Sponsored by
Advanced Research Projects Agency
ARPA Order 2311
ARPA Program Code No. 3D20

Contractor:
San Diego State University Foundation
5402 College Avenue
San Diego, California 92115

Title: Advanced Computer-Based Instruction

Effective Date of Contract: June 19, 1973
Contract Expiration Date: June 30, 1974
Amount of Contract: $126,702.85
Contract No. N61539-73-C-0184

Principal Investigators and Project Scientists:
Frederick Wm. Hornbeck, Ph.D. and Joseph R. Levine, Ph.D.
Department of Psychology
San Diego State University
5402 College Avenue
San Diego, California 92115
Telephone: (714) 286-5823 or 286-5358

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.
ADVANCED COMPUTER-BASED INSTRUCTION--
PROJECT CAMELOT: PRELIMINARY REPORT
ON A CAI SYSTEM

Lynn H. Brock, et al

California State University

Prepared for:
Naval Training Equipment Center
Advanced Research Projects Agency

21 February 1974

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151
**REPORT DOCUMENTATION PAGE**

<table>
<thead>
<tr>
<th>1. REPORT NUMBER</th>
<th>2. GOVT ACCESION NO.</th>
<th>3. RECIPIENT'S CATALOG NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE (and Subtitle)</th>
<th>5. TYPE OF REPORT &amp; PERIOD COVERED</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>7. AUTHOR(S)</th>
<th>8. CONTRACT OR GRANT NUMBER(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lynn H. Brock/Frederick Wm. Hornbeck</td>
<td>N61339-73-C-0184</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. PERFORMING ORGANIZATION NAME AND ADDRESS</th>
<th>10. PROGRAM ELEMENT, PROJECT, TASK AREA &amp; WORK UNIT NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Diego State University Foundation</td>
<td></td>
</tr>
<tr>
<td>5402 College Avenue</td>
<td></td>
</tr>
<tr>
<td>San Diego, CA 92115</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11. CONTROLLING OFFICE NAME AND ADDRESS</th>
<th>12. REPORT DATE</th>
<th>13. NUMBER OF PAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naval Personnel Research and Development Center</td>
<td>21 February 1974</td>
<td>25</td>
</tr>
<tr>
<td>San Diego, CA 92152</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>14. MONITORING AGENCY NAME &amp; ADDRESS (if different from Controlling Office)</th>
<th>15. SECURITY CLASS. (of this report)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naval Training Equipment Center, Orlando, Florida 32813</td>
<td>unclassified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16. DISTRIBUTION STATEMENT (of this Report)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved for public release; distribution unlimited</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>18. SUPPLEMENTARY NOTES</th>
<th>19. KEY WORDS (Continue on reverse side if necessary and identify by block number)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>computer assisted instruction</td>
</tr>
<tr>
<td></td>
<td>computer graphics</td>
</tr>
<tr>
<td></td>
<td>voice synthesisization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20. ABSTRACT (Continue on reverse side if necessary and identify by block number)</th>
</tr>
</thead>
</table>
| CAMELOT is a computer-assisted instructional system designed to support computer-generated graphics, computer-generated vocalizations, and computer-selected slide projection as well as full ASCII alphanumerical code. This report consists of a description of the concept and execution of CAMELOT, a description of the hardware configuration under development for the Naval Personnel Research and Development Center, San Diego (NAV), and the external specifications of the...
20. **Author Language GAIL (Graphics Assisted Instructional Language).**

GAIL is a rich language in comparison to other CAI languages—allowing a maximum of flexibility in strategies and procedures. The inclusion of a macro definition capability allows for efficiency and ease of coding. The GAIL compiler is written in IBM 360 Assembly Language. GAIL can be readily transported to any IBM 360 or 370 series machine. Complete documentation will be included in a NASA Task Report which should be available in the summer of 1978.
Contract: N61339-73-C-0184

Quarterly Technical Report

October 1, 1973 - December 31, 1973

Project CAPSLOT: Preliminary Report on a CAI System
While some effort has been expended on the other items on this contract, the most substantial effort and most significant accomplishments have all involved contract item 4. This item entails the design and delivery of hardware and software components of a computer system to support instructional activities incorporating computer-generated graphic displays, computer-generated vocalizations, and program-selected slides from a random access slide projector. The software also provides for the collection of data specified by courseware authors. This report contains a description of the concepts and facilities incorporated in the software package, a description of the hardware configuration, and the external specifications of the author language in modified BNF notation. Primary responsibility for the design and implementation of this powerful but inexpensive computer-assisted instructional system has been delegated to Mr. Lynn Brock.

Concepts and Facilities

This project was undertaken initially to provide a low-cost graphic display system in support of ongoing research activities at the Naval Personnel Research and Development Center, San Diego (NPRDC). Support of a random-access slide projector is also required for this application. In the early stages of the design process, it was determined that the system could--at relatively little additional cost--also incorporate support of a voice synthesizer. This capability is expected to be of considerable value in the planned research activities of this contractor as well as others at NPRDC.

Having established GMIL as the acronym for the author language (Graphic Assisted Instructional Language) of this CAI system, the overall project acquired the name CAILOT. A fully-equipped CAILOT student station consists of a graphic display terminal, random-access slide projector, and a voice synthesizer. Program authors can select any of these devices for the presentation of information to the student (subject). Students can communicate to the program through the terminal keyboard (full ASCII upper
and lower case character set) and through the positioning of the crosshair
cursor on the graphic display screen.

Programs written in GRAIL are compiled and supported during execution by
software developed in IBM 360 Assembly language. CAMELOT will be quite
portable—subject to easy conversion to any operating system which runs on
the IBM 360-370 series of computers. Some of the noteworthy features of
these two CAMELOT subsystems are listed below.

GRAIL: Graphic Assisted Instructional Language

GRAIL is the 'author language' in the CAMELOT system and provides the
following features:

(1) Display of alphanumeric and/or graphic information. Upper and
lower case alphanumerics may be mixed (along with graphic
information) in the same display.

(2) Selection and display of any one of eighty slides located in the
Random Access Slide Projector.

(3) Vocalization of phonemically encoded messages. Messages may
consist of phonemes, words (from the dictionary file), or larger
textual passages.

(4) Control of the sequence of presentation and analysis, both on
logical condition testing and iteratively. Logical combinations
(and, or) of relational (equal to, not equal, less than, etc.)
expressions, with grouping (indicated via parenthesis) is allowed.

(5) Input from the terminal includes upper and lower case alphanumerics
as well as the current position of the crosshair cursor (upon
instruction).

(6) A significant level of 'compile time' checking including
(a) checks for proper 'nesting' of constructs (i.e., if one
construct must be wholly contained within another then an
error message results if the author violates the restriction).
(b) checks for proper data types (e.g., a character string variable
used in an arithmetic expression is invalid).

Although the above features are almost always encountered in CAI author
languages, the following are relatively unique:

(7) Re-entrant code, which allows several students to be 'taught' using
the same machine instructions, thereby saving storage.

(8) Expressions (character and arithmetic) are allowed virtually
anywhere a variable can occur.

(9) Extensible language facility. The Macro Definition Language (MDL)
is an integral part of GRAIL and allows additional operations and
constructs to be added to the language by the author. The MDL allows the author to have virtually complete freedom in the selection of basic operations or strategies to be used in teaching a course. Facilities in the MDL allow the author to add new commands to the language and examine the operands coded for each use of such operands. As a result of examining those operands, and additionally on the basis of information retained from previous users of the same or other commands, the author has complete control over the resulting operations, and can repetitively generate the same operation (2), or any sequence of operations as desired.

(10) Dynamic storage allocation. Main (core) storage is allocated only when the structure requiring it is entered, thus reducing main storage use.

(11) Portability. All operating system dependent code is isolated into one area, allowing easy conversion to any operating system which runs on the IBM 360-370 series of computers.

(12) Generation of machine code. Actual 'machine code' (computer instructions) is generated by GNAIL. This is typically 5 to 10 times faster in execution than 'interpretive' systems.

EXCALIBUR: Extract and Collect Logically Indicated Basic User Records

EXCALIBUR provides for the collection, maintenance and selection of author selected information with the following features.

(1) Author controlled recording at any time, in addition to recording student logon, logoff times.

(2) FORTRAN (as well as PL/I, COBOL and ASSEMBLER) compatible records, allowing data analysis to be done on any machine supporting magnetic tape and FORTRAN.

(3) Easy selection of records for further analysis through very simple PL/I programs.

Hardware

In our development of CAMSLOT facilities, we have used established standards (e.g. ASCII character code) and industry conventions (e.g. RS-232-C hardware interfaces) whenever possible. Consequently, CAMSLOT can be implemented in a variety of configurations other than that described here. In particular, GNAIL-programs can be used with terminals other than those we are using.
Hardware costs for graphic display capability have been kept at low level through use of a storage-tube device—the Tektronix 4012 Computer Display Terminal. Software is being developed for support of the somewhat more expensive but extremely versatile Tektronix 4014 Computer Display Terminal. With its enhanced graphics option, the 4014

(1) has a display area 11 inches high by 15 inches wide,
(2) has the full ASCII character set (94 printing characters),
(3) has four program-selectable character sizes (from 74 characters per line with 35 lines per display to 133 characters per line with 64 lines per display),
(4) has a vector drawing time of 4000 inches/second,
(5) has—in discrete plot mode—a resolution of 4096 by 4096 addressable points,
(6) has—in interactive mode—1021 points horizontally and 780 points vertically addressable by the thumbwheel controlled crosshair cursor; viewing completely free from parallax,
(7) depending on option selected—either strap selectable or switch selectable input/output data rates of 110, 150, 300, 600, 1200, 2400, 4800 and 9600 bits/second,
(8) has limited vector refresh capability (which allows the temporary display of a limited number of vectors—only one or two—at the 1200 baud data transmission rate—on the otherwise permanent display), and
(9) has gray scale capability.

Permanent hard copies of terminal displays can be obtained by attaching a Tektronix 4610 hard copy unit to the terminal. Hard copy generation may be initiated by a switch on the terminal keyboard, a switch on the hard copy unit, or by computer command. An optional multiplexer allows a single hard copy unit to service up to four terminals.

Richer visual displays are permitted through the inclusion of a Kodak HA-960 random access slide projector in the student station configuration. Any of the 80 slides in the carousel can be selected and displayed under program control. The projector can also be turned on or off by the computer.

The presentation of computer-generated vocalizations is accomplished through a Federal Signal works Votrax (model 5.1) voice synthesizer. This inexpensive device allows the programmer to select the auditory mode of communication for such purposes as addressing preliterate or blind students, conveying additional information without disturbing the visual display, providing instruction in phonics, or simply to provide emphasis through use of an alternate medium. The
VOTRAX provides great flexibility in the presentation of vocal information, since it has an unlimited vocabulary of words, and has none of the mechanical timing peculiarities of cassette tapes or disks.

A purchase order has been placed with Sensors, Data, Decisions, Inc. of San Diego for the design, fabrication, and installation of the Student Station Interface (SSI). This device will pass all eight-bit codes except those specified for device selection. Device selection codes will switch the computer output data stream to the appropriate student station device. (Even the device selection codes can be passed through the SSI using special notation.)

Computer

Our present system configuration is designed to support a maximum of four simultaneous CAILOT users. We expect to be able to accomplish this with a dedicated area of 64K bytes on our IBM System 360/40 computer. Support of a larger number of simultaneous users would require a larger dedicated area and/or the implementation of a timesharing operating system. (We now run under IBM's DOS)

Disk storage utilization will depend on the amount of courseware to be supported and the number of students to use the system. Student records will be dumped to magnetic tape on a regularly scheduled basis--probably daily.

Communications

Because of the small number of terminals in the present configuration, it has not been necessary to resort to multiplexing to obtain cost effectiveness. All communications will be by voice-grade lines on the direct dial network at 1200 baud. Interfacing at the terminal end is via a Bell System 1000A Data Access Arrangement and a General DataComm 202-9A asynchronous modem. At the computer end, a Data General NOVA 1220 mini computer (on the main computer's multiplexer channel) handles intra-line editing of terminal-generated input to the computer. The phone line--NOVA interface is accomplished via a Bell System 1001A Data Access Arrangement and a General DataComm 202-9A asynchronous modem.

Hardware Configuration Constraints

CAILOT is suitable for installation on a wide variety of hardware configurations but there are two constraints aside from obvious size considerations (i.e. more terminals require more core in both the NOVA and the 360). First, the system is easily transportable only to configurations in which the main computer is of the IBM 360-570 series. Second, communications rates of at least 1200 baud are desirable for any significant use of graphics. Rates up to 9600 baud can be maintained by all components of the configuration described here except the voice synthesizers, modems, and teleprinter lines. Higher rates could be supported by locating the terminals in the immediate proximity of the computer to allow hard wiring, by
utilization of conditioned lines in the direct dial system, or by use of microwave or other high speed communications equipment.

GRAIL Language Specifications

GRAIL has been designed to provide maximum flexibility to the courseware author-programmer. This is an essential characteristic for a language developed specifically for research and development applications. At the same time, however, one can achieve the simplicity of other CAI author languages (e.g. PLATO, PLATO's TUTOR) through use of the Macro Definition Language (MDL) which has been incorporated in GRAIL. The presence of MDL should facilitate the simulation or emulation of other CAI systems (e.g. TICCIT) as well as the development of innovative or experimental procedures or strategies. The external specifications for GRAIL instructions follow. (In the interests of economy, we have simply copied a printout of our online version of the specifications rather than have them retyped.)
**NOTATION**

**SYNTACTIC VARIABLES:**

Anything beginning with the character \'#\' is replaced by the form indicated for the variable under "DEFINITIONS".

**METASYMBOLS:**

The characters ' (APOSTROPHE), '<', '>', and '/' are to be treated as metasyMBOLS unless they are enclosed in APOSTROPHES. They have the following meanings:

<, > are used to indicate that the enclosed form is optional and may be omitted at the user's discretion.

/ indicates that a choice must be made among the alternative forms which it separates; if a default is allowed, it precedes the first /, unless otherwise indicated by the text.

The form '....' indicates that the preceeding form may be repeated a number of times (as specified in the text) with each form separated by commas.

The form '....' indicates that the preceeding syntactic variable may be repeated a number of times.

Parentheses '(' and ')' always indicate that actual parentheses occur and in the form '<' form '<>', either . both parentheses must be present or both omitted.

APOSTROPHEs indicate that the enclosed character or characters are to be treated as themselves and not as metasyMBOLS; thus, the form "**" indicates an actual *single* APOSTROPHE - ' and '/' an
**LANGUAGE SPECIFICATIONS**

---

**@LABEL FRAME**

Storage is allocated for variables defined in the frame and $FRAME is set to @LABEL.

**FEND @LABEL**, ERASE=YES / NO>

ENDF defines the end of a frame, causing storage defined for the frame to be released. If the ERASE=NO option is specified then the screen is not erased, otherwise it is erased.

---

**@LABEL TEXT ("TEXT-STRING" / @EXP >...) , <POSIT=(<@LINENO> <, @POS>)

CAUSES the indicated strings of characters to be displayed. If either @LINENO or @POS is specified the string will be displayed at the current alpha-cursor position. If the POSIT parameter is specified, @LINENO indicates the line where the display is to begin and @POS the position within the line where the first character is to be displayed. If @LINENO is omitted, the current line is assumed and if @POS is omitted it is assumed to be 1.

Regardless of the LINE parameter (or in its absence) the alpha-cursor is moved to the first character of the next line after the characters are displayed, unless the CRLF=NO parameter is used, in which case the alpha-cursor will be left at the position immediately following the last character displayed.

---

**@LABEL SHOW (@EXP)**

The slide specified by @EXP is selected and displayed on the random access slide projector. If @EXP is greater than the maximum valid slide number for the terminal then the command is ignored. If the terminal does not have a slide projector (as indicated by the execution variable $SP) then the command is ignored.

---

**@LABEL FCALL @FORNAME,(@EXP-1, @EXP-2, @EXP-3,...)**

The FORTRAN subroutine identified by @FORNAME is called with the arguments specified by @EXP-1, @EXP-2,... The subroutine called must be known to the system or an error message will result during compilation. It is the user's responsibility to ensure that the types of the arguments passed agree with the types expected by the subroutine.

---

**@LABEL EXEC (@EXP)**

The section identified by the first 6 characters of the result of evaluating @EXP is executed, after which control returns to the next statement.
IF THE RESULT IS LESS THAN SIX CHARACTERS, THEN IT IS Padded WITH TRAILING BLANKS. THIS STATEMENT MAY OCCUR ONLY IN A COURSE COMPILE UNIT.

@LABEL COURSE
DEFINES A COURSE WITH THE INDICATED NAME, CAUSES STORAGE TO BE ALLOCATED FOR VARIABLES DEFINED IN THE COURSE AND THE SCREEN TO BE ERASED. $COURSE IS SET TO @LABEL AND $FRAME AND $SECTION ARE SET TO NULL.

CRSEND @LABEL
DEFINES THE END OF A COURSE AND CAUSES ALL STORAGE ALLOCATED FOR THE COURSE TO BE RELEASED.

@LABEL IF (@CON-CLS)
THEN <NULL>
ELSE <NULL>
IEND @LABEL

THESE FOUR OPERATIONS ALLOW FOR THE CONDITIONAL EXECUTION OF SUBSEQUENT STATEMENTS AS FOLLOWS -

1. IF @CON-CLS IS TRUE, THE STATEMENTS BETWEEN THE THEN AND ELSE OPERATIONS ARE EXECUTED, AFTER WHICH CONTROL PASSES TO THE STATEMENT FOLLOWING THE IEND.

2. IF @CON-CLS IS FALSE, THE STATEMENTS BETWEEN THE ELSE AND IEND ARE EXECUTED, AFTER WHICH CONTROL PASSES TO THE STATEMENT FOLLOWING THE IEND.

3. NO STATEMENTS MAY OCCUR BETWEEN THE IF AND THEN OPERATIONS.

4. IF THE 'NULL' OPTION IS SPECIFIED FOR A THEN OR ELSE OPERATION, AND IF CONTROL IS PASSED TO IT BY THE IF OPERATION, THE CONTROL IMMEDIATELY PassES TO THE STATEMENT FOLLOWING THE IEND.

5. IF CONSTRUCTS MAY BE 'NESTED', THAT IS - AN IF CONSTRUCT MAY OCCUR BETWEEN THE THEN AND ELSE, OR ELSE AND IEND, OPERATIONS OF ANOTHER IF CONSTRUCT.

6. ALL FOUR OPERATIONS COMPRISING THE IF CONSTRUCT ARE REQUIRED WHEN THE CONSTRUCT IS USED.

@LABEL DO WHILE / UNTIL (@CON-CLS)
< STATEMENTS >
IEND @LABEL

THESE TWO OPERATIONS ALLOW FOR THE REPETITIVE EXECUTION OF A GROUP OF STATEMENTS AS FOLLOWS -
WHEN CONTROL ENTERS THE DO STATEMENT, THE VALUE OF
@CON-CLS IS TESTED. IF IT IS TRUE (FALSE) AND THE
WHILE (UNTIL) FORM WAS USED, THEN THE STATEMENTS
BETWEEN THE DO AND DEND ARE EXECUTED, AFTER WHICH
CONTROL AGAIN ENTERS THE DO STATEMENT.
WHENEVER CONTROL ENTERS THE DO STATEMENT AND
@CON-CLS IS FALSE (TRUE) AND THE WHILE (UNTIL) FORM WAS
USED, THEN CONTROL MOVES TO THE STATEMENT
FOLLOWING THE DEND.

*LABEL
DOINC @FVAR, @FEXP-1 TO, @FEXP-2 BY, @FEXP-3
DODEC < STATEMENTS >
DEND *LABEL
WHEN CONTROL ENTERS THE DOINC (DODEC) STATEMENT
@FEXP-1 IS EVALUATED AND ASSIGNED TO @FVAR;
@FEXP-2 IS EVALUATED AND SAVED;
@FEXP-3 IS EVALUATED AND SAVED (IF @FEXP-3
IS OMITTED IT IS ASSumed EQUAL TO ONE).
@FVAR IS NOW COMPARED TO @FEXP-2, AND IF @FVAR IS
GREATER THAN (LESS THAN) @FEXP-2 THEN CONTROL MOVES TO
THE STATEMENT FOLLOWING THE DEND. IF @FVAR IS LESS
THAN OR EQUAL (GREATER THAN OR EQUAL) TO
@FEXP-2 THEN THE STATEMENTS
BETWEEN THE DOINC (DODEC) AND THE DEND STATEMENT ARE
EXECUTED. WHEN CONTROL MOVES TO THE DEND STATEMENT
THE SAVED EVALUATION OF @FEXP-3 IS ADDED TO
(SUBTRACTED FROM) @FVAR. CONTROL IS THEN TRANSFERRED
TO THE POINT IN THE DOINC (DODEC) WHERE @FVAR IS
COMPARSED TO @FEXP-2 AND THE ABOVE PROCESS IS
REPEATED.

*LABEL
CASES @FLIT < EXEC=FIRST / ALL>
CASE (@CON-CLS=1)
< STATEMENTS >
CASE (@CON-CLS=2)
< STATEMENTS >
CASE
.
.
.
.
CASE (@CON-CLS=@FLIT)
< STATEMENTS >
GEND *LABEL
THE CASES CONSTRUCT ALLOWS SELECTION OF GROUPS
OF STATEMENTS TO BE EXECUTED BASED UPON THE THE
LOGICAL VALUES OF CONDITIONAL CLAUSES. WHEN THE
CASES STATEMENT IS ENTERED THE @CON-CLS OF EACH
CASE STATEMENT IN THE CONSTRUCT IS EVALUATED,
BEGINNING WITH THE FIRST. IF THE @CON-CLS OF
ANY PARTICULAR CASE STATEMENT IS TRUE THEN THE
STATEMENTS BETWEEN THAT CASE STATEMENT AND THE
NEXT CASE IN EACH STATEMENT OF THE CONSTRUCT

ARE EXECUTED. IF THE EXEC=ALL OPTION IS SPECIFIED THEN THE EVALUATION OF EACH OCON-CLS CONTINUES.

IF THE EXEC EXEC=FIRST OPTION IS SPECIFIED THEN ONLY* THE STATEMENTS BETWEEN THE FIRST CASE STATEMENT WHICH EVALUATED TO TRUE AND THE NEXT CASE OR END STATEMENT WILL BE EXECUTED.

*IF EITHER EXEC OPTION AT LEAST ONE OCON-CLS *MUST* BE TRUE OR EXECUTION OF THE COURSE WILL BE TERMINATED WITH AN ERROR.

OLIT IS USED TO SPECIFY THE NUMBER OF CASE STATEMENTS WITHIN THE CURRENT CASES CONSTRUCT AND MUST BE EXACTLY EQUAL TO THE NUMBER OF CASE STATEMENTS.

LABEL READ <@VAR><.POSIT=(<@LINENO>,<@POS>)>

CAUSES CHARACTERS TO BE READ FROM THE TERMINAL INTO @VAR. IF MORE CHARACTERS ARE ENTERED THAN @VAR CAN HOLD, THEY ARE LOST. IF @VAR IS OMITTED THEN THE EXECUTION VARIABLE SANSVAR IS USED AND HAS A MAXIMUM LENGTH OF 72. IF THE LINE OPTION IS SPECIFIED THEN THE ALPHA-CURSOR IS MOVED TO THE INDICATED POSITION ON THE SCREEN BEFORE INPUT IS ACCEPTED (SEE 'TEXT').

LABEL CALC @VAR,(@EXP)

@EXP IS EVALUATED AND ASSIGNED TO @VAR. @EXP AND @VAR MUST BE OF THE SAME TYPE. IF A CHARACTER STRING ASSIGNMENT IS PERFORMED AND TRUNCATION OCCURS THEN THE EXECUTION VARIABLE SSTRING WILL BE SET TO THE NUMBER OF CHARACTERS TRUNCATED, OTHERWISE IT WILL BE SET TO ZERO.

LABEL READCC <@FVAR-1>,@FVAR-2,<@VAR>

THE CROSSHAIR CURSOR IS ILLUMINATED AND IT'S X-Y COORDINATES ARE READ WHEN THE NEXT CHARACTER IS TYPED BY THE USER. THE X VALUE, THE Y VALUE AND THE CHARACTER TYPED ARE READ INTO @FVAR-1, @FVAR-2 AND @VAR RESPECTIVELY. IF ANY (OR ALL) OF THE OPERANDS ARE OMITTED, THE VALUES ARE READ INTO THE EXECUTION VARIABLES $FCX, $FCY AND $CHR RESPECTIVELY. NOTE THAT THE X AND Y VALUES READ ARE IN SCREEN COORDINATES AND THAT THE RESULTING LENGTH OF @VAR (OR $CHR) WILL ALWAYS BE ONE.

LABEL RECORD <@GEXP-1,...,@GEXP-N> TYPE=(@GEXP)

@GEXP-1,...,@GEXP-N ARE CONCATENATED AND WRITTEN ON THE RECORDER FILE. THE RECORD IS IDENTIFIED BY THE FIRST TWO CHARACTERS OF THE CHARACTER STRING RESULTING FROM EVALUATION OF THE @GEXP SPECIFIED FOR TYPE. IT IS THE USER'S RESPONSIBILITY TO ENSURE THAT THE RESULTING RECORD HAS THE CORRECT FORMAT FOR IT'S TYPE, AND THAT THE TYPE ITSELF IS VALID.

LABEL DELAY @GEXP
EXECUTION OF SUBSEQUENT OPERATIONS IS DELAYED BY THE SPECIFIED NUMBER OF SECONDS.

**@LABEL** ESCAPE **@LABEL**

THIS COMMAND ALLOWS EXIT FROM WITHIN A NEST OF 'IF', 'DO', 'DODEC', 'DOING' AND/OR 'CASES' CONSTRUCTS. WHEN CONTROL ENTERS THE ESCAPE STATEMENT THE DEND, IEND OR CEND HAVING THE MATCHING **@LABEL** WILL BE THE NEXT STATEMENT EXECUTED. NOTE THAT THE STATEMENT HAVING THE MATCHING **@LABEL** *MUST* BE WITHIN THE ACTIVE NEST AT THE TIME THE ESCAPE STATEMENT IS EXECUTED.

**LIMIT @FLIT**

THE LIMIT STATEMENT IS USED TO SPECIFY AN UPPER BOUND ON HOW MANY TIMES THE STATEMENTS WITHIN A 'DO', 'DODEC' OR 'DOING' MAY BE EXECUTED.

IF NO LIMIT STATEMENT IS PRESENT THEN EACH SUCH CONSTRUCT IS LIMITED TO 100 REPETITIONS. IF A LIMIT STATEMENT IS PRESENT, THEN THE NEXT (AND *ONLY* THE NEXT) 'DO', 'DOINC' OR 'DODEC' CONSTRUCT IN THE SOURCE PROGRAM WILL BE LIMITED TO @FLIT REPETITIONS. @FLIT MAY BE LESS THAN 100 BUT MUST BE GREATER THAN ZERO.

**@LABEL** ERASE

THE SCREEN IS ERASED. IF THE TERMINAL IS NOT CAPABLE OF BEING ERASED THEN THE STATEMENT HAS NO EFFECT.

**VTYPE** C(@L-1, @U-1), F(@L-2, @U-2), E(@L-3, @U-3)

THIS OPERATION IS USED TO INDICATE THAT ANY VARIABLE OR FUNCTION REFERENCE IN SUBSEQUENT STATEMENTS WILL HAVE ITS TYPE DETERMINED AS FOLLOWS:

- IF THE FIRST CHARACTER OF THE VARIABLE OR FUNCTION IS IN THE RANGE @L-@U, THEN THE TYPE IS ASSUMED TO BE INDICATED BY THE CHARACTER PRECEDING THE PARENTHESIS IN WHICH THE RANGE WAS ENCLOSED.
- IF THE FIRST CHARACTER OF THE VARIABLE OR FUNCTION DOES NOT FALL IN ANY OF THE RANGES, THEN THE TYPE IS ASSUMED TO BE 'CHARACTER'.
- AT THE TIME THE VTYPE STATEMENT IS ENCOUNTERED THE FOLLOWING ITEMS ARE CHECKED:
  1. THE RANGES MUST BE DISJOINT IE. THE RANGES MUST NOT OVERLAP.
  2. @L-@U MUST BE LESS THAN OR EQUAL TO @U-@U FOR EACH RANGE.
  3. ONLY ONE VTYPE STATEMENT IS ALLOWED PER COMPILE UNIT, AND IT MUST IMMEDIATELY FOLLOW THE COURSE OR SECTION STATEMENT FOR THE COMPILE UNIT.

**VDEF** @VAR-1<@MAXLEN>, @VAR-2<@MAXLEN>...
@VAR-1, @VAR-2... ARE DEFINED AND RESERVED STORAGE BY THEIR APPEARANCE IN THE VDEF STATEMENT. THE TYPE (C, F OR E) OF EACH VARIABLE IS DETERMINED.
BY THE FIRST CHARACTER OF IT'S NAME IN
CONJUNCTION WITH THE VTYPE STATEMENT CURRENTLY IN
FORCE. IF THE LEVEL-COURSE OPTION IS USED
THEN THE STORGE IS MERELY DESCRIBED AND NO STORAGE
IS RESERVED, SINCE THE STORAGE WILL BE ALLOCATED
BY THE COURSE WHICH EXECUTES THE SECTION.
IF THE LEVEL-SECTION OPTION IS USED THEN THE
STORAGE IS DEFINED FOR ALL FRAMES IN THE CURRENT
SECTION.
IF THE LEVEL OPTION IS NOT USED THEN THE VARIABLES
ARE DEFINED (AND MAY BE USED) ONLY AT THE CURRENT
LEVEL.
A NAME MAY APPEAR IN ONLY ONE VDEF OR CDEF
STATEMENT WITHIN THE COMPILE UNIT.

CDEF @VAR-1(@LIT),@VAR-2(@LIT),...
@VAR-1,@VAR-2,... ARE DEFINED, RESERVED STORAGE AND
SET TO THE VALUE INDICATED BY THE ASSOCIATED
@LIT. QUANTITIES DEFINED IN THIS WAY MUST NOT HAVE
THEIR VALUE CHANGED DURING EXECUTION. MORE EFFICIENT
EXECUTION WILL RESULT IF THE NUMBER OF CDEF
STATEMENTS IS KEPT TO A MINIMUM.
ALL OTHER RULES WHICH APPLY TO VARIABLES DEFINED
BY THE VDEF STATEMENT APPLY TO THE CDEF
STATEMENT WITH THE EXCEPTION THAT THE LENGTH
OF CHARACTER VARIABLES IS DETERMINED BY THE
LENGTH OF THE ASSOCIATED @LIT.

$CGRAM- 1 TO 6 CHARACTER VARIABLE
CONTAINS THE NAME OF THE CURRENT FRAME DURING EXECUTION.

AT THE COURSE LEVEL, IT CONTAINS THE NAME OF THE MOST
RECENTLY EXECUTED FRAME.

$SLINE- INTEGER
GIVES THE CURRENT LINE NUMBER OF THE ALPHA-CURSOR

$FPOS- INTEGER GE
GIVES THE CURRENT POSITION, WITHIN THE LINE, OF THE
ALPHA-CURSOR.

$CDATE- 6 CHARACTER VARIABLE
CONTAINS THE CURRENT DATE IN THE FORM MM/DD/YY.

$SCDOV- 6 TO 9 CHARACTER VARIABLE CONTAINING THE CURRENT
DAY OF THE WEEK IE. MONDAY, TUESDAY, ...

$SCNTH- 3 TO 9 CHARACTER VARIABLE CONTAINING THE CURRENT
MONTH IE. JANUARY, MAY, ...

$SCFNAH- 0 TO 40 CHARACTER VARIABLE
CONTAINS THE FIRST NAME OF THE STUDENT CURRENTLY
RUNNING.

$SCLNAH- TO 40 CHARACTER VARIABLE
CONTAINS THE LAST NAME OF THE STUDENT CURRENTLY RUNNING.

$SPNAM- 9 CHARACTER VARIABLE
CONTAINS THE NUMBER OF THE STUDENT CURRENTLY RUNNING.

$SCOURSE- 1 TO 6 CHARACTER VARIABLE
CONTAINS THE NAME OF THE COURSE CURRENTLY RUNNING.
S Y N T A C T I C  V A R I A B L E S

@CON-CLS 1 = @REXP / <(> @REXP : @LOP : @REXP <)>  
@LOP 1 = EQ / NE / GT / LT / GE / LE / NL / NG  
@FVAR 1 = @LOP / AND  
@EXP 1 = < @FEXP / @EEXP >  
@EEXP 1 = ( @FVAR / @FEXP / @FLIT / <(> @FEXP @AOP @FEXP <)> )  
@AOP 1 = + / - / * / / / %  ( // IS THE REMAINDER OPERATOR )  
@COP 1 = )  
@FVAR 1 = @FNAME ( @EXP, @EXP, ... )  
@VARN 1 = @FVAR / @FVAR / @FVAR  
@VAR 1 = @FVAR , @EVAR 1 TO 6 ALPHANUMERIC CHARACTERS ,  
THE FIRST OF WHICH MUST BE ALPHABETIC.  
@FLIT 1 = @FLIT / @ELIT / @CLIT  
@EEXP 1 = @FLIT / @EEXP / @FLIT  
@CLIT 1 = " " @STRING " "  
@FLIT 1 = 0 @FLIT  
@UFLIT 1 = 0 @UFLIT  
@ELIT 1 = @FLITEP < @UFLIT > < @EFLIT >  
@C 1 = * / + / -  
@D 1 = 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9  
@P 1 = 0  
@EEXP 1 = @C / @STRING @C  
@C 1 = " A / B / C / ... / Z / # / O / AOP / @COP / ' ' / " " / @ / ? / / /  
/ / ( / / / ) / < / > / @P / @ / =  
@LABEL 1 = 0A / 0@LABEL / 0A0D  
REQURED LABEL , WHICH MUST BE 6 OR FEWER  
CHARACTERS LONG.  
@TEXT = @STRING 1 = @STRING  
CERTAIN CHARACTERS WILL CAUSE THE EFFECTS NOTED  
NOTED BELOW IF PRESENT IN @STRING -  
# = AT SIGN  
INDICATES THAT THE SINGLE LETTER IMMEDIATELY  
FOLLOWING IT IS TO BE DISPLAYED AS UPPER CASE.  
@O = TWO AT SIGNS  
INDICATES THAT ALL LETTERS BETWEEN THE @S  
AND THE NEXT # ARE TO BE DISPLAYED AS UPPER CASE.  
_ = UNDERSCORE  
INDICATES THAT THE TEXT BETWEEN IT AND THE NEXT  
UNDERSCORE IS TO BE UNDERLINED.  
ANY APOSTROPHES ( ' ) OR AMPERSANDS (@) WHICH ARE  
TO BE DISPLAYED MUST BE PRESENT TWICE.
PHONETIC INPUT DATA FORMAT

PHONEMES ARE IDENTIFIED BY THE OPPS TO THREE CHARACTER CODE USED IN THE VOTRAK LITERATURE, AND ARE SEPARATED FROM THE INFLECTION BY ONE OR MORE COMMAS OR BLANKS. THE INFLECTION IS INDICATED BY A SINGLE DIGIT - 1 FOR IN1, 2 FOR IN2, 3 FOR IN3, AND 4 FOR IN4. AN OMITTED INFLECTION IS ASSUMED TO BE 2.

EXAMPLE:

S, 1, AH1, IY, T, 3, R, UH3, 4, AH1, N, 3, IF, 1, K,
1, S, 1

IS ENCODED AS HEX

X'9F, D3, C9, 2A, 3B, 63, D5, CD, 85, 99, 9F'
<table>
<thead>
<tr>
<th></th>
<th>ERROR MESSAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>STYP0001 -</td>
<td>LITERAL OR VARIABLE NOT KNOWN</td>
</tr>
<tr>
<td>STYP0002 -</td>
<td>INVALID FIXED POINT LITERAL</td>
</tr>
<tr>
<td>STYP0003 -</td>
<td>FIXED POINT LITERAL TO LARGE</td>
</tr>
<tr>
<td>STYP0004 -</td>
<td>INVALID VARIABLE</td>
</tr>
<tr>
<td>STYP0005 -</td>
<td>INVALID FLOATING POINT LITERAL</td>
</tr>
<tr>
<td>STYP0006 -</td>
<td>FLOATING POINT LITERAL TO LARGE</td>
</tr>
<tr>
<td>STYP0007 -</td>
<td>EXPONENT LITERAL ERROR</td>
</tr>
<tr>
<td>STYP0008 -</td>
<td>NAME NOT TYPED PER VTYPE STATEMENT</td>
</tr>
<tr>
<td>STYP0009 -</td>
<td>IMPROPER USE OF QUOTATION MARKS</td>
</tr>
<tr>
<td>STYP0010 -</td>
<td>IMPROPER USE OF PARENTHESIS</td>
</tr>
<tr>
<td>CDEF0001 -</td>
<td>NULL SUBSTRING</td>
</tr>
<tr>
<td>CDEF0002 -</td>
<td>INVALID QUOTMARK IN NAME FIELD</td>
</tr>
<tr>
<td>CDEF0003 -</td>
<td>NO VARIABLE OR IMPROPER VARIABLE FIELD</td>
</tr>
<tr>
<td>CDEF0004 -</td>
<td>LABEL VS. VARIABLE TYPE ERROR</td>
</tr>
<tr>
<td>CDEF0005 -</td>
<td>NAME TYPE UNKNOWN MADE EQUAL TO LITERAL TYPE</td>
</tr>
<tr>
<td>CDEF0006 -</td>
<td>VARIABLE LENGTH NOT GREATER THAN 6 CHARACTERS</td>
</tr>
<tr>
<td>CDEF0007 -</td>
<td>NO LEVEL PARAMETER OR ERROR IN LEVEL PARAMETER</td>
</tr>
<tr>
<td>CDEF0008 -</td>
<td>CDEF GENERATED AT FRAME DEFAULT LEVEL</td>
</tr>
<tr>
<td>CDEF0009 -</td>
<td>NAME GREATER THAN 6 CHARACTERS LONG</td>
</tr>
<tr>
<td>CDEF0010 -</td>
<td>CDEF CODE NOT GENERATED</td>
</tr>
<tr>
<td>VDEF0001 -</td>
<td>NO LEVEL PARAMETER OR MISSING LEVEL PARAMETER</td>
</tr>
<tr>
<td>VDEF0002 -</td>
<td>VDEF GENERATED AT FRAME DEFAULT LEVEL</td>
</tr>
<tr>
<td>VDEF0003 -</td>
<td>NULL SUBSTRING</td>
</tr>
<tr>
<td>VDEF0004 -</td>
<td>NAME GREATER THAN 6 CHARACTERS LONG</td>
</tr>
<tr>
<td>VDEF0005 -</td>
<td>NAME TYPE UNKNOWN</td>
</tr>
<tr>
<td>VDEF0006 -</td>
<td>MAXLINE NOT NUMERIC</td>
</tr>
<tr>
<td>VDEF0007 -</td>
<td>INVALID MAXLINE NUMBER</td>
</tr>
<tr>
<td>VDEF0008 -</td>
<td>NAME TYPE VS. MAXLINE LENGTH ERROR</td>
</tr>
<tr>
<td>VDEF0009 -</td>
<td>CODE NOT GENERATED</td>
</tr>
<tr>
<td>VTPP0001 -</td>
<td>INVALID PARAMETER LENGTH</td>
</tr>
<tr>
<td>VTPP0002 -</td>
<td>INVALID PARAMETER FORMAT</td>
</tr>
<tr>
<td>VTPP0003 -</td>
<td>DUPLICATE PARAMETER</td>
</tr>
<tr>
<td>VTPP0004 -</td>
<td>INVALID G PARAMETER</td>
</tr>
<tr>
<td>VTPP0005 -</td>
<td>INVALID F PARAMETER</td>
</tr>
<tr>
<td>VTPP0006 -</td>
<td>INVALID E PARAMETER</td>
</tr>
<tr>
<td>VTPP0007 -</td>
<td>PARAMETERS OVERLAP</td>
</tr>
<tr>
<td>VTPP0008 -</td>
<td>VTYPE NEEDED ALLOWED THIS ONE</td>
</tr>
<tr>
<td>VTPP0009 -</td>
<td>ONLY ONE CORRECT VTYPE STATEMENT PER RUN</td>
</tr>
</tbody>
</table>
GRAIL FUNCTIONS

$SEFRMF(OFARG)$
The integer contained in $OFARG$ is converted to an F-type number. Note that the conversion is exact if and only if $OFARG$ is less than (approximately) 8 decimal digits.

$SFRME(OFARG)$
The floating point number contained in $OFARG$ is converted to an integer. This conversion is accomplished by ignoring any fractional part. If the resulting integer is to large in value ($>2^{31}-1$ or $<-2^{31}$) then the result is the largest integer of appropriate sign, and the execution variable $SFRME$ is set to 1. If the conversion is accomplished without error $SFRME$ is set to zero.

$GFRMF(OFARG)$
The F-type number identified by $OFARG$ is converted to a character string containing a decimal representation of the number. The form of the result is the number, preceded by a "-" if negative, with leading zeroes suppressed. A zero value is represented by a single zero with no sign. A 'negative zero' is not possible.

$SFRMC(OCARG)$
The string contained in $OCARG$ is converted to an F-type number. $OCARG$ may contain a string of the format 'D...S...B...', where the B's indicate optional spaces, the S indicates an optional sign (+ or -) and the D indicates decimal digits. If the value represented by the string is of too large a magnitude to be represented by an F-type number, then the result is the largest number of appropriate sign and $SFRMC$ is set to -1 or -2 depending on whether the sign of the result is positive or negative, respectively. If the string is not of the correct form, then a result of zero is returned and the execution variable $SFRMC$ is set to the position of the character in $OCARG$ where the error was detected. If no errors occur, $SFRMC$ is set to zero.

$SFRMC(OFARG<OFARG>)$
$OFARG$ is converted to a character string according to the following rules:

- If $OFARG$ is not specified then:
  - If the absolute value of $OFARG$ is less than or equal to 9999,9999 and greater than or equal to .0001 then the resulting character string has the form 'ZZZZ,9999' where the 2's indicate zero suppressed positions (which are not returned if they contain spaces), and the 9's indicate positions which will be zero filled until the last non-zero digit is encountered.
  - If the number is negative, then a '-' is inserted prior to the first character of the resulting string.
IF THE ABOVE CONDITION IS NOT MET THEN THE RESULTING STRING WILL HAVE THE FORM "ZZZZZZ.9999999EXXX" WHERE THE Z'S AND 9'S ARE HANDLED AS ABOVE, AS IS THE SIGN, THE 'E' IS AN ACTUAL CHARACTER IN THE RESULTING STRING AND THE 'S' IS THE SIGN (+ OR -) OF THE EXPONENT (WHICH IS REPRESENTED BY THE 'XX'). THE MEANING OF THIS FORM IS THAT THE NUMBER PRECEDING THE 'E' SHOULD BE MULTIPLIED BY TEN RAISED TO THE XX'TH POWER TO PROVIDE THE CORRECT VALUE. THIS FORM IS VERY SIMILAR TO STANDARD SCIENTIFIC NOTATION.

IF OFARG IS SPECIFIED THEN CONVERSION IS AS ABOVE EXCEPT THAT THE NUMBER OF DIGITS PRINTED AFTER THE DECIMAL POINT IS EQUAL TO OFARG. HOWEVER, IN NO CASE WILL MORE THAN 7 SIGNIFICANT DIGITS BE GENERATED, SINCE THIS IS THE PRECISION LIMIT OF THE COMPUTER.

IF THE RESULTING STRING IS LONGER THAN THE OUTPUT CHARACTER STRING INTO WHICH IT IS BEING STORED, THEN $EFRCMI (A GLOBAL FULL WORD VARIABLE) IS SET TO -1.

$EFRCMI($CARG)
THE CHARACTER STRING IN $CARG IS CONVERTED TO AN E-TYPE NUMBER ACCORDING TO THE FOLLOWING RULES -
THE STRING MUST BE OF THE FORM "B...SD...PD...B...ESX...B..." WHERE THE B'S INDICATE OPTIONAL BLANKS,
THE FIRST 'S' INDICATES AN OPTIONAL SIGN (+ OR -),
THE D'S INDICATE DECIMAL DIGITS TO BE INTERPRETED AS THE SIGNIFICANT DIGITS OF THE NUMBER,
THE 'P' INDICATES AN OPTIONAL DECIMAL POINT,
THE SECOND 'S' INDICATES THE SIGN OF THE EXPONENT AND IS OPTIONAL,
THE 'E' INDICATES THE LETTER E, WHICH IS REQUIRED IF AN EXPONENT IS SPECIFIED, AND THE X'S INDICATE THE EXPONENT WHICH IS ALSO OPTIONAL.

THE FOLLOWING ADDITIONAL RULES APPLY -
1. IF $CARG IS A NULL STRING, THEN $EFRCMI IS SET TO -1.
2. IF THE STRING REPRESENTS A NUMBER TOO LARGE IN MAGNITUDE TO BE STORED IN AN E-TYPE NUMBER, THEN $EFRCMI IS SET TO -2 AND THE RESULT IS SET EQUAL TO THE LARGEST NUMBER OF APPROPRIATE SIGN.
3. IF THE STRING REPRESENTS A NUMBER TOO SMALL TO BE STORED IN AN E-TYPE NUMBER, THEN $EFRCMI IS SET TO -3 AND THE RESULT IS THE SMALLEST NON-ZERO NUMBER OF APPROPRIATE SIGN.
4. IF ANY OTHER FORMAT ERROR IS ENCOUNTERED, THEN $EFRCMI IS SET TO THE POSITION IN $CARG WHERE THE ERROR WAS DISCOVERED AND THE RESULT IS ZERO.
5. IF NO ERRORS OCCUR, THEN $EFRCMI IS SET TO ZERO.

$INSTR($FARG,$CARG-1,$CARG-2,...)
$CARG-1 IS SEARCHED FOR OCCURRENCES OF $CARG-2.
$\text{CARG-3,... AND A VALUE OF TRUE IS RETURNED IF CARG-2, CARG-3,... ALL } '\text{OCCUR}' \text{ IN CARG-1. OTHERWISE, A VALUE OF FALSE IS RETURNED. } CARG-N \text{ IS SAID TO 'OCCUR' IN CARG-1 IF AT LEAST } (FLEN(CARG-N)FARG)/100 \text{ CHARACTERS OF } CARG-N \text{ OCCUR CONTINUOUSLY ANYWHERE IN CARG-1.}

$\text{INSTRO(CARG1,CARG-1,CARG-2,...)}$

\text{THIS FUNCTION BEHAVES LIKE } $\text{INSTR}$ \text{ WITH THE EXCEPTION THAT ANY 'OCCURANCE' OF CARG-N MUST FOLLOW THE 'OCCURANCE' OF CARG-(N-1) IN CARG-1.}

$\text{SCSBSTR(CEXP,ФEXP-1,ФEXP-2)}$

\text{A STRING OF CHARACTERS FROM CEXP, BEGINNING WITH THE 'ФEXP-1'\text{TH AND CONTINUING FOR ФEXP-2 CHARACTERS IS RETURNED. IF ФEXP-1+FEXP-2 EXCEEDS FLEN(CEXP) THEN ONLY THOSE CHARACTERS ACTUALLY IN CEXP ARE RETURNED. IF ФEXP IS ZERO OR NEGATIVE, OR ФEXP-2 IS NEGATIVE THEN THE EXECUTION VARIABLE FSBSBSTR IS SET TO -1.}

\text{AND A NULL STRING IS RETURNED. IF ФEXP-2 IS ZERO THEN A NULL STRING IS RETURNED AND FSBSBSTR IS SET TO ZERO.}

$\text{SCNXUD(CEXP-1,ФEXP<,ФVAR<,CEXP-2>>)}$

\text{THE NEXT 'WORD' IN CEXP-1 IS RETURNED. CEXP-1 IS SCANNED BEGINNING WITH THE CHARACTER AT POSITION ФEXP UNTIL THE FIRST CHARACTER WHICH IS *NOT* A DELIMITER IS FOUND. SUBSEQUENT CHARACTERS FORM THE RESULT UNTIL THE NEXT DELIMITER IS FOUND, AT WHICH TIME ФVAR IS SET (IF SPECIFIED) TO THE POSITION IN CEXP-1 OF THE CHARACTER WHICH CAUSED THE SCAN TO END. IF THE END OF CEXP-1 IS ENCOUNTERED DURING THE SCAN, THEN THE CHARACTERS PRIOR TO THE END OF THE STRING ARE RETURNED AND ФVAR IS SET TO ZERO IF IT WAS SPECIFIED.}

\text{IF ФEXP-2 IS SPECIFIED THEN EACH CHARACTER IN ФEXP-2 IS USED AS A DELIMITER; OTHERWISE, THE FOLLOWING CHARACTERS CONSTITUTE THE 'DEFAULT' DELIMITERS:}

\begin{itemize}
  \item BLANKS
  \item \text{?}
  \item \text{! (EXCLAMATION MARK)}
  \item \text{;}
\end{itemize}