000E+01
ONE MORE STEP
READY TO STOP
Calculator Programming Statements

FOR var-id = initial-value TO final-value DO statement(s) END

The FOR statement is another form of loop statement which allows the user to perform a sequence of statements repeatedly while the variable-id takes on a progression of values between an initial and final value. This progression may go either upward or downward in value. The initial-value and final-value may be variables, literals, or expressions.

EXAMPLE:

```
18.01.35?VAR A:REAL;
18.01.44?VAR Z:REAL;
18.01.51?VAR L:INT;
18.01.57?VAR X:INT;
18.02.04?X=-5
18.02.15?
   FOR L=X+15 DOWNTO 1 DO A=L*X;
18.02.42?
   Z=L+1
18.02.52?
   PRINT L,A
18.03.02?END
```

For statement with downward progression (10 to 1)
Calculator Programming Statements

LOOP [statement(s)] EXITIF condition [statement(s)] END

The LOOP statement is a generalization of the WHILE and REPEAT statements. A set of statements are executed repetitively until the condition of the EXITIF becomes true. This EXITIF becomes part of the loop statement. EXITIF may appear anywhere in the loop and when the condition becomes true, the loop is exited at that point in the code. If the EXITIF is left out, the LOOP will be executed infinitely. If this occurs, the user can abort the command by depressing the [BREAK] key.

EXAMPLE:

LOOP [statement(s)] EXITIF condition [statement(s)] END

13.17.46?x=45.45; I=99; D=TRUE;
VAR $:STRING;
13.22.36? LOOP
13.22.42> ASK $,'TYPE EXP'
13.23.05> XEQ CAT('PRINTLN ','S')
13.23.28> EXITIF $='0'
13.23.43> END
TYPE EXP6+6*2
18
TYPE EXP 4*4
16
TYPE EXP5
5
TYPE EXPX*2
9.0900000000000E+001
TYPE EXP1
99
TYPE EXP0
TRUE
TYPE EXP0
0
13.26.19?

Loop of statements is executed until the user types an input of S = 0.
Calculator Programming Statements

WHILE condition DO \{statement(s)\} END

REPEAT \{statement(s)\} UNTIL condition

The WHILE and REPEAT statements are a type of LOOP statement. The statements contained in the loop will be executed WHILE a certain condition exists or be repeated UNTIL a certain condition occurs.

The WHILE statement evaluates a condition, which must reduce to a BOOLEAN result. If the condition is FALSE, the statements are skipped. If the condition is TRUE, the statements are executed and the condition is then re-evaluated. If the condition is again TRUE, the statements are re-executed and the condition is then re-evaluated. This process continues until the condition is FALSE. When this happens, the statements are skipped and execution continues with the next statement.

The REPEAT statement is similar to the WHILE statement. The differences are that the REPEAT statement first executes the statements it controls and then evaluates and checks the condition, and that the statements are re-executed as long as the condition is FALSE, i.e., the statements will always be executed at least once.

EXAMPLE:

```
12.21.18? VAR S:STRING
12.22.16? VAR I:INT
12.22.25? S='CHECK WHILE'
12.22.43? I=1
12.22.49? WHILE I<10 DO PRINTLN SUB(S,I*1),2*I
12.23.29> I=I+1
12.23.38> END
```

- While I < 10 print a subset of the string S with a format 2*I
Calculator Programming Statements

XEQ string-expression

The XEQ statement allows the user to specify that the contents of a string-expression are to be interpreted as commands to the system.

EXAMPLE:

12.16.39?
12.18.01?VAR S:STRING
12.18.21?VAR ABC:REAL
12.18.30? S='1+2+3'
12.18.46? ABC=10
12.18.56?
  XEQ CAT("PRINTLN ",S)
     6
  XEQ 'PRINTLN S'
1+2+3
12.19.47?
  S='ABC'
12.20.26?
  XEQ CAT("PRINTLN ",S)
1.000000000000E+001

S='FOR I=1 TO 5 DO PRINTLN I END';
13.31.43?XEQ S
  1
  2
  3
  4
  5

- PRINTLN is concatenated (CAT) with contents of S (1+2+3) and then executed (XEQ). PRINTLN 1+2+3 evaluates expression and prints the value (6).

- PRINTLN S is executed, printing value of S, which is 1+2+3

- Redefine S

- PRINTLN is concatenated with contents of S (ABC) then executed (XEQ). PRINTLN ABC evaluates ABC and prints the value of the variable (10).

- User can equate a number of statements to a string variable and then execute these statements by executing that string variable (XEQ S)
SET TRACE variable-id

CLEAR TRACE variable-id

The SET TRACE statement is used to set up a trace of a variable. Each time the variable's value is assigned, the variable-id and the new value are printed. A TRACE is limited to one argument. A TRACE applies to entire program. Once a variable is traced, it will be traced wherever it is used. Records and array variables cannot be traced (10-18-78).

The CLEAR TRACE command releases a TRACE on a variable.

EXAMPLE:

```
11.21.52?VAR I,J:INT
11.22.18?VAR X:REAL
11.22.21?VAR B:BOOL
11.22.28?VAR S:STRING
11.22.36? J=1; X=0
11.22.56? SET TRACE J
  J  NOW TRACED.
11.23.15?SET TRACE X
  X  NOW TRACED.
11.23.26? LOOP
  11.26.55> J=J+1;
  11.27.05> IF J < 5 THEN X=X+2;
  11.27.23> ELSE X=X-1;
  11.27.38> END
  11.27.44> PRINTLN '***':2*J;
  11.28.07> EXITIF J=6;
  11.28.36>END

  J  = 2
  ***
  J  = 3
  ***
  J  = 4
  ***
  J  = 5
  ***
  J  = 6
  ***
```

- Set trace on J
- Set trace on X

```
  X  = 2.00000000000000E+00
  ***
  X  = 4.00000000000000E+00
  ***
  X  = 6.00000000000000E+00
  ***
  X  = 5.00000000000000E+00
  ***
  X  = 4.00000000000000E+00
  ***
```

- TRACE output is printed each time the value of the variable is changed.

```
11.28.48? CLEAR TRACE J
  J  NO LONGER TRACED.
11.29.57?X=45.5; J=99;
  X  = 4.55000000000000E+00
```

- Clear Trace on J
- Change values being traced
- Trace output
The ASK statement allows the user to interrupt program processing and accept input from the terminal. The ASK statement has two parameters. The first is the name of the variable which receives the user's input. It can be any simple variable type (STRING, BOOL, REAL, INT, KEY) or an expression consisting of these types or FUNCTIONS. The expression is evaluated and then assigned to the response variable name. The second parameter is an expression which is typed to the user as a "prompt" for input.

This expression may be the actual string expression enclosed in quotes or a string variable which has been previously defined. ISIS does not supply a separator between the printed output (prompt) and the user's typed input (response); therefore, the user should supply his own separation character(s) within the prompt definition if he desires to be able to discriminate between the prompt and his response. If, in typing the input, (response) to ASK, the user makes a typographical error, he may hit the BREAK key in which case ISIS will disregard what has been typed, indicate this action by typing 3 dots (...) and then reprompt the user for correct input.

EXAMPLE:

14.31.19?VAR SANS: STRING  
14.31.33?VAR IANS: INT  
14.31.44?VAR RANS: REAL  
14.31.53?VAR BANS: BOOL  
14.32.03?VAR PROMPT: STRING  
14.32.14?PROMPT='INPUT='  
14.32.30?ASK SANS,PROMPT  
INPUT=THIS IS A STRING INPUT!  
14.32.59?ASK IANS,PROMPT  
INPUT=1254  
14.33.25?ASK RANS,'INPUT REAL NO.'  
INPUT REAL NO.1.456732E+2  
14.34.15?ASK BANS,'INPUT BOOL VALUE'  
INPUT BOOL VALUE=TRUE  
14.34.57?PRINT SANS,'IANS,'RANS,'BANS'  
THIS IS A STRING INPUT!  
1254 1.4567320000000E+02  
TRUE

14.38.05?ASK SANS,'INPUT ?'  
INPUT ? NOTICE BLANK AFTER THE "!"  

14.41.33?ASK SANS,'TYPE NOW ->'  
TYPE NOW =AM TYING ERRRRR...  
TYPE NOW =WAS TYING ERROR

- Define users PROMPT variable.

- Interrupt program, PRINT prompt and wait for user to type response. (STRING, INT, REAL & BOOL)

- A blank is added by NOS system when you have uneven number of characters. This is due to the way the 6600 stores characters.

- User made error, then hit the BREAK key. ISIS prints ... and reprompts user for input.
Examples of expressions as input:

```plaintext
11.08.52?VAR A:REAL; VAR Y:REAL; VAR Z:REAL;
11.09.18;X=16.; Y=5.; Z=2.;
VAR X ; REAL
11.08.52?ASK SANS;'STRING INPT'
VAR SANS ; STRING
STRING INPT X+Y-3/Z
11.10.32?PRINT SANS
X+Y-3/Z

11.10.44?ASK RANS;'REAL INPT'
VAR RANS ; REAL
REAL INPT X+Y-3/Z
11.11.36?PRINT RANS
1.950000000000E+001

11.11.42?ASK RANS;'INPT'
INPTRANS+2
11.12.39?PRINT RANS
2.150000000000E+001
```

Array elements can be used in ASK statement:

```plaintext
08.27.13?VAR RAY:ARRAY[1..3] OF REAL
08.28.54?VAR IRAY:ARRAY[1..2] OF INT
08.29.19?VAR STRAY:ARRAY[1..41] OF STRING
08.29.42?
ASK IRAY[23]..'INPUT 2ND ELEMENT *'
INPUT 2ND ELEMENT * 635
08.30.26?
PRINTLN IRAY
[ 1 ] = 0
[ 2 ] = 635
```
The PRINT or PRINTLN statement evaluates each expression and prints its value. After printing the output, PRINT leaves the cursor at its current position, whereas PRINTLN advances the cursor to the beginning of the next line. The optional FORMAT is similar to that of PASCAL in that each expression may have its own format. The format may specify total field width, scaling factors or base conversions. Default formats are as follows:

- **REAL** numbers - an E format type with field width of 22 (ex. 003.1417 E+00)
- **INTEGER** numbers - a field width of 10 with all numbers right justified
- **STRING** - the field width is equal to the length of the complete string and string is left justified
- **BOOLEAN** - a field width of 10 and right justified

Variables that have been declared but not defined are assigned default values by the ISIS system. Integers and real variables are set equal to zero, Booleans are set to FALSE and strings are of zero length. Discussion of optional formats is supplied below.

**Format options**

- **:FORMAT1** - is the total field width. The expression is printed in the E format for real variables (PRINT x:10, y:15). It can be used with integer, real, and string variables. Where FORMAT1 is smaller than the number of characters in a string the field width is ignored and the complete string is printed. If FORMAT is greater than the string length, blanks are added until the string length specified by the format is printed.

- **:FORMAT1:FORMAT2** - sets up total field width (FORMAT1) (as described above) and defines the number of significant digits to the right of the decimal point. This means the variable is printed in a fixed format. This format applies only to real expressions.

**EXAMPLE:** (see next page)

```
09.50.10?VAR Z:ARRAY[1..3] OF INT
09.50.40?
Z[2]=126
09.51.07?PRINT Z
[ 1 ] = 0
[ 2 ] = 126
[ 3 ] = 0
```

- Print an array of numbers.
Calculators Programming Statements

PRINT continued

09.35.18?TYPE REC=RECORD NUM: INT; FLAG: BOO; NAM: STRING; END;
09.36.18?VAR X: INT
09.36.32?VAR Y: R; REAL;
09.36.42?VAR S: STRING
09.36.52?VAR B: BOOL
09.37.01?VAR VREC: REC
09.37.10?

PRINTLN S,''=S'',X,'''=X'',B,'''=B'',R,'''=R''
09.38.12?

PRINTLN VREC
NUM = 0
FLAG = FALSE
NAM =

09.38.36?X=15; Y=250,452;
09.39.16?PRINT SORT(X*2-5), X/2*10-4
5.0000000000000E+000 7.1000000000000E+001
09.39.53?PRINT SORT(X*2-5):20:4
5.0000

09.40.27?

PRINT X,X:20
15

09.40.59?

PRINT Y
2.5045200000000E+002
09.41.40?

PRINT Y:20
2.5045200000000E+002
09.41.55?

PRINT Y:20:10
250,452000000
09.42.18?

PRINT Y:20:5
250,4520000

12.13.27?

VAR I,,INT
12.15.05?REPEAT
12.15.39?PRINT I
12.15.51?PRINTLN SUB(S; I;1):2*I
12.16.18?I=I+1
12.16.27?UNTIL I=10
1 C
2 K
3 4
4 U N T I
5 6 7 8 9

Declarative statements

- Undefined variables printed to show their default values
- Print undefined record variables

(Please note default strings are of zero length (NULL))
- Print expressions

- Print INTEGRER (with and without format)

- Print REALS (with and without format)

- Prints I (default format)
- Prints a subset (I) of length 1 from the string S with a format of (2*I), which increases as the value of I increase
Library Access Statements

SET NAME [LIB].[SHELF].[BOOK].[CHAPTER].[PAGE]

The SET NAME statement assigns a name to the ACTIVE page. This allows the user to name a new ACTIVE page and to change the name of an existing ACTIVE page. The page name contains 5 levels for identification purposes. If the ACTIVE page has not been previously assigned a page name (SET NAME or USE) then each level of the name must be specified (SET NAME LIB . SHELF . BOOK . CHAPTER . PAGE). Otherwise, the page name may be abbreviated by specifying only the levels of the name which the user wishes to change. (SET NAME .. BK5 .. PGI sets the active page name to LIB . SHELF . BK5 . CHAPTER . PGI). Others remain the same as have been previously set.

When the user changes the name of a page and wishes to avoid retyping lower level parts of the name which do not change, he must type the DOT separator followed by a blank to replace the next lower level name. See 2nd, 3rd, and 5th examples.

EXAMPLE:

<table>
<thead>
<tr>
<th>SET NAME ALIB.MUST.ISIS.EDITOR.SPECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIB.MUST.ISIS.EDITOR.SPECS IS NAME OF ACTIVE.</td>
</tr>
<tr>
<td>SET NAME DATABASE.</td>
</tr>
<tr>
<td>ALIB.MUST.ISIS.DATABASE.SPECS IS NAME OF ACTIVE.</td>
</tr>
<tr>
<td>SET NAME MISC.NEWPOOP.</td>
</tr>
<tr>
<td>ALIB.MISC.NEWPOOP.DATABASE.SPECS IS NAME OF ACTIVE.</td>
</tr>
<tr>
<td>SET NAME BOYD</td>
</tr>
<tr>
<td>ALIB.MISC.NEWPOOP.DATABASE.BOYD IS NAME OF ACTIVE.</td>
</tr>
<tr>
<td>SET NAME COPY.</td>
</tr>
<tr>
<td>ALIB.COPY.NEWPOOP.DATABASE.BOYD IS NAME OF ACTIVE.</td>
</tr>
</tbody>
</table>

30
Library Access Statements

USE [LIB].[SHELF].[BOOK].[CHAPTER].[PAGE] [:{A1|A2|A3}]

The USE statement is used to read the contents of a specified page from the database library into the ACTIVE working page. In the case of the ACTIVE page, its name becomes what was specified in the USE statement and the contents become what was extracted from the page in the data base. The user also has the option of reading the page into one of the other working pages, A1, A2, or A3 (read only pages). This is done by following the library page name with a : and working page name (:A1).

EXAMPLE:

14.01.28?SHOW NAME
ALIB.MISC.NEWPOOP.DATABASE.SPECS IS NAME OF ACTIVE.

14.04.36?USE MUST.ISIS.
ALIB.MUST.ISIS.DATABASE.SPECS USED AS ACTIVE

14.05.27?USE EDITOR. :A3
ALIB.MUST.ISIS.EDITOR.SPECS USED AS A3

14.06.08?USE DATABASE.SOURCE :A1
ALIB.MUST.ISIS.DATABASE.SOURCE USED AS A1

14.07.49?USE EDITOR.
ALIB.MUST.ISIS.EDITOR.SOURCE USED AS ACTIVE
Library Access Statements

USING \{A1\{A2\{A3\} statement-verb

This statement allows the user to examine and use the read only working pages, A1, A2, and A3. Since A1, A2, and A3 are read-only pages, the "statement verb" is limited to the read only verbs shown below:

- LIST - List the working page (A1, A2, or A3)
- RUN - Add the working page (A1, A2, or A3) to the INPUT file
- EXEC - Execute the contents of the working page (A1, A2, or A3)
- COUNT - Count the number of lines in the working page (A1, A2, or A3)

The USING statement gives the user a way to examine pages of information without putting it in the Active page and thus destroying the ACTIVE page contents.

EXAMPLE:

10.39.0?USE JOELIB.ISIS.JOB.CONTROL.HEADER:A2
JOELIB.ISIS.JOB.CONTROL.HEADER USED AS A2

10.39.52?USING A2 LIST
1.  \=ISIS,1600,CH120006,
2.  \=USER:961300N,
3.  \=CHANGE:101481,LRC,

10.40.15?USING A2 COUNT
3 ITEMS IN SPECIFIED RANGE.
H=3, I=0

10.40.43?USING A2 RUN
3 ITEMS IN SPECIFIED RANGE.

10.41.27?SHOW RUN
3 LINES IN RUN.

10.41.52?USE JOELIB.ISIS.JOB.CONTROL.RUNISIS:A2
JOELIB.ISIS.JOB.CONTROL.RUNISIS USED AS A2

10.44.39?USING A2 RUN
3 ITEMS IN SPECIFIED RANGE.

10.44.58?SHOW RUN
6 LINES IN RUN.

10.45.16?USING A2 LIST
1.  \=ATTACH.ISIS,
2.  \=ISIS,
3.  =)
SAVE [*]

The SAVE statement is used to create a new page in the library or to replace an already existing page. SAVE places the ACTIVE page into the library under the same name as the current ACTIVE page name. The SAVE statement followed by an * will replace a page in the library of the same name as the ACTIVE page. At the present time, the user may replace a page (SAVE*) even if the page does not already exist. This is an error which will be corrected in the next version of ISIS. The user should be cautious in saving or replacing a page to make sure the ACTIVE page name is correct. The page name may be changed using the SET NAME statement before saving or replacing it.

EXAMPLE:

16.25.3?SET NAME ALIB.SHELF.BK.CH.CG
ALIB.SHELF.BK.CH.CG IS NAME OF ACTIVE,
16.26.1?SAVE
ALIB.SHELF.BK.CH.CG SAVED.
16.26.2?SET NAME XXX
ALIB.SHELF.BK.CH.XXX IS NAME OF ACTIVE,
16.26.5?SAVE
ALIB.SHELF.BK.CH.XXX SAVED.

- Save the ACTIVE page by storing it in the library under this name.
- Save the contents of the ACTIVE page under this page name in the library

16.27.1?SAVE*
ALIB.SHELF.BK.CH.XXX SAVED.

- Replace the contents on the already existing page of the library with a new version
Library Access Statements

PURGE pagename

The specified page is eliminated from the library. If the specified page is the only page in its chapter, the chapter is eliminated from the library. If this chapter is the only chapter in its book, then that book is eliminated from the library. Finally, if the book is the only book in its shelf, then that shelf is eliminated from the library.

EXAMPLE:

11.21.18?SET NAME ALIB,GRANT,BOOK,CHOP,PG2
ALIB,GRANT,BOOK,CHOP,PG2 IS NAME OF ACTIVE.
11.22.32?
SHOW PAGES ALIB...
ALIB .GRANT .BOOK .CHOP .PGALL
 . . . . . CCA
 . . . . . PG1
 . SHELF .BK .CH .CG
 . . . . . XXX
11.24.24?
PURGE PG2
ALIB,GRANT,BOOK,CHOP,PG2 PURGED.
11.25.33?
PURGE SHELF,CHAP,PAGE
ALIB,SHLF,BOOK,CHAP,PAGE PURGED.
11.26.06?
SHOW PAGES ALIB...
ALIB .GRANT .BOOK .CHOP .PGALL
 . . . . . CCA
 . . . . . PG1
 . SHELF .BK .CH .CG
 . . . . . XXX

- Set the ACTIVE page name
- Display the library ALIB
- Purge PG2
- Purge PAGE (resides on different shelf and chapter)
- Display the library to show there are 2 less pages in it now
Library Access Statements

DISPOSE [OF] \{ACTIVE|A1|A2|A3\}

The DISPOSE statement allows the user to get rid of the current name and contents of a working file.

EXAMPLE:

14.36.2?LIST
1. =A=10
2. =B=22
3. =X=A+B
4. =Y=X+A
5. =PRINTLN A,B,X,Y.
10. =A=ODD(B); PRINTLN B,A
12. =B0=ODD(BBBB); PRINTLN BBBB,B0
14.36.52?DISPOSE ACTIVE
14.37.11?LIST
NO ITEMS IN SPECIFIED RANGE.
The INSERT statement is used to add new lines to the ACTIVE page. A range must be specified and may be specified from the table below. The user is prompted for successive lines to fill the specified range. If an increment (inc) value is not included in the command range, then it is assumed all incrementing will be by 1.

Care should be taken when inserting lines between two existing lines. ISIS will not allow inserting a line on top of or beyond an already existing line. If the user inadvertently does this, ISIS will abort the command and all inserted lines up to this point will be placed in the text. Check the line increment (inc) and the number of lines to be inserted and make sure there is enough room between existing lines to accommodate the number you are planning to insert.

Range Options:

- \( \text{ln} \) - insert one line at line number \( \text{ln} \)
- \( \text{ln}(n\%l) \) - beginning with line number \( \text{ln} \) insert \( n\%l \) lines
- \( \text{ln}_1/\text{ln}_2 \) - insert lines between and including \( \text{ln}_1 \) and \( \text{ln}_2 \) incrementing by 1
- \( \text{ln}_1/\text{ln}_2(\text{lim}) \) - insert lines between \( \text{ln}_1 \) and \( \text{ln}_2 \) not to exceed \( \text{lim} \) number of lines
- \( \text{ln}/\text{inc} \) - insert lines beginning with \( \text{ln} \) and continue inserting while line numbers are being incremented by inc. Insertion can be halted by depressing the BREAK key. In the event the user tries to insert a line on top of an already existing line the ISIS system will print an error message and halt insertion.
- \( \text{ln}_1/\text{ln}_2/\text{inc} \) - insert lines between \( \text{ln}_1 \) and \( \text{ln}_2 \) incrementing the line number by inc.
- \( \text{ln}_1,\text{ln}_2,\text{ln}_3 \) - insert discrete line numbers - They do not have to be in any order.

**EXAMPLES:**

- INSERT 10 - Insert one line at 10
- INSERT 2,10,4,5 - Insert at line 2, line 10, line 4, and line 5
- INSERT 4//.2 - Insert lines beginning with line number 4 with line numbers incremented by .2 (4, 4.2, 4.4, etc)
DELETE  range  \[[:\{NL|NK|NI}\}\]

The DELETE statement is used to remove lines from the ACTIVE page. Deleted lines and their line number are printed unless the user selects an alternate display option below.

Range Options:

- \$n\$ - delete line number \$n\$
- \$n (n\$)\$ - beginning with line number \$n\$, delete \$n\$ lines.
- \$n_1/n_2\$ - delete all lines between and including \$n_1\$ and \$n_2\$.
- \$n_1/n_2(\#im)\$ - delete lines between and including \$n_1\$ and \$n_2\$ with a limit of \$\#im\$ number of lines (\$\#im\$ may be an expression).
- \$n_1,n_2,n_3\$ - delete discrete line numbers. They do not have to be in any order.
- ALL - delete all lines in the ACTIVE page.

Display Options:

- NL - does not list the contents of the line(s) being deleted or line number(s)
- NI - list the line number only
- NK - list contents of the line without the line number

EXAMPLES:

- DELETE 10 - Delete line number 10
- DELETE 15.1/25.2:NL - Delete all lines between 15.1 and 25.2 with no listing
- DELETE 5.5 (5) - Delete 5 lines beginning with line number 5.5
Text Editing Statements

REPLACE range [:NL|NK|NI]

The REPLACE statement is used to replace existing lines in the ACTIVE page. The user is prompted for successive lines to replace the lines in the specified range. The prompt is a display of the line to be replaced.

Range Options: $ln$ - replace line number $ln$

$ln$ (NL) - beginning with line number $ln$ replace $nl$ lines.

$ln_1/ln_2$ - replace all lines between and including $ln_1$ and $ln_2$

$ln_1/ln_2$ (lim) - replace lines from $ln_1$ through $ln_2$

not to exceed $lim$ number of lines

$ln_1$, $ln_2$, $ln_3$ - replace discrete line numbers. Line numbers do not have to be in any order

ALL - replace all lines in the ACTIVE page

Display Options: NL - Does not list the contents of the line(s) being replaced or line number(s)

NI - List the line number only

NK - List contents of the line without the line number

EXAMPLES:

REPLACE $2/5(3)$:NL - Replace line 2 through 5 but not more than 3 lines with no listing.

REPLACE $4(4)$:NI - Replace 4 lines beginning at line 4 and list the line numbers replaced.

REPLACE $4.1/4.9$ - Replace all lines between line 4.1 and line 4.9.
Text Editing Statements

CHANGE string-id TO string-id IN range [\{NL|NI|NK|ECHO\}]

The CHANGE statement is used to modify existing items on the ACTIVE page. A string variable which has been declared earlier or the actual string can be used (set off by quote ('') marks) in the command. The range must be included and must be one of these listed below.

Range Options:

- \(\ln\) - change string in line number \(\ln\)
- \(\ln (\ln)\) - beginning with line number \(\ln\), change string in \(\ln\) lines
- \(\ln_1/\ln_2\) - change string appearing in lines between and including \(\ln_1\) and \(\ln_2\)
- \(\ln_1/\ln_2 (\lnm)\) - change strings in lines from \(\ln_1\) through \(\ln_2\) not to exceed \(\lnm\) number of lines
- \(\ln_1, \ln_2, \ln_3\) - change strings in discrete line numbers - The line numbers do not have to be in any order
- ALL - change a string on all lines in ACTIVE page

Display Options:

- NL - Does not list the line(s) or line number(s) being changed.
- NI - List the line number only.
- NK - List contents of the line without the line number.
- ECHO - List old version of line and list new version of line.

EXAMPLES:

- CHANGE 'I > 3' TO 'I <= 3' IN 5 - Change string 'I > 3' to 'I <= 3' in line 5.
- CHANGE 'LINE' TO SVAR IN 2/10 - Change line to string variable SVAR in lines 2 through 10.
- CHANGE 'COUNTER' TO 'CNT' IN ALL:NL - Change the string 'COUNTER' to 'CNT' throughout entire page with no listing of changes.
Text Editing Statements

ADD  string-id  [AT column]  IN  range  [:\{NL\|NK\|NI\}]  

The ADD statement is used to alter existing lines on the ACTIVE page. The string to be added can be a string variable which has been declared earlier (SVAR) or the actual string can be used (set off by quote (') marks) in the command. If the AT column option is not specified, the string is appended to the end of the line. The altered line contents and line number are listed unless a display option has been specified. The range must be included and must be one of those listed below.

Range Options:  
\[n\]  - add string to one line at \[n\]  
\[n\](nl)  - beginning with line number \[n\]  add string to \[n\] lines.  
\[n_1\]/\[n_2\]/inc  - add string to lines between \[n_1\] and \[n_2\] incrementing by 1  
\[n_1\]/\[n_2\] (lim)  - add string to lines between \[n_1\] and \[n_2\] not to exceed \[lim\] number of lines  
\[n_1\], \[n_2\], \[n_3\]  - add string to discrete line numbers - Line numbers do not have to be in any order  
ALL  - add string to all lines in the ACTIVE page.

Display Options:  
NL - Does not list line(s) or line number(s) being altered.  
NI - List the line number only.  
NK - List contents of the line without the line number.

EXAMPLE:  
ADD 'STATUS' IN 5  - Add the string 'STATUS' to the end of line 5.  
ADD SVAR AT 5 IN 5.1  - Add the string variable SVAR at column 5 in line 5.1.  
ADD 'CONDITION' AT 9  
IN 5/10  - Add the string 'CONDITION' at column 9 beginning in line number 5 through line number 10.
Text Editing Statements

LIST [range] [:{NI|NK}]}

The LIST statement is used to view all or part of the ACTIVE page. The RANGE is optional and if it is not specified, the entire ACTIVE page will be listed.

Range Options:

- \( \ell n \) - list line number \( \ell n \).
- \( \ell n(n) \) - beginning with line number (\( \ell n \)) list \( n' \) number of lines
- \( \ell n_1/\ell n_2 \) - list all lines between and including \( \ell n_1 \) and \( \ell n_2 \)
- \( \ell n_1/\ell n_2(\ell m) \) - list lines between \( \ell n_1 \) and \( \ell n_2 \) with a limit of \( \ell m \) number of lines (\( \ell m \) may be an expression, \( 16\sin(A) \), etc)
- \( \ell n_1,\ell n_2,\ell n_3 \) - Discrete line listing
- ALL - list complete ACTIVE page

Display Option:

- NI - list the line number(s) only
- NK - list the contents of the line(s) without the line number(s)

EXAMPLES:

- LIST - List entire ACTIVE page
- LIST 11/15 - List everything between and including line 11 and line 15
- LIST 5, 4.5, 5.1 - List 3 lines, line 4, line 4.5, and line 5.1
Text Editing Statements

COUNT [range]

The COUNT statement allows the user to count the number of lines in the ACTIVE page within a specified range. The range is optional and, if not specified, the instruction will apply to the entire ACTIVE page.

Range Options:
- \( \ln_1/\ln_2 \) - count number of lines between \( \ln_1 \) and \( \ln_2 \)
- \( \ln_1/\ln_2, \ln_3/\ln_4 \) - count number of lines between 3 sets of line numbers
- \( \ln_5/\ln_6 \) - count number of lines between line number 4 and line number 10

EXAMPLE:

18.12.17 LIST

4. =ISIST51,T200,CM160000.
6. =USER,961300N.
8. =CHARGE,101481,LRC.
10. =GET,HALGO.
12. =GET,LST5=ATOM912.
14. =PACK,LST5.
16. =COPYBF,INPUT,INFIL.
18. =REWIND,INFIL.
20. =RFL,160000.
22. =REDUCE,=.
24. =NOEXIT.
26. =HALGO.
28. =*

18.12.31 COUNT
13 ITEMS IN SPECIFIED RANGE.
H=13; I=0
18.13.15 COUNT 4/10
4 ITEMS IN SPECIFIED RANGE.
H=4; I=0
18.13.34 COUNT 4/10,12/28
13 ITEMS IN SPECIFIED RANGE.
H=13; I=0

- List the ACTIVE page contents, a typical HAL/S applications program text
- EOR separator
- Count number of lines in ACTIVE page
- Count number of lines between line number 4 and line number 10
- Count total number of lines between line number 4 and line number 10 and between line number 12 and line number 28
Text Editing Statements

EXEC [range][;ECHO]

The EXEC statement allows the user to execute the ACTIVE page contents. Any portion of it may be executed by using the range option. It should be noted that declarative statements (VAR, TYPE, etc) cannot be executed but once. An error message is given if this occurs.

Range Options:  
\[ln\] - execute line number \(ln\)
\[ln_1/ln_2\] - execute lines between \(ln_1\) and \(ln_2\)
\[ln_1/ln_2\ (nl)\] - execute lines between \(ln_1\) and \(ln_2\) not to exceed \(nl\) number of lines

Display Option:  
ECHO - displays a line of code as it is executed

EXAMPLE:

14.04.5?LIST
1. \(=A=10\)
2. \(=B=11\)
3. \(=X=A+B\)
4. \(=Y=X+A\)
5. \(=PRINTLN A, B, X, Y\)
14.05.3?EXEC
1.000000000000000E+001  1.10000000000000E+001  2.10000000000000E+001  3.10000000000000E+001

14.12.3?EXEC ;ECHO
1. \(=A=10\)
2. \(=B=22\)
3. \(=X=A+B\)
4. \(=Y=X+A\)
5. \(=PRINTLN A, B, X, Y\)
1.000000000000000E+001  2.20000000000000E+001  3.20000000000000E+001  4.20000000000000E+001

- List ACTIVE page
- Execute the ACTIVE page
- Execute the ACTIVE page and print out each line of code as it is executed.
Text Editing Statements

EXEC continued

10.42.2: USE ALIB.ISIS.JOB.CONTROL.BUILD2
ALIB.ISIS.JOB.CONTROL.BUILD2 USED AS ACTIVE
10.43.12? LIST
   1. =USE ALIB.ISIS.JOB.CONTROL.HEADER:A3; USING A3 RUN:NK
   2. =USE ALIB.ISIS.JOB.CONTROL.COPY:A3; USING A3 RUN:NK
   3. =USE JOELIB.ISIS.SOURCE.PASINTF.SEPT18:A3; USING A3 RUN
10.43.21? USE COPY:A1
ALIB.ISIS.JOB.CONTROL.COPY USED AS A1
10.44.32? USING A1 LIST
   1. =ATTACH,ISISC10/M=W.
   2. =REWIND,ISISC10.
   3. =COPYEI,INPUT,ISISC10.
   4. =*
10.44.45?
EXEC
ALIB.ISIS.JOB.CONTROL.HEADER USED AS A3
3 ITEMS IN SPECIFIED RANGE.
ALIB.ISIS.JOB.CONTROL.COPY USED AS A3
4 ITEMS IN SPECIFIED RANGE.
JOELIB.ISIS.SOURCE.PASINTF.SEPT18 USED AS A3
699 ITEMS IN SPECIFIED RANGE.
10.46.28?
SEND
'AIVIEEP' SENT TO BATCH EXECUTION.

- Read a library page into the ACTIVE page
- List ACTIVE page
- Read a library page into the Al working page
- List the Al page contents
- Execute the ACTIVE page
- USE statement executed
- RUN statement executed
- Second line
- Third line
- INPUT file (generated by RUN) is sent to the CDC 6600 computer
Tool Invocation Statements

RUN [range] [:ECHO]

The RUN statement allows the user to create an input file (control cards, program source and data) for submission to the NOS internal reader. An * is used in place of an EOR mark for separating records such as the control cards, program source, and data on the INPUT file. If control cards are stored on one page, program source on another and data on still another, the user simply executes the RUN statement 3 times in succession. RUN will concatenate the indicated page or page range to the INPUT file for each time it is executed. The range is optional and if not specified will be the entire working page.

Range Options:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ln</td>
<td>- add line number ln to INPUT file</td>
</tr>
<tr>
<td>ln(nl)</td>
<td>- beginning with line number ln add n lines to the INPUT file</td>
</tr>
<tr>
<td>ln1/ln2</td>
<td>- add lines between ln1 and ln2 to the INPUT file</td>
</tr>
<tr>
<td>ln1/ln2(lim)</td>
<td>- add lines between ln1 and ln2 not to exceed lim number of lines to INPUT file</td>
</tr>
<tr>
<td>ln1, ln2, ln3</td>
<td>- add discrete line numbers to INPUT file - the line numbers do not have to be in any order</td>
</tr>
</tbody>
</table>

Display Option: ECHO - displays a line of code as it is executed.

18.18.33?USE ALIB.ISIS.JOB.CONTROL.BUILD2 ALIB.ISIS.JOB.CONTROL.BUILD2 USED AS ACTIVE
18.19.03?LIST
  1. =USE ALIB.ISIS.JOB.CONTROL.HEADER:A3 USING A3 RUN:NK
  2. =USE ALIB.ISIS.JOB.CONTROL.COPY2:A3 USING A3 RUN:NK
  3. =USE JOELIB.ISIS.SOURCE.PASINTF.SEPT18:A3 USING A3 RUN
18.19.23?USE COPY:A1 ALIB.ISIS.JOB.CONTROL.COPY USED AS A1
18.19.50?LIST
  1. =USE ALIB.ISIS.JOB.CONTROL.HEADER:A3 USING A3 RUN:NK
  2. =USE ALIB.ISIS.JOB.CONTROL.COPY2:A3 USING A3 RUN:NK
  3. =USE JOELIB.ISIS.SOURCE.PASINTF.SEPT18:A3 USING A3 RUN
18.19.58?EXEC
ALIB.ISIS.JOB.CONTROL.HEADER USED AS A3 3 ITEMS IN SPECIFIED RANGE.
ALIB.ISIS.JOB.CONTROL.COPY2 USED AS A3 5 ITEMS IN SPECIFIED RANGE.
JOELIB.ISIS.SOURCE.PASINTF.SEPT18 USED AS A3 699 ITEMS IN SPECIFIED RANGE.
18.20.37?
RUN continued.

EXAMPLE:

10.53.23 USE ALIB.GRANT.BOOK.CHOP.CCA:A1
ALIB.GRANT.BOOK.CHOP.CCA USED AS A1
10.54.06 USING A1 LIST
4. =ISISTST,T200,CM160000,
6. =USER:961300N,
8. =CHARGE,101481,LRC.
10. =GET;HALGO.
12. =GET;LST5=ATOM912.
14. =PACK;LST5.
16. =COPYBF;INPUT;INFIL.
18. =REWIND;INFIL.
20. =RFL;160000.
22. =REDUCE,-.
24. =NOEXIT.
26. =HALGO.
28. =*
10.54.21 USE PG1
ALIB.GRANT.BOOK.CHOP.PG1 USED AS ACTIVE
10.54.57 LIST
2. =GPROC;PROGRAM;
4. =C HALMAT TEST CASE ARRAYS
6. =C R,S,T BEING INTEGER ARRAYS OF 10
8. =C U BEING A 5X5 INTEGER ARRAY
10. = DECLARE R ARRAY(10) INTEGER;
12. = DECLARE S ARRAY(10) INTEGER;
14. = DECLARE T ARRAY(10) INTEGER;
16. = DECLARE U ARRAY(10) INTEGER;
18. = DECLARE U ARRAY(5,5) INTEGER;
20. = DECLARE INTEGER,A,B,C,D;
22. = A=20;
24. = B=18;
26. = C=16;
28. = R$(5)=10;
30. = R$(7)=12;
32. = U$(3,4)=15;
34. = D=(A+B)+C;
36. = D=A+(B+C);
38. = A=R$(5);
40. = S$(2)=R$(7);
42. = A=U$(3,4);
44. = U$(4,3)=A;
46. = CLOSE GOPROC;
48. =*
10.56.58 USING A1 RUN
13 ITEMS IN SPECIFIED RANGE.
10.57.19 USING ACTIVE RUN
24 ITEMS IN SPECIFIED RANGE.
10.57.42? SHOW RUN
37 LINES IN RUN.
10.57.54? SEND 'AWVIKXX' SENT TO BATCH EXECUTION.
Tool Invocation Statements

SEND

The SEND command allows the user to submit the INPUT file to the NOS interval reader. NOS will respond by printing a 7 character identification code for the job. If the INPUT file happened to be empty, then a message is typed indicating this.

EXAMPLE:

```
14.57.38?SEND
 'AWVIOYA' SENT TO BATCH EXECUTION.
14.57.59?             NOS accepts job and returns
                        the job ID name 'AWVIOYA'

15.48.55?SEND
NOTHING TO SEND.
COMMAND ABORTED. (ADDRESS: 8)        Empty INPUT file
```
SHOW SHOWS

A self-help type of command. The SHOW SHOWS command prints a list of all statements which the user may SHOW, or the words he may select to follow the SHOW command verb.

EXAMPLE:

<table>
<thead>
<tr>
<th>09.16.55?SHOW SHOWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREYS</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>*PROCS</td>
</tr>
<tr>
<td>→ TRAPS</td>
</tr>
</tbody>
</table>

*Not available at present.
→ Not available to user
**Interrogation Statements**

**SHOW RESERVED**

The SHOW RESERVED statement is used to determine what words have been reserved for ISIS and cannot be used as identifier names in user programs.

**EXAMPLE:**

```
09.28.02?SHOW RESERVED
ABBREV ADD ALL AND ARRAY ASK
AT BY CALL CHANGE CLEAR
COMPILE COPY COUNT DELETE
DOWNTO DO DUMP ELSE
EXEC EXITIF FALSE FOREACH
IF INSERT INVOKE IF
MODIFY MOVE IN NOT
OR OVER PRINTLN PRINT
READ RECORD REMOVE REPLACEMENT
RUN SAVE SET REPEAT
STORE THEN TO RESTORE
UNION UNTIL USE SHOW
WRITE XEQ USING VAR
WHILE
```

**SHOW STATEMENTS**

The SHOW STATEMENTS statement is used to display all available statement verbs in the ISIS system.

**EXAMPLE:**

```
09.28.38?SHOW STATEMENTS
ABBREV ADD ASK *CALL CHANGE CLEAR
*DUMP COUNT DELETE
*EXEC FOREACH FOR
*IF LIST LOOP
*MOVE MODIFY
*READ PROC PURGE
*REPEAT READ
*REMOVE RESTORE Restore
*SEND SET RUN
*TYPE SHOW SAVE
*WRITE WHILE
```

*Not implemented at present
SHOW AVAIL

This statement allows the user to obtain an estimate of program resources. The resource space already in use and remaining available are given in words where 10 characters are stored per word. Program resources listed are:
- string variable space (STRING),
- program types (TYPE),
- program variables (ID),
- procedures and function/names (PF), and
- system information, not directly controllable by the user (STACK, CORE, and XEQ).

EXAMPLE:

```
09.30.11?SHOW AVAIL
IN USE AVAILABLE
---------------------
STRING: 31 119
TYPE: 11 52
ID: 34 116
PF: 0 15
STACK: 14 986
CORE: 15 385
XEQ: 0 20
```

SHOW ABBREVS

The SHOW ABBREVS statement is used to determine which statement verbs have been abbreviated by the user and their abbreviation(s).

EXAMPLE:

```
09.28.17?SHOW ABBREVS
**NONE**
09.29.12?ABBREV P;PR:PRINT
09.29.35?SHOW ABBREVS
'PR':PRINT
'P':PRINT
```
Interrogation Statements

SHOW TYPES

The SHOW TYPES statement is used to allow the user to display all types which have been placed in the type-id table. If the user wishes to inquire about one specific type, he should use the SHOW ID statement.

EXAMPLE:

14.18.40?SHOW TYPES
'BOOLEAN' := BOOLEAN;
'INTEGER' := INTEGER;
'KEYWORD' := KEYWORD;
'REAL' := REAL;
'STRING' := STRING;
14.19.51?TYPE PERSONS:REAL
14.20.08?TYPE ADDR:RECORD NUM:INTEGER NAME:STRING END;
14.21.35?
SHOW TYPES
'ADDR' := RECORD
   NUM: INTEGER;
   NAME: STRING;
END;
'BOOLEAN' := BOOLEAN;
'INTEGER' := INTEGER;
'KEYWORD' := KEYWORD;
'PERSONS' := REAL;
'REAL' := REAL;
'STRING' := STRING;

- Initial data types provided by ISIS
- Add new types
- New types have been added
Interrogation Statements

SHOW VARS

The SHOW VARS statement is used to determine which variables have been placed in the identifier table. All declared variables are printed. If the user wishes to inquire about one specific variable he should use the SHOW ID statement.

EXAMPLE:

14.00.47?SHOW VARS
** NONE **

14.02.45?TYPE PERSONS:REAL
14.03.03?TYPE MESS : ARRAY[1..5] OF BOOL
14.03.40?TYPE REC1 : RECORD NUM:INT; FLAG:BOOL; NAM:STRING; END;
14.04.28?TYPE REC2 : ARRAY[1..3] OF REC1;
14.05.16?

VAR ST1,ST2:STRING
14.05.32?VAR ABC:REAL
14.05.44?VAR B1,B2:BOOL
14.05.56?VAR KIM,KATE:PERSONS
14.06.12?VAR AREC: REC1
14.06.23?VAR BREC: REC2
14.06.37?

SHOW VARS

'ABC   ': REAL;
'AREC   ': RECORD
    NUM: INT;
    FLAG: BOOL;
    NAM: STRING;
END;
'BREC   ': ARRAY [1..3] OF
    RECORD
    NUM: INT;
    FLAG: BOOL;
    NAM: STRING;
END;
'B1     ': BOOL;
'B2     ': BOOL;
'KATE   ': REAL;
'KIM    ': REAL;
'ST1    ': STRING;
'ST2    ': STRING;
**Interrogation Statements**

SHOW ID \{variable-id|type-id|system-id\}

The SHOW ID statement allows the user to obtain a description of program or system identifiers. The identifier name, type, and usage is displayed.

**EXAMPLE:**

```
09.50.52?VAR X:REAL
09.51.36?VAR B,C:BOOL:
09.52.06?VAR I,J,K:INT
09.52.24?VAR IND:STRING
09.52.40?VAR AREC:RECORD NUM:INT; FLG:BOOL; END;
09.53.10?VAR RAY:ARRAY[1..3] OF REAL;
09.53.35?TYPE PERSONS=REAL

10.03.56?AREC.NUM=22;
09.59.49?J=11
10.00.11? RAY[2]=125.35;
10.01.31?
```

```plaintext
VARIABLE
  'X'  : REAL;
VARIABLE
  'B'  : BOOL;
VARIABLE
  'J'  : INT;
VARIABLE
  'RAY' : ARRAY [1..3] OF REAL;
VARIABLE
  'AREC' : RECORD
       NUM: INT;
       FLG: BOOL;
       END;

TYPE
  'PERSONS' : REAL;
```

**NOTE:** ISIS is printing : instead of .. at present. This will be corrected in the next version of ISIS.
Interrogation Statements

SHOW SETS

The SHOW SETS statement displays all items the user may SET. This SET statement may be voided by the CLEAR statement.

EXAMPLE:

09.29.42?SHOW SETS
INDEX NAME TRACE TRAP

SHOW CLEARS

The SHOW CLEARS statement is used to determine which items can be All SET commands ARE voided by using the CLEAR command.

EXAMPLE:

09.29.54?SHOW CLEARS
INDEX NAME RUN TRACE TRAP

SHOW NAME

The SHOW NAME displays the current name of the ACTIVE page.

EXAMPLE:

12.24.20?SHOW NAME
... 
12.24.39?SET NAME ILIB.SHEL.BK.CHAP.PG1
ILIB.SHEL.BK.CHAP.PG1 IS NAME OF ACTIVE.
12.25.12?SAVE
ILIB.SHEL.BK.CHAP.PG1 SAVED.
12.25.27?
SHOW NAME
ILIB.SHEL.BK.CHAP.PG1
- ACTIVE page has not yet been named
- Set name of ACTIVE page
- Show name of ACTIVE page
SHOW PAGES [LIB].[SHELF].[BOOK].[CHAPTER].[PAGE]

The SHOW PAGES statement is used to display the structure of a complete library or a portion of a library. To display a complete library the user need only type the top level of the library name followed by 4 dots separated by blanks (SHOW PAGES ALIB ...). All pages for all levels of the library will be displayed. If the library name is not included in the SHOW PAGES statement, it will use the library name of the ACTIVE page. This is the only level at which an assumption is made. If other level names are not included in the statement, then the library is searched for the specific combination of these levels which are specified and all combinations are listed. The SHOW PAGES may also be used to search for certain page names. For example, to find all page names in a specific chapter, the user would type the following:
SHOW PAGES LIB.SHL.BK.CHAPTER. To find all pages named SAM on a particular shelf, the user would type: SHOW PAGES .SHL... SAM. All books and chapters on that shelf would be searched for pages named SAM. The name specified in SHOW PAGES can thus be used to define an area of search for all pages or for specifically named pages within a desired region.

Note that this command (SHOW PAGES) does not in any way affect the name of the ACTIVE page. This command only displays the names of pages of a library that have previously been saved.

EXAMPLE:

14.15.41?SET NAME ALIB.MUST.ISIS.EDITOR.SPECS
ALIB.MUST.ISIS.EDITOR.SPECS IS NAME OF ACTIVE.
14.16.38?SHOW PAGES. ... ...
ALIB .GRANT .BOOK .CHOP .PG2
* ... * ... PGALL
* ... * ... CCA
* ... * ... PG1
.SHELF .BOOK .CHAP .PAGE
* ... * ... BK ... CH ... CG
* ... * ... XXX
.MUST .ISIS .EDITOR .SPECS
* ... * ... .USERDOC
* ... * ... .SOURCE
* ... * ... .MAINTSP
* ... * ... .OBJECT
* ... * ... .DATABASE.USERDOC
* ... * ... .SPEC
* ... * ... .SOURCE
* ... * ... .CMDLST
.MISC .NEWPOOP.DATABASE.SOURCE
* ... * ... .CMDLST
* ... * ... .SPEC

- Set ACTIVE page name
- Display the entire library (using the library name of ACTIVE page)
SHOW PAGES continued

14.18.32?SHOW PAGES ALIB.GRANT... - Show all pages contained in
   ALIB .GRANT .BOOK .CHOP .PG2
   . . . . .PGALL
   . . . . .CCA
   . . . . .PG1

14.18.54?SHOW PAGES SOURCE - Show where the SOURCE pages
   ALIB .MUST .ISIS .EDITOR .SOURCE
   . . . . .DATABASE .SOURCE
   .MISC .NEWPOOP .DATABASE .SOURCE

   ALIB .MUST .ISIS .DATABASE .USERDOC .SPECS
   . . . . .SOURCE
   . . . . .CMOLST

14.21.18?SHOW PAGES .ISIS .SPECS - Show all chapters containing
   ALIB .MUST .ISIS .EDITOR .SPECS
   . . . . .DATABASE .SPECS

SPECS residing in the ISIS book
SHOW OPTIONS

The SHOW OPTIONS statement is used to determine the print options which are available for some editor commands. These options allow the user to modify the printed output resulting from the editor statements.

EXAMPLE:

09.20.48?SHOW OPTIONS
ECHO NI NK NL *M *T

* Not implemented at present

SHOW COLUMNS

The SHOW COLUMNS statement is used to show position of code in a line. A line of 59 characters is printed. This line is formed by repeating the character sequence "123456789." A source line is then listed using LIST and its column positions determined by alignment with the numbers.

EXAMPLE:

12.38.55?SHOW COLUMNS
123456789.123456789.123456789.123456789.123456789.123456789
12.39.14?LIST 1
1. =ISISC,T2000,UM699000.
RM1150 GRANTHAM-ISIS

This is column 40 in this line of code
Interrogation Statements

SHOW RUN

The SHOW RUN statement determines the number of lines of code in the INPUT file and then prints the number.

EXAMPLE:

```
13.39.49? LIST
  4. =ISISTST,T200,CM160000.
  6. =USER,961300N.
  8. =CHARGE,101481,LRC.
 10. =GET,HALGO.
 12. =GET;LST5=ATOM912.
 14. =PACK;LST5.
 16. =COPYBF;INPUT,INFILE.
 18. =REWIND;INFILE.
 20. =RFL,160000.
 22. =REDUCE; ...
 24. =NOEXIT.
 26. =HALGO.
 28. =*
 30. = GOPROC%PROGRAM;
 32. = DECLARE R ARRAY(10) INTEGER;
 34. = DECLARE S ARRAY(10) INTEGER;
 36. = DECLARE T ARRAY(10) INTEGER;
 38. = DECLARE U ARRAY(5,5) INTEGER;
 40. = DECLARE INTEGER;A,B,C,D;
 42. = DECLARE INTEGER;A,B,C,D;
 44. = A=20; B=18; C=16;
 46. = R[5]=10;
 50. = U[3,4]=15;
 52. = D=(A+B+C);
 54. = D=A+B+C;
 56. = A=R[5];
 58. = CLOSE GOPROC;
 60. =*
 62. =
```

- Clear INPUT file
- Put lines 4 through 24 of the ACTIVE page into INPUT file
- Add lines 28 through 60 to INPUT files
- Display the total number of lines in the INPUT file

LaRC/Control card file for compiling and executing a typical HAL/S program.

Typical HAL/S application program text

EOR separator

EOR separator

58
### INDEX

<table>
<thead>
<tr>
<th>Command/Function</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREV command</td>
<td>12, 15</td>
</tr>
<tr>
<td>abbreviations, IPL</td>
<td>11</td>
</tr>
<tr>
<td>ABBREVS, SHOW</td>
<td>50</td>
</tr>
<tr>
<td>ABS function</td>
<td>14</td>
</tr>
<tr>
<td>ACTIVE page</td>
<td>6, 12, 13, 30-33, 35</td>
</tr>
<tr>
<td>ADD</td>
<td>12, 40</td>
</tr>
<tr>
<td>.ALARM</td>
<td>8</td>
</tr>
<tr>
<td>ADD</td>
<td>14</td>
</tr>
<tr>
<td>ARCTAN function</td>
<td>14</td>
</tr>
<tr>
<td>Available files</td>
<td>6</td>
</tr>
<tr>
<td>AVAIL, SHOW</td>
<td>50</td>
</tr>
<tr>
<td>ASK</td>
<td>12, 26</td>
</tr>
<tr>
<td>A1 page</td>
<td>6, 12, 31, 32, 35</td>
</tr>
<tr>
<td>A2 page</td>
<td>6, 12, 31, 32, 35</td>
</tr>
<tr>
<td>A3 page</td>
<td>6, 12, 31, 32, 35</td>
</tr>
<tr>
<td>BOOLEAN(BOOL) type</td>
<td>7, 14, 28</td>
</tr>
<tr>
<td>BREAK key</td>
<td>11, 12, 13, 22, 26</td>
</tr>
<tr>
<td>Calculator statements</td>
<td>15-29</td>
</tr>
<tr>
<td>CAT function</td>
<td>14, 24</td>
</tr>
<tr>
<td>CHANGE</td>
<td>12, 39</td>
</tr>
<tr>
<td>CLEAR</td>
<td>25</td>
</tr>
<tr>
<td>CLEAR RUN</td>
<td>58</td>
</tr>
<tr>
<td>CLEAR, SHOW</td>
<td>57</td>
</tr>
<tr>
<td>CLEAR TRACE</td>
<td>12, 25</td>
</tr>
<tr>
<td>.CLOCK</td>
<td>8</td>
</tr>
<tr>
<td>col, IPL abbreviation</td>
<td>11</td>
</tr>
<tr>
<td>COLUMNS, SHOW</td>
<td>57</td>
</tr>
<tr>
<td>COS function</td>
<td>14</td>
</tr>
<tr>
<td>COUNT</td>
<td>6, 12, 42</td>
</tr>
<tr>
<td>.DATE</td>
<td>8</td>
</tr>
<tr>
<td>DELETE</td>
<td>12, 37</td>
</tr>
<tr>
<td>.DELTA</td>
<td>8</td>
</tr>
<tr>
<td>DISPOSE</td>
<td>12, 35</td>
</tr>
<tr>
<td>DIV operator</td>
<td>14</td>
</tr>
<tr>
<td>DOWNTO</td>
<td>12, 21</td>
</tr>
<tr>
<td>ECHO</td>
<td>39, 43, 45</td>
</tr>
<tr>
<td>ELSE</td>
<td>20</td>
</tr>
<tr>
<td>ERASE</td>
<td>12, 18</td>
</tr>
<tr>
<td>EXEC</td>
<td>6, 12, 43</td>
</tr>
<tr>
<td>EXITIF</td>
<td>12, 19, 22</td>
</tr>
<tr>
<td>EXP function</td>
<td>14</td>
</tr>
<tr>
<td>functions, IPL</td>
<td>14</td>
</tr>
<tr>
<td>files, available</td>
<td>6</td>
</tr>
<tr>
<td>FOR</td>
<td>12, 21</td>
</tr>
<tr>
<td>FORMAT1, FORMAT2</td>
<td>28</td>
</tr>
</tbody>
</table>
INDEX cont'd

id, IPL abbreviation 11
ID, SHOW 53
IF 7, 12, 20
inc, IPL abbreviation 11, 36
INSERT 12, 36
INTEGER (INT) type 7, 14, 28
Interrogation Statements 12, 48-57
IPL/PASCAL Differences 7
IPL Statement Summary 12
ISIS 6, 7

KEY 7

LEN function 14
library 6, 32
Library Statements 12, 30-35
%im, IPL abbreviation 11, 36-46
LIST 6, 12, 41
LN function 14
%n, IPL abbreviation 11, 36-46
LOC function 14
LOOP 12, 22
loop statements 7

MOD function 14

NAME 12, 35, 36
NAME, SHOW 54
n%, IPL abbreviation 11, 36-46
NL, NK, NI options 12, 37-41

ODD function 14
Operators 14
Options, list 12, 37-41
OPTIONS, SHOW 57
OR 14
ORD function 14

Page, available, A1, A2, A3, ACTIVE 6
PAGES, SHOW 55
PASCAL differences 7
PRINT 12, 28
PRINTLN 12, 28
prompt 11
PURGE 12, 34
INDEX cont'd

range 36-46
REAL type 7, 14, 28
record, IPL 16
record, SYSTEM 8
REPEAT 12, 23
REPLACE 12, 38
RESERVED, SHOW 49
Reserved words 49
ROUND function 14
RUN command 6, 12, 45
RUN, SHOW 58
SAVE 33
SEND 12, 47
SET NAME 12, 30
SET TRACE 12, 25
SETS, SHOW 54
SHOW ABBREVS 50
SHOW AVAIL 50
SHOW CLEARS 54
SHOW COLUMNS 57
SHOW ID 53
SHOW NAME 57
SHOW OPTIONS 55
SHOW PAGES 55
SHOW RESERVED 49
SHOW RUN 58
SHOW SETS 54
SHOW SHOWS 48
SHOW STATEMENTS 49
SHOW TYPES 51
SHOW VARS 52
SHOWS, SHOW 48
SIN function 14
SQRT function 14
SQR function 14
Statement Summary 12
STATEMENTS, SHOW 49
STOP 5
STRING type 7, 14, 28
SUB 14
Syntax, IPL 10
System-id 53
SYSTEM.
  SYSTEM.ALARM 8
  SYSTEM.CLOCK 8
  SYSTEM.DATE 8
  SYSTEM.DELTA 8
  SYSTEM.TIME 8
  SYSTEM.VERBOSE 8
System variables 8, 9
<table>
<thead>
<tr>
<th>Feature</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Editing Statements</td>
<td>12, 36-44</td>
</tr>
<tr>
<td>Text editor</td>
<td>6</td>
</tr>
<tr>
<td>.TIME</td>
<td>8</td>
</tr>
<tr>
<td>Tool Invocation Statements</td>
<td>12, 45-57</td>
</tr>
<tr>
<td>TRUNC function</td>
<td>14</td>
</tr>
<tr>
<td>TYPE</td>
<td>12, 16</td>
</tr>
<tr>
<td>TYPES, SHOW</td>
<td>51</td>
</tr>
<tr>
<td>type-id</td>
<td>16, 53</td>
</tr>
<tr>
<td>UNTIL</td>
<td>12, 23</td>
</tr>
<tr>
<td>USE</td>
<td>12, 31</td>
</tr>
<tr>
<td>USING</td>
<td>12, 32</td>
</tr>
<tr>
<td>VAR</td>
<td>17</td>
</tr>
<tr>
<td>Variable-id</td>
<td>17, 53</td>
</tr>
<tr>
<td>VARS, SHOW</td>
<td>52</td>
</tr>
<tr>
<td>.VERBOSE</td>
<td>8</td>
</tr>
<tr>
<td>WHILE</td>
<td>12, 23</td>
</tr>
<tr>
<td>Working pages</td>
<td>6</td>
</tr>
<tr>
<td>XEQ</td>
<td>24</td>
</tr>
</tbody>
</table>
References

### Abstract

The Interactive Software Invocation System (ISIS) is an interactive data management system. ISIS is being developed to provide the user with a powerful system for developing software in an interactive environment. ISIS will protect the user from the idiosyncrasies of the host computer system by providing a complete range of capabilities including desk top calculator, data and text editor, file manager, and tool invoker. The user should have no need for direct access to the host computing system. This documentation covers the operational concepts and syntax of the Interactive Programming Language, IPL, for ISIS.