Semantic Update System (ISUS) For FORTRAN

System Name: ISUS (Interactive Version)

U S E R ' S G U I D E

RM-661/3

February 1980

Software Research Associates
P. O. Box 2432
San Francisco, CA 94126 USA

(415) 957-1441

Semantic Update System (ISUS) For FORTRAN

System Name: ISUS (Interactive Version)

U S E R ' S G U I D E

Software Research Associates
P. O. Box 2432
San Francisco, CA 94126 USA

Keywords: LAFR, MTTO, PGLS, SS

LIST OF FIGURES

ACKNOWLEDGEMENTS

This document was prepared by Mr. Eben Sprinsock, with assistance from Dr. Edward Miller, Ms. Elizabeth Walker, Ms. Cynthia Depp, and Mr. Jim Henderson.

1 INTRODUCTION

This manual describes the use and operation of a special software maintenance tool, the Interactive Semantic Update System (ISUS). The purpose of ISUS is to provide a tailored environment for modification and update of FORTRAN programs that automates many of the functions previously done only by hand if at all.

1.1 Background

The ISUS development project began in early 1978 when an SRA research project produced an initial specification for a Semantic Update System (See Ref. 2). The goal of that effort was to produce a system that has these features:

- It should incorporate the normal features of a source code control system, like CDC's UPDATE
- It should be capable of accepting user commands to make changes in a FORTRAN software system and:
  - (1) Perform complicated changes automatically
  - (2) Indicate to the user when "side effects" or "consequences" were found
- It should be a freestanding system not reliant on other processors.

At the same time, we were able to identify a number of command classes that made it evident that the simple features of an Editor would never be sufficient to do the job asked of the Interactive Semantic Update System.

The Interactive Semantic Update System has been developed in stages. A first system, ISUS1, implemented only the most trivial update functions. This manual provides complete documentation for the ISUS2 system, which represents the first milestone in the long-range project. ISUS2 has many -- but not all -- of the features that are described in the long-term specification document (See Ref. 9). ISUS2's features are:

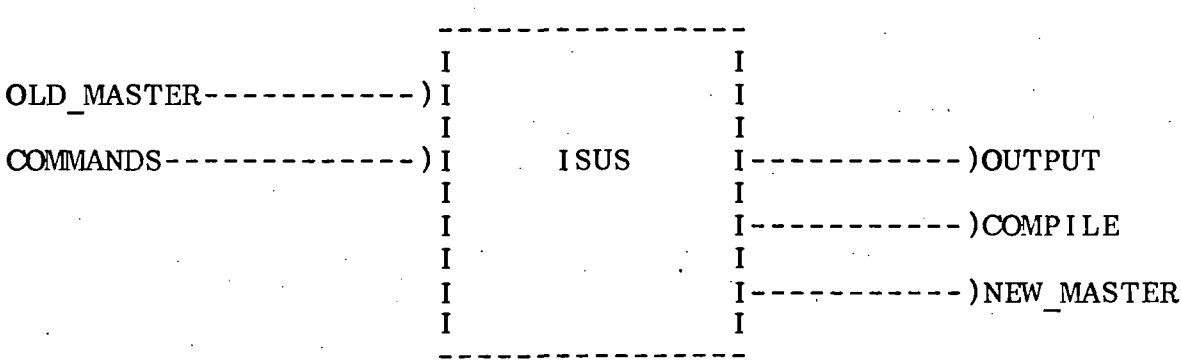
- It processes X3.9 FORTRAN
- It handles basic source code control updating -- non-Semantic update mode
- It processes approximately half of the "final" set of commands envisioned for ISUS
- It can compute some -- but not all -- of the side effects the commands produce; these side effects are described in detail in the remainder of this manual

1.2 Updating System Principles

The basic objective of an update system is to control the evolution of a software system as it is modified, upgraded, changed and otherwise "managed" from one state to another. Most update systems -- like ISUS -- have the following major facets:

- A master file, on which the "original" source programs in text format are kept
- The "system," which in the case of ISUS comprises a number of software components that in aggregrate serve the user by permitting various kinds of modifications
- A compile-file output, onto which the update system writes new programs ready to be compiled
- A new-master file, which serves as a future master file when modifications to an old master file have been completed and checked out
- An output file, on which the system writes responses
- An input file, sometimes called the command file, on which the user's requests are written

ISUS has all of these, as the following diagram indicates:



This is the standard structure of ISUS operation; not every file must be used, but at least the two on the left must be present -- the commands and the old master are needed for the system's operation.

Any new master file is compatible with ISUS so the new master can later be used as the old master. A special program is used to convert from standard FORTRAN input format (like that fed to the compiler) to the old/new master format. The ISUS master file format is described in Reference 11.

1.3 Structure of Manual

This manual, which assumes a reader reasonably familiar with update principles, is laid out as follows.

Section 2 describes a typical ISUS session and describes some of the features of the system as it is intended to be used. Section 3 discusses the ISUS library system and how it is used.

Automated consequence analysis for ISUS is described in Section 4. For ISUS, which this manual describes, the automated consequence analysis is limited to single-module analysis only.

Section 5 describes implementation restrictions that apply to this version of the system.

Section 6 provides a complete description of the format used in the command descriptions that appear as the second part of this manual.

Section 7 provides the input/output of the system for a complete series of modifications. This is a lengthy example and can be used to study how to best employ ISUS in automated program maintenance.

2 TYPICAL ISUS SESSION

This section describes a typical ISUS session. The reader is assumed to be familiar with:

(1) The FORTRAN language

(2) Fundamental conepts of source code maintenance systems

These assumptions are augmented by the environmental considerations given below, and also by installation dependent considerations (see attached installation dependent Memorandum).

2.1 Environment

This discussion will be based on some assumptions about the "setup" for the typical session, as follows:

- The old master has been created in the past (i.e. there have been previous ISUS runs done on it) and the file is available for use.
- The user has an idea of the intended changes to be made to the FORTRAN system (remember, ISUS is not an editor, and many of the commands are very powerful)
- ISUS is being used in "batch" mode.

With this as a context, we can see that a "standard" session is going to involve the following sorts of activities (each of which is described in more detail later on):

(This is shown on the next page so that the structure can be visualized in its entirety.)

```
COMMENT Sample Session

COMMENT (1)  Signon Section

  START
  ID = name
  PASSWORD = password
  ...other signon commands...

COMMENT (2)  Basic Modification Loop

  COMMENT (2.1)  Module Selection Section

    MODULE = name

  COMMENT (2.2)  Module Modification Section

    ----------------------------------------
    Commands that modify the Module...
    ...any of the ISUS commands described
    later in this manual.
    ----------------------------------------

  COMMENT (2.3)  Module Disposition Section

    PRINT
    COMPILE

COMMENT (3) Wrapup and Signoff Commands Section

  STOP
```

This structure is important because it tells something about the various states each module can be in. The reader should study this structure carefully before going on.

2.2 Starting the Session

The ISUS session begins with a signon procedure that includes the user specifying his name and password (these facilities exist only trivially in ISUS), and initializing the system with this command sequence:

```
START
ID = identifier
PASSWORD = password
RUN = run-name  ((optional))
```

For ISUS, the identifier and password given must match those stored in the old master for the session to continue. After the identifier and password have been verified the system is ready to accept other commands. Note that the ID command must follow the START command, and the PASSWORD command must follow the ID command. See Figure A for an example of the signon sequence.

2.3 The Active Module

Once the system is "ready to go" the user can select a module for analysis and modification. This is done with the MODULE command.

The format for the module command provides for the name of the Module that is to be made active, as follows:

```
MODULE = name
```

The system will find the named module and analyze it so that it can be operated upon by the other commands.

```
COMMAND?
START
 PROCESSING: START
 OLD MASTER LOADED. CONTENTS ARE AS FOLLOWS:
     SIZE OF DIRECTORY =     4
      NAME    TY   STRT   LEN
     -------  --   ----   ---
     EX1       M      1    67
     LABTEST   M     68    24
     DOTEST    M     92    15
     DELTEST   M    107    21

COMMAND?
ID = TEST
 PROCESSING: ID = TEST

COMMAND?
PASSWORD = TESTPW
 PROCESSING: PASSWORD = TESTPW

COMMAND?
RUN = ONE
 PROCESSING: RUN = ONE
 RUN NAME IS ONE

COMMAND?
MODULE = DELTEST
 PROCESSING: MODULE = DELTEST
 LOADING MODULE DELTEST
 MODULE LOADED,   21 STATEMENTS READ.
```

Figure A. Sample of ISUS Signon Procedure

The "active" module is the one that is loaded as a result of a MODULE command, and continues throughout a set of modifications, until another MODULE command or a STOP command is encountered.

The "active module" can be seen by the user in one of two ways: (1) by using the PRINT command, and (2) in terms of the standard statement numbering. These two facets of ISUS use are described next.

2.3.1 Statement Numbering

The ISUS system works on statements, not lines. Each FORTRAN statement is numbered beginning from the first statement of the module, based on a perdetermined stepsize. If the stepsize is 100 (for ISUS it is), then statements are numbered as follows:

```
Statement 1:   Number 100
Statement 2:   Number 200
etc.
```

Note that even if a statement resides on two or more lines it is still treated as a unit and has only one statement number.

When statements are added to a module -- to the active module -- they are added between existing statements, using an increment of 10. This imposes a limit on the total number of new statements that can be added between any two existing statements without changing the statement numbering scheme.

The statement numbering takes place when the module is selected and input from the master file (See Sec. 3). If a module is selected and then changed, and then released back to the master file it can be reselected later in a session and appear renumbered. In otherwords, the statement numbers are assigned at the time when a module is taken from the master file and made active, and each time this is done the numbering is generated afresh.

2.3.2 The PRINT Command

The user can see what the current state of the active module is at any time by using the PRINT command.

The print command causes a formatted listing of the active module -- including any and all modifications made up to this time -- to be printed. A sample of this output is shown in Figure B. Some of the points about this figure are the following:

- The statement numbers (Point A in Figure B)
- The statement texts (Point B)
- The delete flag which marks statements that have been deleted (Point C) (deleted statements are only shown if the ALL option is used with the PRINT command)

```
COMMAND?
PRINT ALL
 PROCESSING: PRINT ALL
   100 *DECK DELTEST
   200       SUBROUTINE DELTEST(M,A,H)
   300 CBEGDOC
   400 C     THIS ROUTINE TESTS CONSEQUENCE ANALYSIS OF DEL COMMAND.
   500 CENDDOC
   600X      DIMENSION M(10),H(2,3)
   700X      DO 100 I=H(1,1),H(2,3)
   800       M(I)=A
   900X100   CONTINUE
   910       REAL M(20),H(2,3)
  1000       IF(I.GT.100)GOTO 200
  1100X      A=M(10)
  1200       GOTO 300
  1300 200   H(1,2)=A+1
  1400       H(1,3)=H(1,2)*5
  1500       M(1)=0
  1600 300   J=A+M(1)
  1700       WRITE(100)J
  1800       IF(A.EQ.1)GOTO 400
  1900       M(A)=J
  2000X400   RETURN
  2100       END
```

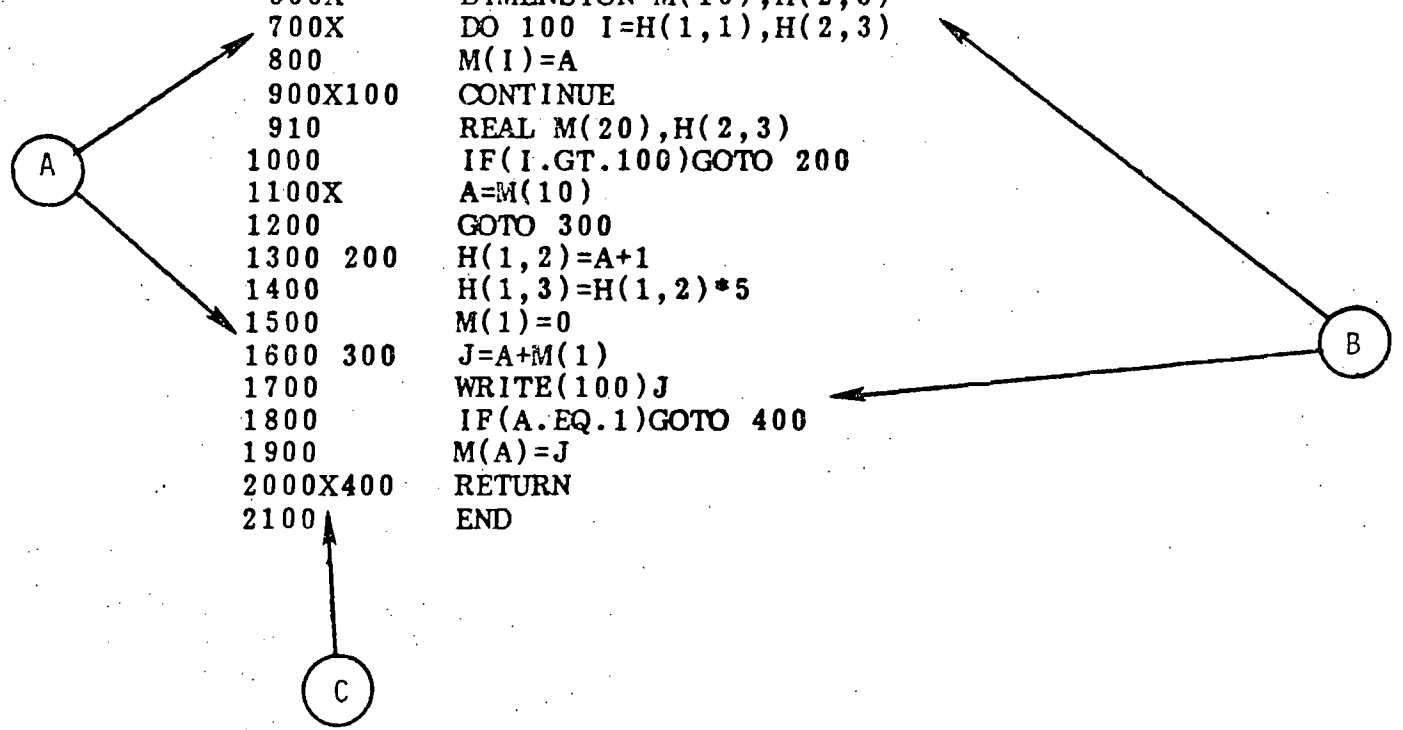


Figure B. Sample of PRINT Command

The reader should study this format to see how a single FORTRAN module is represented by ISUS to the user.

2.4 Response to Commands

The commands that ISUS provides to handle FORTRAN program modifications all have a common feature: they each specify the way a particular modification is to be made.

Note: In ISUS only a subset of all commands is implemented.

Each command applies to the active module, unless otherwise specified. When a command is executed ISUS responds by providing a short message that indicates the effect of the command.

Some commands have consequences as well as direct effects; consequences are described in Sec. 4, below.

When a command is issued and it is accepted by ISUS then that set of indicated actions is performed on all appropriate statements in the currently active module. To see the effect of one command the user can employ this sequence...

```
PRINT
candidate command
PRINT
```

which will provide a "before" and "after" set of images of the modification.

2.5 Example Command Script

A sample script is shown in Figure C; this is actual output from ISUS so the sample illustrates both a sequence of useful commands and the responses ISUS gives to those commands.

```
COMMAND?
DIMENSION ADD D 2 10 SET=(3)
 PROCESSING: DIMENSION ADD D 2 10 SET=(3)
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT     200, NO CHANGE MADE.
 DIMENSION ADDED - ALTERED STATEMENT  2200
 NEW TEXT --- DIMENSION D(1,2,10,3,4)
 DIMENSION ADDED - ALTERED STATEMENT  2300
 NEW TEXT --- D(I,J,3,K,L)=0.
 DIMENSION ADDED - ALTERED STATEMENT  2400
 NEW TEXT --- D(I,J(K),3,L(M(N)),K)=234

COMMAND?
DIMENSION CHANGE E 3 20
 PROCESSING: DIMENSION CHANGE E 3 20
 EXTENT CHANGED - ALTERED STATEMENT  2800
 NEW TEXT --- DIMENSION E(10,5,20,5)

COMMAND?
DIMENSION REORDER F (1,3,4,2)
 PROCESSING: DIMENSION REORDER F (1,3,4,2)
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT     200, NO CHANGE MADE.
 DIMENSIONS REORDERED - ALTERED STATEMENT  3400
 NEW TEXT --- DIMENSION F(10,4,5,3)
 DIMENSIONS REORDERED - ALTERED STATEMENT  3500
 NEW TEXT --- F(1,3,4,2)=F(1,3,4,2)+1
 DIMENSIONS REORDERED - ALTERED STATEMENT  3600
 NEW TEXT --- AA=(F(1,3,4,2)*Z+34)

COMMAND?
VARIABLE CHANGE QUISP TRIX
 PROCESSING: VARIABLE CHANGE QUISP TRIX
 VARIABLE CHANGED - ALTERED STATEMENT 6300
 NEW TEXT --- TRIX(2,3,4)=TRIX(3,4,5)
 VARIABLE CHANGED - ALTERED STATEMENT 6400
 NEW TEXT --- V(TRIX(3,4,2),1,7)=34
```

Figure C. Typical Command Output

Following are some important points about ISUS command input:

- The system prompts the user for commands by printing "COMMAND?" followed by a carriage retrun
- The first "$" encountered on an input command line ends the system's scan of the line
- Command keywords cannot have internal blanks; the blanks is the command delimiter
- COMMENT commands are just that -- comments that are not otherwise processed
- After an UPDATE REP or UPDATE ADD command the system prompts the user for input lines with statement numbers, and reads the input lines in standard FORTRAN format:

  (a) Columns 1-5 for the statement label
  (b) Column 6 for the continuation indicator (blank means no continuation)
  (c) Columns 7-72 for the statement text

- The "$END" that is used to end statement text input must appear in Columns 1-4
- Commands after the "STOP" command are ignored.

3 THE ISUS LIBRARY SYSTEM

This section describes the operation of the ISUS library system, on which the information about a FORTRAN software system is stored during a run.

3.1 ISUS Library Basics

The Semantic Update System (ISUS) maintains control of a set of FORTRAN programs that are stored on a special file called a master. The information in the master file consists (in principle) of the following:

(1) Directory data about all of the COMMONS and all of the Modules in the FORTRAN software system.

(2) The specification of all of the COMMONS -- that is, their actual texts.

(3) The specification of the Modules themselves.

The ISUS user has control of these only through the commands that are recognized by the ISUS system.

There are two activities you will want to do with the support of ISUS that have to do with the library:

(1) Make a new library after a set of changes.

(2) Generate a COMPILE file for a new software system, which may or may not occur after making changes.

In addition, a special setup job is needed to convert from an existing library system format to the ISUS format.

The format for ISUS is open and direct; ISUS relies on the system to perform character packing (most often on strings of blanks). The format for the internal structure of the ISUS master is given in Ref. 11.

3.2 ISUS Commands Affecting Library Operations

These commands affect the library: START, MODULE, COMPILE, STOP. Here is the set of actions taken in each case:

START -- The standard Old Master file is read and analyzed so it is ready for use by ISUS.

MODULE -- The currently active module (see Sec. 2.2) is written to the working master file with all of the changes made to it included in the (now) new master copy. Then the specified module is read by ISUS and made active.

COMPILE -- The currently active module is flagged for output to the compilation file at the end of the ISUS session.

STOP -- The currently active module is written to the working master file, with all changes that have been made to it. All modules that have been flagged are put on the COMPILE file so that they can be recompiled and used in subsequent versions of the system. The working master is written out as the new master file.

These four actions allow the user to modify and update any part of a large and complex software system.

3.3 Special Features of Library System Use

Normally, the operation of the ISUS library system is unseen by the user. The only output the user must be aware of is the listing of library contents printed on the standard output unit when ISUS begins its operation.

4 CONSEQUENCE ANALYSIS

This section describes the way in which ISUS advises the user of consequences of changes. The presentation here is "operational" rather than "technical" and is intended for practical applications. The reader with interest in how these computations are performed should consult (Ref. 13).

4.1 Basic Action of Commands

Once an active module exists, ISUS commands are used to automatically make single or multiple modifications to the statements in that module. When discussing "consequenes" we are not describing the normal set of modifications -- called direct effects -- that the command entails.

For example, the command:

VARIABLE CHANGE old-variable new-variable

will automatically change all instances of "old-variable" for "new-variable" in the current module. This has no other consequences of these kinds:

- It does not render the program syntactically illegal
- It does not produce effects that could cause a problem in either superior or inferior modules (callers or callees)
- It does not interact with any other commands

This command is "safe" in the sense that it goes through without difficulty. Naturally, if the user is unwise and commands a name change that results in a program that is not equivalent to the original one that is unfortunate; however, ISUS makes not attempt to assure that the program remains identical before and after the modification. (Remember, ISUS's purpose is to help modify programs!)

4.2 Automatic Consequence Analysis

Certain ISUS commands can generate consequences that affect the ultimate usefulness of the FORTRAN module being modified. For example, consider the command:

UPDATE DEL statement-number

when applied to an Assignment Statement (something of the form variable = expression) can have severe consequences because the assigned variable name now may have no value. In this case, ISUS responds by informing the user of these kinds of information:

- That there may be data flow consequences to the indicated statement deletion.

- Providing a list of statements that lie in the successor set of the deleted statement (which logically follow the deleted statement).

- Indicating which ones of those may be invalidated because the variable assignment was deleted.

- Providing for searches of subsidiary/subsequent modules to assure that the consequence identification is comprehensive (in ISUS, only single module searching is done).

Additional consequence analyses will be added to ISUS in future versions.

An example of the kind of consequence analysis performed by ISUS is shown in Figure D. In this example, an UPDATE DELete 1100 command was issued. The consequence analysis shows that the deleted statement(s) have logically following statements which may be affected by the deletion. The second part of the listing shows the subset of the potentially affected statements that actually contain a reference to the variable that, by virtue of the original deletion, is no longer valid. These three statements if ever executed without further modification of the program will probably cause a "variable undefined" error (or equivalent).

```
COMMAND?
UPDATE DEL 600
 PROCESSING: UPDATE DEL 600
 STATEMENT   600 DELETED.
 *** DELETED STATEMENT WAS A DECLARATION.
 *** THE FOLLOWING VARIABLES MAY NO LONGER BE CORRECTLY DEFINED.
 ***       M
 ***       H

COMMAND?
UPDATE DEL 1100
 PROCESSING: UPDATE DEL 1100
 STATEMENT  1100 DELETED.
 *** DELETED STATEMENT WAS ASSIGNMENT TO A VARIABLE.
 *** THE FOLLOWING LIST ARE THE STATEMENTS WHICH
 *** LOGICALLY FOLLOW THE DELETED STATEMENT.
 ***        1200   1600   1700   1800   1900   2000   2100
 *** THE FOLLOWING LIST IS OF THE STATEMENTS WHICH
 *** LOGICALLY FOLLOW THE DELETED STATMENT AND CONTAIN A
 *** REFERENCE TO THE VARIABLE ASSIGNED TO IN THE DELETED
 *** STATEMENT.
 ***        1600
 ***        1800
 ***        1900

COMMAND?
UPDATE DEL 2000
 PROCESSING: UPDATE DEL 2000
 STATEMENT  2000 DELETED.
 *** START OR END OF A STRUCTURE HAS BEEN DELETED.
 *** PROGRAM MAY NO LONGER BE SYNTACTICALLY CORRECT.
 STATEMENT  1800:
 MESSAGE: TARGET OF GOTO NOT FOUND.
 *** AS INDICATED IN THE ABOVE MESSAGE(S), THE LOGICAL
 *** STRUCTURE OF THE CURRENT MODULE IS FLAWED, AND SO THE
 *** DIRECTED GRAPH OF THE MODULE IS UNUSABLE. ANY
 *** CONSEQUENCE ANALYSIS THAT REQUIRES THE DIRECTED
 *** GRAPH WILL BE INHIBITED UNTIL THE LOGICAL STRUCTURE
 *** OF THE MODULE IS SOUND.
```

Figure D. Sample of Consequence Analysis Output

5 RESTRICTIONS

Like any system, ISUS has certain restrictions that are necessary to allow the system to "fit" in a particular environment. ISUS, because it operates on source programs, also has a few restrictions that pertain to the kind of statements and the syntactic rules that govern them; these restrictions are intended to simplify some of the modules.

Restrictions that apply to ISUS are the following; when the restriction is an implementation-time "parameter" this fact is noted.

(1) The total number of statements in one module must be less than or equal to 500. This is an implementation parameter governed by the size of a control table.

(2) There may be no more than 9 continuation lines in any FORTRAN statement.

The normal limit in FORTRAN is 19; the value 9 is used as the maximum to limit the size of the statement buffer. It is very, very rare for a statement to have over 9 continuations.

(3) Each FORTRAN keyword must be delimited by a "blank."

This syntactic rule is used to simplify the parser/recognizer that is used inside ISUS. The rule means that the user cannot write...

```
DO10I=1,10
```

but must instead write...

```
DO 10 I=1,10
```

The actual rule is that each syntactic entity must be delimited by a blank or a special character. The special characters that act as delimiters in FORTRAN are: + - . , * $ / ( ) etc.

Most FORTRAN programmers do this as a normal part of making their programs readable. Blank/special-character delimitation is in keeping with modern software engineering principles.

(4) There must be no instantiated blanks in FORTRAN syntactic entities.

This is related to the rule above about delimitation, but goes the other way. You cannot write "D I M E N S I O N" and have ISUS understand it; it must be "DIMENSION".

99% of all programmers follow this convention naturally.

(5) The number of syntactic units is limited to 200 per statement. A syntactic unit or lexical item is a keyword, an item of punctuation, an operator, a variable or constant, or some other legal part of a statement.

This value is chosen as an engineering compromise for practical programs; it can be changed if necessary by increasing the length of an internal table.

(6) The maximum number of characters per statement is 666. This is the same as ten lines, column 7 through 72, or 66 characters per line. Note that the label on the statement occurs on the FIRST line only; other label information would be an error.

(7) The total number of COMMONs plus Modules in a program master file given to ISUS to process must be less than or equal to 100. This means there can be one module with 99 COMMONs, or one COMMON used by 99 modules, or any such combination. The limitation arises from the ISUS Program Master subsystem, where the size is determined by a table.

(8) Each command line must meet the following criteria:

a. There must be less than or equal to 80 characters

b. There must be fewer than 30 "tokens" (non-blank substrings) in each command

Commands cannot be continued onto subsequent lines.

(9) To analyze the logic of a module ISUS requires that each FORTRAN DO statement have its own CONTINUE statement as its target. Whenever ISUS scans the logical structure of a module and finds a DO without a CONTINUE, ISUS will insert a new CONTINUE statement with a new label for the DO, and put the new label in the DO statement. ISUS does this automatically as part of consequence analysis.

6 COMMAND DOCUMENTATION

This section describes the way each ISUS command is documented, in the ISUS Command Reference Manual (Ref. 14).

Each command's description occupies one page in the manual, and the same format is used throughout. The main topics and the kind of information to be found are as follows:

NAME...

The Name of the Command

SYNTAX...

Brief syntax description with required keywords in UPPER CASE, and meta-variables in lower case.

DESCRIPTION...

Short description of what the command does, and other information that helps the user tell how to use the command properly.

EXAMPLE...

Example of the use of the command and the effect it would have in an actual situation.

CONSEQUENCES...

Here there will be short descriptions of the consequences of the command and what they might mean.

Note that the "consequences" mentioned here are ONLY those which result from the direct and indirect actions of ISUS on the users' behalf. This means that when a command is supposed to do something that action is NOT described here in the CONSEQUENCES section. What is described are side-effects and implications that using the command could have.

This is described in more detail in Sec. 4.

NOTES...

Special conditions or exceptions that apply to this command.

Sometimes, problems with the command are described here.

SEE ALSO...

A short list of other commands that are related to this command.

---

7 COMPLETE EXAMPLE

This section describes a complete example ISUS session. The set of commands to be executed is the following:

```
START
ID = TEST
PASSWORD = TESTPW
RUN = ONE
MODULE = EX1
PRINT
COMMENT  *1*  THE DIMENSION COMMANDS
DIMENSION DEL A 3
DIMENSION ADD B 2 20
DIMENSION ADD D 2 10 SET=(3)
DIMENSION CHANGE E 3 20
DIMENSION REORDER F (1,3,2,4)
DIMENSION EXCHANGE B 1 2
PRINT
COMMENT  *2*  THE CONSTANT COMMANDS
CONSTANT CHANGE 7 55
CONSTANT DEL 9
PRINT
COMMENT  *3*  THE VARIABLE COMMANDS
VARIABLE DEL H
VARIABLE CHANGE QUISP TRIX
PRINT
MODULE = LABTEST
PRINT
COMMENT  *4*  THE LABEL COMMANDS
LABEL ADD 2000 300
LABEL CHANGE 200 400
LABEL DEL 1900
PRINT
RUN = TWO
MODULE = DOTEST
PRINT
COMMENT  *5*  THE DO COMMANDS
DO ADD 1200 1300 I=49,1,-1
DO CHANGE 1150 INDEX = KOUNT
DO CHANGE 1150 INDEX = INDEX ONLY
DO ADD 1300 1300 J=A(INDEX)+1,A(200)
PRINT
MODULE = DELTEST
PRINT
COMMENT  *6*  THE UPDATE COMMANDS
UPDATE ADD 400
C      ALSO THE ADD AND REP COMMANDS.
$END
UPDATE DEL 600
UPDATE DEL 700
UPDATE REP 900
       REAL M(20),H(2,3)
$END
```

```
UPDATE DEL 1100
UPDATE REP 2000
400    IF(H(1,2).LT.100) GOTO 200
       RETURN
$END
PRINT
COMMENT  *7*  THE SEARCH COMMANDS
POS 100
FIND /H(1/
SUB /H(1/H(2/ A 1300 1400
COMMENT  *8*  THE COMPILE COMMAND
COMPILE
COMPILE DOTEST
STOP
```

These commands are assumed to be applied to an old master file which contains the appropriately named modules, in this case: EX1, LABTEST, DOTEST, and DELTEST.

The output which ISUS produces from this set of commands is provided at the end of this section, as Figure E (there are 26 pages to Figure E). For convenience in the description, we will refer to the parts of Figure E by page number and within each page, by item number (when necessary).

Page 1

The system is signed on with the START command. ISUS responds by indicating that the old master is loaded, stating the size of the directory (here, four modules), and then listing them. The numbers that are shown are the number of statements and the addresses used by ISUS in storing the modules in a random access Working Master file. The listing shows the type (TY) of each entry, either "M" for Module (main program, subroutine, function, or block data) or "G" for Global (group of COMMON declarations).

Next, the user identifies his ID and PASSWORD, and specifies the RUN name, here "ONE". Last, the user indicates he wishes to work with the module named "EX1" and the system responds by making it the active module. The PRINT command that follows produces the output on Page 2 and Page 3 of the sample.

Page 2, 3

The results of the PRINT command mentioned above.

Page 4

These commands illustrate the power of ISUS in modifying DIMENSION statements. The first command specifies to modify the

```
  SUSIRUN OLD.1 NEW.1 )
EXECUTING INTERACTIVE SEMANTIC UPDATE SYSTEM SUS FOR FORTRAN
TODAY IS 31-JAN-80 THE TIME IS 08:41:04.

SYSTEM NAME: SUS2

OLD.1 IS THE OLD MASTER TO BE PROCESSED
NEW.1 IS THE MODIFIED NEW MASTER TO BE PRODUCED
--------

COMMAND?
START
 PROCESSING: START
 OLD MASTER LOADED. CONTENTS ARE AS FOLLOWS:
      SIZE OF DIRECTORY =     4
       NAME    TY   STRT   LEN
      -------  --   ----   ---
      EX1       M      1    67
      LABTEST   M     68    24
      DOTEST    M     92    15
      DELTEST   M    107    21

COMMAND?
ID = TEST
 PROCESSING: ID = TEST

COMMAND?
PASSWORD = TESTPW
 PROCESSING: PASSWORD = TESTPW

COMMAND?
RUN = ONE
 PROCESSING: RUN = ONE
 RUN NAME IS ONE

COMMAND?
MODULE = EX1
 PROCESSING: MODULE = EX1
 LOADING MODULE EX1     .
 MODULE LOADED,  67 STATEMENTS READ.
```

Figure E, Page 1 -- Complete ISUS Output Example

```
COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK EX1
   200       SUBROUTINE EX1(A,B,C,D,E,F,N)
   300 C     E X A M P L E  1
   400 C     ----------------
   500 C
   600 C     THIS EXAMPLE IS USED TO ILLUSTRATE THE OPERATION OF SOME OF
   700 C     THE COMMANDS FOR THE SEMANTIC UPDATE SYSTEM (SUS).
   800 C
   900 C
  1000       DIMENSION A(10,10,10,10)
  1100       A(1,2,4,5)=N(2,3,4,5)
  1200       A(1,2,I(J(K)),L(M))=1
  1300 C
  1400 C     DIMENSION ADD COMMAND
  1500 C
  1600       DIMENSION B(1,2,3,4)
  1700       B(1,2,3,4)=0
  1800       B(1,2,3,4)=C(1,23)
  1900 C
  2000 C     DIMENSION ADD WITH SET OPTION
  2100 C
  2200       DIMENSION D(1,2,3,4)
  2300       D(I,J,K,L)=0.
  2400       D(I,J(K),L(M(N)),K)=234
  2500 C
  2600 C     DIMENSION CHANGE COMMAND
  2700 C
  2800       DIMENSION E(10,5,10,5)
  2900       E(1,1,1,1)=0.0
  3000       Z=E(I,J,K,L)
  3100 C
  3200 C     DIMENSION REORDER COMMAND
  3300 C
  3400       DIMENSION F(10,3,4,5)
  3500       F(1,2,3,4)=F(1,2,3,4)+1
  3600       AA=(F(1,2,3,4)*Z+34)
```

Figure E, Page 2 -- Complete ISUS Output Example (Continued)

```
3700 C
3800 C      DIMENSION EXCHANGE COMMAND
3900 C
4000        DIMENSION G(10,2,4,6)
4100        G(1,2,3,4)=G(1,2,3,7)+32
4200        G(1,I(J(K)),3,5)=27
4300 C
4400 C      CONSTANT CHANGE COMMAND
4500 C
4600        W=7
4700        Z=7+C(72)
4800        X(11)=Y(7+4)
4900 C
5000 C      CONSTANT DELETE COMMAND
5100 C
5200        R=2
5300        S(2,9,4)=1.0
5400        T(X(2),7)=.1E03
5500 C
5600 C      VARIABLE DELETE COMMAND
5700 C
5800        H=1
5900        Q(J(H(1)))=1
6000 C
6100 C      VARIABLE CHANGE COMMAND
6200 C
6300        QUISP(2,3,4)=QUISP(3,4,5)
6400        V(QUISP(3,4,2),1,7)=34
6500 C
6600        STOP
6700        END
```

Figure E, Page 3 -- Complete ISUS Output Example (Continued)

```
COMMAND?
COMMENT  *1*  THE DIMENSION COMMANDS
 PROCESSING: COMMENT  *1*  THE DIMENSION COMMANDS

COMMAND?
DIMENSION DEL A 3
 PROCESSING: DIMENSION DEL A 3
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT   200, NO CHANGE MADE.
 DIMENSION DELETED - ALTERED STATEMENT  1000
 NEW TEXT --- DIMENSION A(10,10,10)
 DIMENSION DELETED - ALTERED STATEMENT  1100
 NEW TEXT --- A(1,2,5)=N(2,3,4,5)
 DIMENSION DELETED - ALTERED STATEMENT  1200
 NEW TEXT --- A(1,2,L(M))=1

COMMAND?
DIMENSION ADD B 2 20
 PROCESSING: DIMENSION ADD B 2 20
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT    200, NO CHANGE MADE.
 DIMENSION ADDED - ALTERED STATEMENT  1600
 NEW TEXT --- DIMENSION B(1,2,20,3,4)
 DIMENSION ADDED - ALTERED STATEMENT  1700
 NEW TEXT --- B(1,2,1,3,4)=0
 DIMENSION ADDED - ALTERED STATEMENT  1800
 NEW TEXT --- B(1,2,1,3,4)=C(1,23)

COMMAND?
DIMENSION ADD D 2 10 SET=(3)
 PROCESSING: DIMENSION ADD D 2 10 SET=(3)
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT    200, NO CHANGE MADE.
 DIMENSION ADDED - ALTERED STATEMENT  2200
 NEW TEXT --- DIMENSION D(1,2,10,3,4)
 DIMENSION ADDED - ALTERED STATEMENT  2300
 NEW TEXT --- D(I,J,3,K,L)=0.
 DIMENSION ADDED - ALTERED STATEMENT  2400
 NEW TEXT --- D(I,J(K),3,L(M(N)),K)=234
```

Figure E, Page 4 -- Complete ISUS Output Example (Continued)

```
COMMAND?
DIMENSION CHANGE E 3 20
 PROCESSING: DIMENSION CHANGE E 3 20
 EXTENT CHANGED - ALTERED STATEMENT  2800
 NEW TEXT --- DIMENSION E(10,5,20,5)


COMMAND?
DIMENSION REORDER F (1,3,2,4)
 PROCESSING: DIMENSION REORDER F (1,3,2,4)
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT    200, NO CHANGE MADE.
 DIMENSIONS REORDERED - ALTERED STATEMENT  3400
 NEW TEXT --- DIMENSION F(10,4,3,5)
 DIMENSIONS REORDERED - ALTERED STATEMENT  3500
 NEW TEXT --- F(1,3,2,4)=F(1,3,2,4)+1
 DIMENSIONS REORDERED - ALTERED STATEMENT  3600
 NEW TEXT --- AA=(F(1,3,2,4)*Z+34)
```

dimensioned variable A by deleting the third dimension: DIMENSION DEL A 3.

The response by the system is to find instances of A and to modify them as requested, and to report to the user the kind of modifications done. Each modification is reported to the user.

The other commands shown on Page 4 are variations of the several kinds of DIMENSION modification commands. Finally, the user asks for a PRINT to see what the current state of the module is. This is done on Pages 5 and 6.

Page 5,6

The result of the PRINT command just mentioned. The reader should study the results of the set of modifications to see how ISUS interprets each command.

Page 7

The commands now switch to modifications of constants in the program, which is still the same one as before (EX1). Note that when the command modifies the program sufficiently to affect the syntactic legality of the modified program, then ISUS exlains this fact to the user. This is done with the second of the two commands shown on page 7.

After these modifications, the user then uses the PRINT command to tell where the changes have occurred. This takes up the rest of page 7 and page 8.

Page 8

This is the remainder of the PRINT command after the constant change commands were run.

Page 9

Now the user tries some VARIABLE commands, which are used to modify variable names. As before, when severe changes that may affect the syntactic correctness of the program are made the system notifies the user.

After these commands are executed, the user issues another PRINT command, and the post-modification state of the example program EX1 is shown in pages 9 and 10.

Page 10

This is the result of the PRINT command, and shows the final state of the module EX1.

```
COMMAND?
DIMENSION EXCHANGE B 1 2
 PROCESSING: DIMENSION EXCHANGE B 1 2
 UNSUBSCRIPTED OCCURENCE OF VARIABLE IN STATEMENT    200, NO CHANGE MADE.
 DIMENSIONS EXCHANGED - ALTERED STATEMENT  1600
 NEW TEXT --- DIMENSION B(2,1,20,3,4)
 DIMENSIONS EXCHANGED - ALTERED STATEMENT  1700
 NEW TEXT --- B(2,1,1,3,4)=0
 DIMENSIONS EXCHANGED - ALTERED STATEMENT  1800
 NEW TEXT --- B(2,1,1,3,4)=C(1,23)

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK EX1
   200       SUBROUTINE EX1(A,B,C,D,E,F,N)
   300 C     E X A M P L E  1
   400 C     ----------------
   500 C
   600 C     THIS EXAMPLE IS USED TO ILLUSTRATE THE OPERATION OF SOME OF
   700 C     THE COMMANDS FOR THE SEMANTIC UPDATE SYSTEM (SUS).
   800 C
   900 C
  1000       DIMENSION A(10,10,10)
  1100       A(1,2,5)=N(2,3,4,5)
  1200       A(1,2,L(M))=1
  1300 C
  1400 C     DIMENSION ADD COMMAND
  1500 C
  1600       DIMENSION B(2,1,20,3,4)
  1700       B(2,1,1,3,4)=0
  1800       B(2,1,1,3,4)=C(1,23)
  1900 C
  2000 C     DIMENSION ADD WITH SET OPTION
  2100 C
  2200       DIMENSION D(1,2,10,3,4)
  2300       D(I,J,3,K,L)=0.
  2400       D(I,J(K),3,L(M(N)),K)=234
  2500 C
  2600 C     DIMENSION CHANGE COMMAND
  2700 C
  2800       DIMENSION E(10,5,20,5)
  2900       E(1,1,1,1)=0.0
  3000       Z=E(I,J,K,L)
  3100 C
  3200 C     DIMENSION REORDER COMMAND
  3300 C
  3400       DIMENSION F(10,4,3,5)
  3500       F(1,3,2,4)=F(1,3,2,4)+1
  3600       AA=(F(1,3,2,4)*Z+34)
```

Figure E, Page 5 -- Complete ISUS Output Example (Continued)

```
3700 C
3800 C      DIMENSION EXCHANGE COMMAND
3900 C
4000        DIMENSION G(10,2,4,6)
4100        G(1,2,3,4)=G(1,2,3,7)+32
4200        G(1,I(J(K)),3,5)=27
4300 C
4400 C      CONSTANT CHANGE COMMAND
4500 C
4600        W=7
4700        Z=7+C(72)
4800        X(11)=Y(7+4)
4900 C
5000 C      CONSTANT DELETE COMMAND
5100 C
5200        R=2
5300        S(2,9,4)=1.0
5400        T(X(2),7)=.1E03
5500 C
5600 C      VARIABLE DELETE COMMAND
5700 C
5800        H=1
5900        Q(J(H(1)))=1
6000 C
6100 C      VARIABLE CHANGE COMMAND
6200 C
6300        QUISP(2,3,4)=QUISP(3,4,5)
6400        V(QUISP(3,4,2),1,7)=34
6500 C
6600        STOP
6700        END
```

Figure E, Page 6 -- Complete ISUS Output Example (Continued)

```
COMMAND?
COMMENT  *2*  THE CONSTANT COMMANDS
 PROCESSING: COMMENT  *2*  THE CONSTANT COMMANDS

COMMAND?
CONSTANT CHANGE 7 55
 PROCESSING: CONSTANT CHANGE 7 55
 CONSTANT CHANGED - ALTERED STATEMENT 4100
 NEW TEXT --- G(1,2,3,4)=G(1,2,3,55)+32
 CONSTANT CHANGED - ALTERED STATEMENT 4600
 NEW TEXT --- W=55
 CONSTANT CHANGED - ALTERED STATEMENT 4700
 NEW TEXT --- Z=55+C(72)
 CONSTANT CHANGED - ALTERED STATEMENT 4800
 NEW TEXT --- X(11)=Y(55+4)
 CONSTANT CHANGED - ALTERED STATEMENT 5400
 NEW TEXT --- T(X(2),55)=.1E03
 CONSTANT CHANGED - ALTERED STATEMENT 6400
 NEW TEXT --- V(QUISP(3,4,2),1,55)=34

COMMAND?
CONSTANT DEL 9
 PROCESSING: CONSTANT DEL 9
 CONSTANT DELETED - ALTERED STATEMENT 5300
 NEW TEXT --- S(2,,4)=1.0
 *** PROGRAM IS PROBABLY NO LONGER SYNTACTICALLY CORRECT.

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK EX1
   200       SUBROUTINE EX1(A,B,C,D,E,F,N)
   300 C     E X A M P L E  1
   400 C     ---------------
   500 C
   600 C     THIS EXAMPLE IS USED TO ILLUSTRATE THE OPERATION OF SOME OF
   700 C     THE COMMANDS FOR THE SEMANTIC UPDATE SYSTEM (SUS).
   800 C
   900 C
  1000       DIMENSION A(10,10,10)
  1100       A(1,2,5)=N(2,3,4,5)
  1200       A(1,2,L(M))=1
  1300 C
  1400 C     DIMENSION ADD COMMAND
  1500 C
  1600       DIMENSION B(2,1,20,3,4)
  1700       B(2,1,1,3,4)=0
  1800       B(2,1,1,3,4)=C(1,23)
  1900 C
  2000 C     DIMENSION ADD WITH SET OPTION
  2100 C
  2200       DIMENSION D(1,2,10,3,4)
  2300       D(I,J,3,K,L)=0.
  2400       D(I,J(K),3,L(M(N)),K)=234
```

Figure E, Page 7 -- Complete ISUS Output Example (Continued)

```
2500 C
2600 C      DIMENSION CHANGE COMMAND
2700 C
2800        DIMENSION E(10,5,20,5)
2900        E(1,1,1,1)=0.0
3000        Z=E(I,J,K,L)
3100 C
3200 C      DIMENSION REORDER COMMAND
3300 C
3400        DIMENSION F(10,4,3,5)
3500        F(1,3,2,4)=F(1,3,2,4)+1
3600        AA=(F(1,3,2,4)*Z+34)
3700 C
3800 C      DIMENSION EXCHANGE COMMAND
3900 C
4000        DIMENSION G(10,2,4,6)
4100        G(1,2,3,4)=G(1,2,3,55)+32
4200        G(1,I(J(K)),3,5)=27
4300 C
4400 C      CONSTANT CHANGE COMMAND
4500 C
4600        W=55
4700        Z=55+C(72)
4800        X(11)=Y(55+4)
4900 C
5000 C      CONSTANT DELETE COMMAND
5100 C
5200        R=2
5300        S(2,,4)=1.0
5400        T(X(2),55)=.1E03
5500 C
5600 C      VARIABLE DELETE COMMAND
5700 C
5800        H=1
5900        Q(J(H(1)))=1
6000 C
6100 C      VARIABLE CHANGE COMMAND
6200 C
6300        QUISP(2,3,4)=QUISP(3,4,5)
6400        V(QUISP(3,4,2),1,55)=34
6500 C
6600        STOP
6700        END
```

Figure E, Page 8 -- Complete ISUS Output Example (Continued)

```
COMMAND?
COMMENT  *3*  THE VARIABLE COMMANDS
 PROCESSING: COMMENT  *3*  THE VARIABLE COMMANDS

COMMAND?
VARIABLE DEL H
 PROCESSING: VARIABLE DEL H
 VARIABLE DELETED - ALTERED STATEMENT 5800
 NEW TEXT --- =1
 VARIABLE DELETED - ALTERED STATEMENT 5900
 NEW TEXT --- Q(J())=1
 *** PROGRAM IS PROBABLY NO LONGER SYNTACTICALLY CORRECT.

COMMAND?
VARIABLE CHANGE QUISP TRIX
 PROCESSING: VARIABLE CHANGE QUISP TRIX
 VARIABLE CHANGED - ALTERED STATEMENT 6300
 NEW TEXT --- TRIX(2,3,4)=TRIX(3,4,5)
 VARIABLE CHANGED - ALTERED STATEMENT 6400
 NEW TEXT --- V(TRIX(3,4,2),1,55)=34

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK EX1
   200       SUBROUTINE EX1(A,B,C,D,E,F,N)
   300 C     E X A M P L E  1
   400 C     ---------------
   500 C
   600 C     THIS EXAMPLE IS USED TO ILLUSTRATE THE OPERATION OF SOME OF
   700 C     THE COMMANDS FOR THE SEMANTIC UPDATE SYSTEM (SUS).
   800 C
   900 C
  1000       DIMENSION A(10,10,10)
  1100       A(1,2,5)=N(2,3,4,5)
  1200       A(1,2,L(M))=1
  1300 C
  1400 C     DIMENSION ADD COMMAND
  1500 C
  1600       DIMENSION B(2,1,20,3,4)
  1700       B(2,1,1,3,4)=0
  1800       B(2,1,1,3,4)=C(1,23)
  1900 C
  2000 C     DIMENSION ADD WITH SET OPTION
  2100 C
  2200       DIMENSION D(1,2,10,3,4)
  2300       D(I,J,3,K,L)=0.
  2400       D(I,J(K),3,L(M(N)),K)=234
```

Figure E, Page 9 -- Complete ISUS Output Example (Continued)

```
2500 C
2600 C      DIMENSION CHANGE COMMAND
2700 C
2800        DIMENSION E(10,5,20,5)
2900        E(1,1,1,1)=0.0
3000        Z=E(I,J,K,L)
3100 C
3200 C      DIMENSION REORDER COMMAND
3300 C
3400        DIMENSION F(10,4,3,5)
3500        F(1,3,2,4)=F(1,3,2,4)+1
3600        AA=(F(1,3,2,4)*Z+34)
3700 C
3800 C      DIMENSION EXCHANGE COMMAND
3900 C
4000        DIMENSION G(10,2,4,6)
4100        G(1,2,3,4)=G(1,2,3,55)+32
4200        G(1,I(J(K)),3,5)=27
4300 C
4400 C      CONSTANT CHANGE COMMAND
4500 C
4600        W=55
4700        Z=55+C(72)
4800        X(11)=Y(55+4)
4900 C
5000 C      CONSTANT DELETE COMMAND
5100 C
5200        R=2
5300        S(2,,4)=1.0
5400        T(X(2),55)=.1E03
5500 C
5600 C      VARIABLE DELETE COMMAND
5700 C
5800        =1
5900        Q(J())=1
6000 C
6100 C      VARIABLE CHANGE COMMAND
6200 C
6300        TRIX(2,3,4)=TRIX(3,4,5)
6400        V(TRIX(3,4,2),1,55)=34
6500 C
6600        STOP
6700        END
```

Figure E, Page 10 -- Complete ISUS Output Example (Continued)

Pages 11, 12, 13

Now the user specifies that he wishes to analyze a different module, in this case, LABTEST. ISUS responds by first making sure that all of the modifications to EX1 have been saved appropriately, and then by loading the content of LABTEST.

In the analysis of LABTEST, one of the DO statements is found not to be targeted to a unique CONTINUE statement, so ISUS automatically adds the statement numbered 1975 and makes it the target of the DO located at statement 1100 (see Page 14). The new statement label, "30000" is selected so that it does not match any existing statement label.

Now the user begins modifying LABTEST's labels, using the label commands of ISUS.

The first command adds a statement label 300 at statement location 2000 (see below, after the PRINT command). The second command modifies the 200 label to become 400; note that all instances where such a label is found are indicated by the ISUS system.

The third command deletes the label on statement location 1900; this is actually the label which was originally 200, but which was changed -- by the prior command -- to the label 400. Now ISUS shows the impact of this modification clearly and, after all the statement changes are reported, indicates that the impact is severe enough to warrant issuing the message that the program is no longer syntactically correct. Finally, the logical structure of the module is analyzed, and ISUS prints messages indicating specific problems encountered.

Page 14

The user now issues a PRINT command to show the contents of LABTEST after modification.

At this point, the user declares the end of run "ONE" and the beginning of run "TWO", and also changes his attention to the module DOTEST. Note that the current module (LABTEST) is saved and the new module is made into the active one.

```
COMMAND?
MODULE = LABTEST
 PROCESSING: MODULE = LABTEST
 SAVING MODULE EX1
 MODULE SAVED,  67 STATEMENTS WRITTEN.
 LOADING MODULE LABTEST.
 MODULE LOADED,  24 STATEMENTS READ.
 DO STATMENT  1100 DOES NOT HAVE CONTINUE AS TARGET.
 ADDED NEW CONTINUE STATEMENT  1975 FOR DO STATEMENT.

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK LABTEST
   200       SUBROUTINE LABTEST
   300 CBEGDOC
   400 C     THIS PROGRAM TESTS THE LABEL COMMANDS.
   500 CENDDOC
   600 C
   700       DIMENSION LINE(200)
   800 C
   900       READ(200,200,ERR=200)(LINE(I),I=1,200)
  1000       WRITE(200,200,ERR=200)(LINE(I),I=1,200)
  1100       DO 30000 J=1,200
  1200       IF(LINE(J).EQ.200)GOTO 200
  1300       GOTO 200
  1400       GOTO(200,200,200,200),LINE(J)-200
  1500       GOTO J,(200,200,200,200,200)
  1600       IF(LINE(J)-200)200,200,200
  1700       RETURN 200
  1800 500   ASSIGN 200 TO J
  1900 200   LINE(200)=200
  1975 30000 CONTINUE
  2000       LINE(200-LINE(200))=J*200
  2100       IF(J.GT.200)LINE(200)=0
  2200 100   FORMAT(200A1)
  2300       RETURN
  2400       END

COMMAND?
LABEL ADD 2000 300
 PROCESSING: LABEL ADD 2000 300
 LABEL ADDED TO STATEMENT --- NEW TEXT:
 300   LINE(200-LINE(200))=J*200
```

Figure E, Page 11 -- Complete ISUS Output Example (Continued)

```
COMMAND?
LABEL CHANGE 200 400
 PROCESSING: LABEL CHANGE 200 400
 STATEMENT   900 ALTERED --- NEW TEXT:
       READ(200,400,ERR=400)(LINE(I),I=1,200)
 STATEMENT  1000 ALTERED --- NEW TEXT:
       WRITE(200,400,ERR=400)(LINE(I),I=1,200)
 STATEMENT  1200 ALTERED --- NEW TEXT:
       IF(LINE(J).EQ.200)GOTO 400
 STATEMENT  1300 ALTERED --- NEW TEXT:
       GOTO 400
 STATEMENT  1400 ALTERED --- NEW TEXT:
       GOTO(400,400,400,400),LINE(J)-200
 STATEMENT  1500 ALTERED --- NEW TEXT:
       GOTO J,(400,400,400,400,400)
 STATEMENT  1600 ALTERED --- NEW TEXT:
       IF(LINE(J)-200)400,400,400
 STATEMENT  1700 ALTERED --- NEW TEXT:
       RETURN 400
 STATEMENT  1800 ALTERED --- NEW TEXT:
 500   ASSIGN 400 TO J
 STATEMENT  1900 ALTERED --- NEW TEXT:
 400   LINE(200)=200
```

Figure E, Page 12 -- Complete ISUS Output Example (Continued)

```
COMMAND?
LABEL DEL 1900
 PROCESSING: LABEL DEL 1900
 STATEMENT   900 ALTERED --- NEW TEXT:
       READ(200,,ERR=)(LINE(I),I=1,200)
 STATEMENT  1000 ALTERED --- NEW TEXT:
       WRITE(200,,ERR=)(LINE(I),I=1,200)
 STATEMENT  1200 ALTERED --- NEW TEXT:
       IF(LINE(J).EQ.200)GOTO
 STATEMENT  1300 ALTERED --- NEW TEXT:
       GOTO
 STATEMENT  1400 ALTERED --- NEW TEXT:
       GOTO(,,,),LINE(J)-200
 STATEMENT  1500 ALTERED --- NEW TEXT:
       GOTO J,(,,,,)
 STATEMENT  1600 ALTERED --- NEW TEXT:
       IF(LINE(J)-200),,
 STATEMENT  1700 ALTERED --- NEW TEXT:
       RETURN
 STATEMENT  1800 ALTERED --- NEW TEXT:
 500   ASSIGN TO J
 STATEMENT  1900 ALTERED --- NEW TEXT:
       LINE(200)=200
 *** PROGRAM IS PROBABLY NO LONGER SYNTACTICALLY CORRECT.
 STATMENT  1300:
 MESSAGE: GOTO WITHOUT LABEL.
 STATEMENT  1300:
 MESSAGE: COULD NOT FIND NEXT EXECUTABLE STATEMENT.
 STATEMENT  1200:
```

```
MESSAGE: GOTO WITHOUT LABEL.
STATEMENT  1400:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1400:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1400:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1400:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1500:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1500:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1500:
MESSAGE: LABEL NOT FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1500:
MESSAGE: NO LABEL FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1500:
MESSAGE: NO LABEL FOUND IN COMPUTED OR ASSIGNED GOTO.
STATEMENT  1600:
MESSAGE: TARGET OF ARITHMETIC IF LABEL NOT FOUND.
STATEMENT  1600:
MESSAGE: LABEL NOT FOUND IN ARITHMETIC IF.
STATEMENT  1600:
MESSAGE: LABEL NOT FOUND IN ARITHMETIC IF.
*** AS INDICATED IN THE ABOVE MESSAGE(S), THE LOGICAL
*** STRUCTURE OF THE CURRENT MODULE IS FLAWED, AND SO THE
*** DIRECTED GRAPH OF THE MODULE IS UNUSABLE. ANY
*** CONSEQUENCE ANALYSIS THAT REQUIRES THE DIRECTED
*** GRAPH WILL BE INHIBITED UNTIL THE LOGICAL STRUCTURE
*** OF THE MODULE IS SOUND.
```

Figure E, Page 13 -- Complete ISUS Output Example (Continued)

```
COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK LABTEST
   200       SUBROUTINE LABTEST
   300 CBEGDOC
   400 C     THIS PROGRAM TESTS THE LABEL COMMANDS.
   500 CENDDOC
   600 C
   700       DIMENSION LINE(200)
   800 C
   900       READ(200,,ERR=)(LINE(I),I=1,200)
  1000       WRITE(200,,ERR=)(LINE(I),I=1,200)
  1100       DO 30000 J=1,200
  1200       IF(LINE(J).EQ.200)GOTO
  1300       GOTO
  1400       GOTO(,,,),LINE(J)-200
  1500       GOTO J,(,,,,)
  1600       IF(LINE(J)-200),,
  1700       RETURN
  1800 500   ASSIGN TO J
  1900       LINE(200)=200
  1975 30000 CONTINUE
  2000 300   LINE(200-LINE(200))=J*200
  2100       IF(J.GT.200)LINE(200)=0
  2200 100   FORMAT(200A1)
  2300       RETURN
  2400       END

COMMAND?
RUN = TWO
 PROCESSING: RUN = TWO
 RUN NAME IS TWO

COMMAND?
MODULE = DOTEST
 PROCESSING: MODULE = DOTEST
 SAVING MODULE LABTEST
 MODULE SAVED,  25 STATEMENTS WRITTEN.
 LOADING MODULE DOTEST .
 MODULE LOADED,  15 STATEMENTS READ.
```

Figure E, Page 14 -- Complete ISUS Output Example (Continued)

Page 15, 16

First the user does a PRINT command to see the contents of the module, and then performs some DO commands. These add DO statements to the module and then modify the variable names used as indexes to the DO loops.

Note that when additional statements must be created ISUS indicates its actions clearly to the user.

Finally, the user asks for a PRINT to see what the program looks like after modification.

Pages 17 - 19

Now the user selects another module for modifications, here the one called DELTEST. Note that ISUS indicates what it has done with the presently active module (i.e. DOTEST) as it prepares DELTEST for processing.

Next the user runs a PRINT command to see the status of the DELTEST module.

The user then starts a passage in the run which illustrates the use of the classic update commands. The first command simply adds a statement, which is assigned the location 410 (see the output of the PRINT command on page 19).

The next command eliminates a declaration, and this means that certain variables may no longer be defined; ISUS lists these for the user's consideration.

The next command deletes statement 700, which is actually a DO statement. Deleting a DO without deleting its target may cause a structure error and this fact is presented to the user.

The following command shows what happens when a labeled statement is replaced: ISUS advises the user that the program may no longer even compile. Note that subsequent commands that restore the program to compilable state are not assumed to occur.

Next there is an example of the consequence analysis that ISUS can handle when an assignment statement is modified and/or deleted. The output shows both: (a) the set of successor statements (in the logical sense) that are affected, and (b) the set of successor statements that involve the variable name that was deleted. This is an important list because it suggests the impact of the original deletion.

```
COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK DOTEST
   200       SUBROUTINE DOTEST
   300 CBEGDOC
   400 C     THIS ROUTINE HELPS TEST DO COMMANDS.
   500 C
   600 CENDDOC
   700 C
   800       DIMENSION A(50)
   900 C
  1000 30000 IF(A(1).EQ.0)GOTO 29999
  1100       I=1
  1200       A(I)=A(I+1)
  1300       J=J+A(I)
  1400 29999 RETURN
  1500       END

COMMAND?
COMMENT  *5* THE DO COMMANDS
 PROCESSING: COMMENT  *5* THE DO COMMANDS

COMMAND?
DO ADD 1200 1300 I=49,1,-1
 PROCESSING: DO ADD 1200 1300 I=49,1,-1
 CREATED STATEMENT  1150:
       DO 29998 I=49,1,-1
 CREATED STATEMENT  1350:
 29998 CONTINUE

COMMAND?
DO CHANGE 1150 INDEX = KOUNT
 PROCESSING: DO CHANGE 1150 INDEX = KOUNT
 ALTERED STATEMENT  1100--- NEW TEXT:
       KOUNT=1
 ALTERED STATEMENT  1150--- NEW TEXT:
       DO 29998 KOUNT=49,1,-1
 ALTERED STATEMENT  1200--- NEW TEXT:
       A(KOUNT)=A(KOUNT+1)
 ALTERED STATEMENT  1300--- NEW TEXT:
       J=J+A(KOUNT)
```

Figure E, Page 15 -- Complete ISUS Output Example (Continued)

```
COMMAND?
DO CHANGE 1150 INDEX = INDEX ONLY
 PROCESSING: DO CHANGE 1150 INDEX = INDEX ONLY
 ALTERED STATEMENT  1150--- NEW TEXT:
       DO 29998 INDEX=49,1,-1
 ALTERED STATEMENT  1200--- NEW TEXT:
       A(INDEX)=A(INDEX+1)
 ALTERED STATEMENT  1300--- NEW TEXT:
       J=J+A(INDEX)

COMMAND?
DO ADD 1300 1300 J=A(INDEX)+1,A(200)
 PROCESSING: DO ADD 1300 1300 J=A(INDEX)+1,A(200)
 CREATED STATEMENT  1250:
       DO 29997 J=A(INDEX)+1,A(200)
 CREATED STATEMENT  1325:
 29997 CONTINUE

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK DOTEST
   200       SUBROUTINE DOTEST
   300 CBEGDOC
   400 C     THIS ROUTINE HELPS TEST DO COMMANDS.
   500 C
   600 CENDDOC
   700 C
   800       DIMENSION A(50)
   900 C
  1000 30000 IF(A(1).EQ.0)GOTO 29999
  1100       KOUNT=1
  1150       DO 29998 INDEX=49,1,-1
  1200       A(INDEX)=A(INDEX+1)
  1250       DO 29997 J=A(INDEX)+1,A(200)
  1300       J=J+A(INDEX)
  1325 29997 CONTINUE
  1350 29998 CONTINUE
  1400 29999 RETURN
  1500       END
```

Figure E, Page 16 -- Complete ISUS Output Example (Continued)

```
COMMAND?
MODULE = DELTEST
 PROCESSING: MODULE = DELTEST
 SAVING MODULE DOTEST
 MODULE SAVED,  19 STATEMENTS WRITTEN.
 LOADING MODULE DELTEST.
 MODULE LOADED,  21 STATEMENTS READ.

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK DELTEST
   200       SUBROUTINE DELTEST(M,A,H)
   300 CBEGDOC
   400 C     THIS ROUTINE TESTS CONSEQUENCE ANALYSIS OF DEL COMMAND.
   500 CENDDOC
   600       DIMENSION M(10),H(2,3)
   700       DO 100 I=H(1,1),H(2,3)
   800       M(I)=A
   900 100   CONTINUE
  1000       IF(I.GT.100)GOTO 200
  1100       A=M(10)
  1200       GOTO 300
  1300 200   H(1,2)=A+1
  1400       H(1,3)=H(1,2)*5
  1500       M(1)=0
  1600 300   J=A+M(1)
  1700       WRITE(100)J
  1800       IF(A.EQ.1)GOTO 400
  1900       M(A)=J
  2000 400   RETURN
  2100       END

COMMAND?
COMMENT  *6*  THE UPDATE COMMANDS
 PROCESSING: COMMENT  *6*  THE UPDATE COMMANDS

COMMAND?
UPDATE ADD 400
 PROCESSING: UPDATE ADD 400
C     ALSO THE ADD AND REP COMMANDS.
 ADDED STATEMENT   410.
$END
```

Figure E, Page 17 -- Complete ISUS Output Example (Continued)

```
COMMAND?
UPDATE DEL 600
 PROCESSING: UPDATE DEL 600
 STATEMENT   600 DELETED.
 *** DELETED STATEMENT WAS A DECLARATION.
 *** THE FOLLOWING VARIABLES MAY NO LONGER BE CORRECTLY DEFINED.
 ***       M
 ***       H

COMMAND?
UPDATE DEL 700
 PROCESSING: UPDATE DEL 700
 STATEMENT   700 DELETED.
 *** START OR END OF A STRUCTURE HAS BEEN DELETED.
 *** PROGRAM MAY NO LONGER BE SYNTACTICALLY CORRECT.

COMMAND?
UPDATE REP 900
 PROCESSING: UPDATE REP 900
      REAL M(20),H(2,3)
$END
 ADDED STATEMENT   910.
 STATEMENT   900 DELETED.
 *** DELETED STATEMENT HAD A LABEL.
 *** PROGRAM MAY NO LONGER BE SYNTACTICALLY CORRECT.

COMMAND?
UPDATE DEL 1100
 PROCESSING: UPDATE DEL 1100
 STATEMENT  1100 DELETED.
 *** DELETED STATEMENT WAS ASSIGNMENT TO A VARIABLE.
 *** THE FOLLOWING LIST ARE THE STATEMENTS WHICH
 *** LOGICALLY FOLLOW THE DELETED STATEMENT.
 ***       1200   1600   1700   1800   1900   2000   2100
 *** THE FOLLOWING LIST IS OF THE STATEMENTS WHICH
 *** LOGICALLY FOLLOW THE DELETED STATMENT AND CONTAIN A
 *** REFERENCE TO THE VARIABLE ASSIGNED TO IN THE DELETED
 *** STATEMENT.
 ***       1600
 ***       1800
 ***       1900
```

Figure E, Page 18 -- Complete ISUS Output Example (Continued)

```
COMMAND?
UPDATE REP 2000
 PROCESSING: UPDATE REP 2000
400    IF(H(1,2).LT.100)GOTO 200
       RETURN
 ADDED STATEMENT  2010.
$END
 ADDED STATEMENT  2020.
 STATEMENT  2000 DELETED.
 *** START OR END OF A STRUCTURE HAS BEEN DELETED.
 *** PROGRAM MAY NO LONGER BE SYNTACTICALLY CORRECT.

COMMAND?
PRINT
 PROCESSING: PRINT
   100 *DECK DELTEST
   200       SUBROUTINE DELTEST(M,A,H)
   300 CBEGDOC
   400 C     THIS ROUTINE TESTS CONSEQUENCE ANALYSIS OF DEL COMMAND.
   410 C     ALSO THE ADD AND REP COMMANDS.
   500 CENDDOC
   800       M(I)=A
   910       REAL M(20),H(2,3)
  1000       IF(I.GT.100)GOTO 200
  1200       GOTO 300
  1300 200   H(1,2)=A+1
  1400       H(1,3)=H(1,2)*5
  1500       M(1)=0
  1600 300   J=A+M(1)
  1700       WRITE(100)J
  1800       IF(A.EQ.1)GOTO 400
  1900       M(A)=J
  2010 400   IF(H(1,2).LT.100)GOTO 200
  2020       RETURN
  2100       END
```

Figure E, Page 19 -- Complete ISUS Output Example (Continued)

Finally, there is an example of an UPDATE REPlace command that affects a program structure. As with the prior case, ISUS informs the user of the potential for problems with this change.

The PRINT command which follows shows the current contents of program. Note on this listing that the statements that have been deleted are not shown.

Pages 20, 21

Next the user issues a POS 100 command to position the ISUS statement cursor to the beginning of the module. Then a FIND command is used to search the module from the cursor for the first occurence of the token sequence H(1 which is located on statement 1300. Then the user issues a SUB command to substitute H(2 for all occurences of H(1 in statements 1300 to 1400.

Now the modifications are complete, and the user requests that two of the modules modified be copied to the compile file which will be created by ISUS at the end of the session by use of the COMPILE command. Note that a COMPILE command without a module name refers to the currently active module, and that COMPILE commands can appear anywhere in the command session; they do not have to be only at the end of the session.

The final command is the STOP command, and at this point ISUS indicates that it is writing both new master and compile files. This is done automatically at the end of every ISUS session.

```
COMMAND?
COMMENT  *7*  THE SEARCH COMMANDS
 PROCESSING: COMMENT  *7*  THE SEARCH COMMANDS

COMMAND?
POS 100
 PROCESSING: POS 100
   100 *DECK DELTEST

COMMAND?
FIND /H(1/
 PROCESSING: FIND /H(1/
  1300 200   H(1,2)=A+1

COMMAND?
SUB /H(1/H(2/ A 1300 1400
 PROCESSING: SUB /H(1/H(2/ A 1300 1400
  1300 200   H(2,2)=A+1
  1400       H(2,3)=H(1,2)*5
  1400       H(2,3)=H(2,2)*5
```

Figure E, Page 20 -- Complete ISUS Output Example (Continued)

```
COMMAND?
COMMENT  *8*  THE COMPILE COMMAND
 PROCESSING: COMMENT  *8*  THE COMPILE COMMAND

COMMAND?
COMPILE
 PROCESSING: COMPILE
 DELTEST WILL BE COPIED TO COMPILE FILE AT END OF SESSION.

COMMAND?
COMPILE DOTEST
 PROCESSING: COMPILE DOTEST
 DOTEST  WILL BE COPIED TO COMPILE FILE AT END OF SESSION.

COMMAND?
STOP
 PROCESSING: STOP
 SAVING MODULE DELTEST
 MODULE SAVED,  20 STATEMENTS WRITTEN.

 WRITING NEW MASTER AND COMPILE FILES.

STOP
```

Figure E, Page 21 -- Complete ISUS Output Example (Continued)

BIBLIOGRAPHY

These are references that describe other features of the SUS system, and also document some of the history of the SUS project.

1. E. F. Miller and J. S. Praninskas, "Semantic Update Systems -- A Conceptual Analysis, Interim Technical Report," SRA, RP-104, October 1977.

2. E. F. Miller and J. S. Praninskas, "Semantic Update Systems -- A System Specification," SRA Technical Report, RP-105, November 1977.

3. "Proposal for Semantic Update System Implementation," Submitted to U. S. Army Ballistic Research Laboratory, SRA PR-391 (Company Private), December 1977.

4. E. F. Miller, M. A. Hirschberg, and J. S. Praninskas, "A Semantic Updating System for Repairing Software," SRA, RN-418, 1 March 1978.

5. J. S. Praninskas, "An Approach to Side Effect Analysis in Semantic Update, with Particular Reference to COBOL Programs," SRA Internal Document, RN-459, July 1978 (Company Private).

6. W. G. Frickel, "A Semantic Update System for Software Maintenance," SRA Technical Briefing, BR-498, December 1978.

7. M. A. Hirschberg, W. G. Frickel, and E. F. Miller, "A Semantic Update System for Software Maintenance," Presented at 1979 CompCon Spring, San Francisco, California, 26 February 1979 -- 1 March 1979.

8. E. F. Miller, "A Semantic Update System for Software Maintenance," SRA Internal Technical Note, TN-506, 18 December 1979.

9. "Semantic Update System: SUS System Specification," SRA Reference Manual, RM-559/2, May 1979 (Revised August 1979).

10. R. J. Raker, "SUS01 Installation at BRL," SRA Technical Note TN-620/1, October 1979.

11. E. Sprinsock, "SUS Program Master File Desciption," SRA Technical Note, TN-660, December 1979.

12. E. Sprinsock, "SUS Graph Computations," SRA Technical Note, TN-672, January 1980.

13. R. J. Raker and E. Sprinsock, "SUS2 Installation at Ballistic Research Labratory," SRA Technical Note, TN-673, February 1980.

14. E. Sprinsock, "ISUS Command Reference Manual," SRA Reference Manual, RM-662, February 1980.

SEMANTIC UPDATE SYSTEM -- IMPLEMENTED COMMAND LIST

This summary gives the commands by classes that are implemented in the SEMANTIC UPDATE SYSTEM (ISUS). For more information, please see the individual command descriptions in the Command Reference Manual (Ref. 14).

1 BASIC COMMANDS

START
PASSWORD = password
ID = identifier
RUN = run-name
COMMENT any-text
DIR
STATUS
HELP
HELP number
STOP

2 DIMENSION COMMANDS

DIMENSION DEL variable position
DIMENSION ADD variable position extent
DIMENSION ADD variable position extent SET = (value)
DIMENSION CHANGE variable position new-extent
DIMENSION REORDER variable (order-list)
DIMENSION EXCHANGE variable position-1 position-2

3 CONSTANT/VARIABLE COMMANDS

CONSTANT CHANGE old-value new-value
CONSTANT DEL value
VARIABLE CHANGE old-variable new-variable
VARIABLE DEL variable

4 MODULE SELECTION COMMANDS

MODULE = module-name

5 UPDATE COMMANDS (CLASSIC UPDATE)

UPDATE ADD statement-number
UPDATE DEL statement-number
UPDATE REP statement-number
$END
RESTORE statement-number

6 PROGRAM PRODUCTION COMMANDS

```
COMPILE
COMPILE name
POS
POS statement-number
PRINT
PRINT ALL
VIEW
VIEW ALL
VIEW range
VIEW ALL range
```

7 LABEL REPLACEMENT COMMANDS

```
LABEL ADD statement-number label
LABEL CHANGE old-label new-label
LABEL DEL statement-number
```

8 DO STATEMENT MANIPULATION COMMANDS

```
DO ADD begin-statement end-statement text
DO CHANGE statement-number INDEX = variable
DO CHANGE statement-number INDEX = variable ONLY
```

9 SEARCH COMMANDS

```
FIND del pattern del
SUB del pattern del replacement del
SUB del pattern del replacement del A
SUB del pattern del replacement del range
SUB del pattern del replacement del A range
```

SEMANTIC UPDATE SYSTEM --- ALPHABETIC COMMAND LIST

This summary gives the commands implemented in the SEMANTIC UPDATE SYSTEM (ISUS) in alphabetic order. For more information please see the individual command descriptions in the Command Reference Manual (Ref. 14).

```
COMMENT any-text
COMPILE
COMPILE name
CONSTANT CHANGE old-value new-value
CONSTANT DEL value
DIMENSION ADD variable position extent
DIMENSION ADD variable position extent SET =  (value )
DIMENSION CHANGE variable position new-extent
DIMENSION DEL variable position
DIMENSION EXCHANGE variable position-1 position-2
DIMENSION REORDER variable ( order-list )
DIR
DO ADD begin-statement end-statement text
DO CHANGE statment-number INDEX = variable
DO CHANGE statment-number INDEX = variable ONLY
FIND del pattern del
HELP
HELP number
ID = identifier
LABEL ADD statement-number label
LABEL CHANGE old-label new-label
LABEL DEL statement-number
MODULE = module-name
PASSWORD = password
POS
POS statement-number
PRINT
PRINT ALL
RESTORE statement-number
RUN = run-name
START
STATUS
SUB del pattern del replacement del
SUB del pattern del replacement del A
SUB del pattern del replacement del range
SUB del pattern del replacement del A range
STOP
UPDATE ADD statement-number
UPDATE DEL statement-number
UPDATE REP statement-number
VARIABLE CHANGE old-variable new-variable
VARIABLE DEL variable
VIEW
VIEW ALL
VIEW range
VIEW ALL range
$END
```