2 Feb 73

Attached is one installment of a preliminary users' manual for the language system. Comments of all flavors are solicited. If you see something you don't like complain. If you see something you like, speak up, lest it go away.

Dave

LINE ADDRESSES

A FUNDAMENTAL CONSTRUCT IN THE LANGUAGE SYSTEM IS THE LINE ADDRESS. A LINE ADDRESS LOCATES A PARTICULAR LINE OF TEXT, BY BLOCK NUMBER WITHIN THE PROGRAM AND LINE NUMBER WITHIN THE BLOCK. A FAIRLY RICH SYNTAX IS PROVIDED FOR LINE ADDRESS EXPRESSIONS. USER-SETTABLE MODES ARE PROVIDED WHICH DETERMINE THE TOPOLOGY OF THE PROGRAM AS SEEN BY CONTEXT-SEARCH SUBEXPRESSIONS. "LINE POINTERS" ARE PROVIDED TO HOLD THE ADDRESSES OF IMPORTANT LINES IN THE PROGRAM.

```
LINEADDRESS = ABS [ RELLIST ]

   ABS = POINTER ! '/' FUNCTION '/'

      FUNCTION = IDENTIFIER

      IDENTIFIER = LETTER ! IDENTIFIER ( LETTER ! DIGIT )

   RELLIST = REL [ RELLIST ]

      REL = [ '+' ! '-' ] TARGET

         TARGET = INTEGER ! '"' STRING '"' ! ':' LABEL ':'

            LABEL = STRING

            STRING = CHAR [ STRING ]

            INTEGER = DIGIT [ INTEGER ]

      POINTER =  'A' ! 'B' ! 'C' ! 'D' ! '.' ! '*'
```

LINE ADDRESSES (CONT.)

SEMANTICS:

AS SHOWN IN THE SYNTAX ABOVE, A LINE ADDRESS CONSISTS OF AN ABSOLUTE ADDRESS, FOLLOWED BY AN OPTIONAL SEQUENCE OF RELATIVE ADDRESSES. EVALUATION CONSISTS OF STARTING AT THE ABSOLUTE ADDRESS, AND MAKING THE SEQUENCE OF RELATIVE MOVES SPECIFIED BY THE RELATIVE ADDRESSES.

AN ABSOLUTE ADDRESS IS EITHER A BLOCK NAME (FUNCTION NAME OR "GLOBAL") MEANING THE FIRST LINE OF THAT BLOCK, OR A POINTER NAME, MEANING THE LINE POINTED TO BY THAT POINTER. (SEE BELOW ABOUT POINTERS.)

A RELATIVE ADDRESS SPECIFIES A DIRECTION TO SEARCH ("+" = FORWARD, "-" = BACKWARD; DEFAULT = "+"), AND A TARGET FOR WHICH TO SEARCH. THE TARGET MAY BE AN INTEGER DISPLACEMENT, OR A CONTEXT SEARCH FOR EITHER AN ARBITRARY STRING, OR A LABEL. (NOTE: CONTEXT SEARCHES DO NOT START WITH THE LINE "GIVEN" TO THEM, BUT ALWAYS ATTEMPT TO MOVE AT LEAST ONE LINE.)

PROGRAM TOPOLOGY MODES

TWO INDEPENDENT USER-SETTABLE MODES ARE PROVIDED FOR CONTROLLING CONTEXT SEARCHES.

A) BLOCK/PROGRAM MODE:
   IN BLOCK MODE, THE PROGRAM IS SEEN AS A COLLECTION OF BLOCKS. THE RANGE OF A SEARCH IS THE BLOCK IN WHICH IT STARTS.
   IN PROGRAM MODE, THE PROGRAM IS SEEN AS A SINGLE BODY OF TEXT. THE RANGE OF A SEARCH IS THE ENTIRE PROGRAM.

B) LINEAR/CIRCULAR MODES:
   IN LINEAR MODE, THE SEARCH RANGE IS SEEN AS A LINEAR SEQUENCE OF LINES. A SEARCH CANNOT CONTINUE PAST THE END OF THE RANGE.
   IN CIRCULAR MODE, THE SEARCH RANGE IS SEEN AS CIRCULAR. A SEARCH CAN WRAP AROUND, BUT CANNOT CONTINUE PAST THE LINE AT WHICH IT WAS STARTED.

EACH MODE CAN BE SET, AND REMAINS IN EFFECT UNTIL EXPLICITLY CHANGED. THE DEFAULT MODES ARE PROGRAM AND CIRCULAR.

THE TOPOLOGY MODES, PLUS THE ABILITY TO SEARCH BACKWARD AS WELL AS FORWARD, SHOULD AID IN EASY AND CONFIDENT USE OF THE CONTEXT SEARCH FACILITIES.

LINE POINTERS

LINE POINTERS ARE NAMED VARIABLES IN THE LANGUAGE SYSTEM WHICH CAN BE USED TO REMEMBER EVALUATED LINE ADDRESSES. POINTERS "STICK" TO THEIR LINES, IN THE SENSE THAT THEY REMAIN VALID THROUGH INSERTIONS AND DELETIONS OF TEXT ELSEWHERE IN THE PROGRAM. A POINTER IS SET TO NULL ORIGINALLY, AND WHENEVER THE LINE TO WHICH IS WAS REFERRING IS DELETED.

THERE ARE CURRENTLY SIX POINTERS IN THE LANGUAGE SYSTEM, NAMED "A", "B", "C", "D", ".", AND "*".

CURRENT-LINE POINTER (".")

THE "CURRENT LINE" POINTER, DENOTED ".", IS JUST LIKE ANY OTHER LINE POINTER, EXCEPT THAT IT IS AUTOMATICALLY SET AS A SIDE EFFECT OF VARIOUS COMMANDS. THESE COMMANDS ATTEMPT TO SET "." TO THE LINE WHICH THE USER IS MOST LIKELY TO REFERENCE NEXT. THE DETAILS OF HOW THIS LINE IS SELECTED ARE GIVEN IN THE DESCRIPTIONS OF THE INDIVIDUAL COMMANDS.

ACTIVE-LINE POINTER ("*")

THE "ACTIVE LINE" POINTER, DENOTED "*", IS LIKE ANY OTHER LINE POINTER, EXCEPT FOR THREE SPECIAL CHARACTERISTICS.

A) IT IS SET BY THE LANGUAGE SYSTEM, AT APPROPRIATE TIMES, TO POINT TO THE ACTIVE STATEMENT-LINE. (I.E., THE ONE CONTAINING THE CURRENT P-COUNTER.)

B) IT IS EXAMINED BY THE LANGUAGE SYSTEM WHEN EXECUTION IS TO BE RESUMED. CONTROL PASSES TO THE STATEMENT-LINE TO WHICH IT POINTS.

C) ITS BEHAVIOUR WHEN EXAMINED OR SET BY THE USER IS SLIGHTLY PECULIAR.

TO BE MORE SPECIFIC:

A) AUTOMATICALLY SETTING *

1) WHENEVER EXECUTION OF THE PROGRAM IS SUSPENDED, * IS SET TO AGREE WITH THE P-COUNTER OF THE TOP STACK FRAME.
2) WHENEVER ONE OR MORE FRAMES ARE MANUALLY POPPED OFF THE STACK, * IS SET TO AGREE WITH THE P-COUNTER IN THE NEW TOP FRAME OF THE STACK.

B) RESUMING EXECUTION AT *

1) WHEN EXECUTION IS RESUMED, * MUST INDICATE A VALID RESUMPTION POINT IN THE PROGRAM. THAT IS, IT MUST EITHER AGREE WITH THE P-COUNTER OF THE TOP STACK FRAME OR POINT TO THE BEGINNING OF SOME STATEMENT LINE IN THE TOP FUNCTION IN THE STACK.
2) IF * IS MOVED FROM THE MIDDLE OF A STATEMENT, THE TOP OF THE STACK WILL BE CLEANED UP APPROPRIATELY. IT IS THE USER'S RESPONSIBILITY TO VERIFY THAT THE PROPOSED REROUTING OF THE FLOW OF CONTROL IS OTHERWISE REASONABLE.

C) EXAMINING AND SETTING *

1) IF * IS EXAMINED USING THE "=" COMMAND (SEE BELOW ABOUT =) WHEN IT IS POINTING INTO SOME PARTIALLY COMPLETED STATEMENT-LINE, AN EXTRA PIECE OF INFORMATION IS TYPED OUT: NAMELY, THE PERCENTAGE OF THE CODE FOR THAT STATEMENT LINE WHICH HAS BEEN EXECUTED.

EXAMPLE:

```
=*        MIGHT PRINT     /F/+6 (+35%)
```

2) IF * IS USED IN A LINE ADDRESS EXPRESSION, THIS EXTRA PERCENTAGE-COMPLETED INFORMATION IS DISCARDED. THUS, IN PARTICULAR, THE COMMAND *=* IS NOT NECESSARILY A NO-OP. IT CAUSES EXECUTION OF A PARTAILLY COMPLETED STATEMENT-LINE TO BE RESTARTED.

LINE GROUPS

A LINE GROUP IS A SEQUENCE OF CONTIGUOUS LINES. THE USUAL WAY OF DENOTING A LINE GROUP IS WITH A PAIR OF LINE ADDRESSES. IF THE GROUP CONTAINS ONLY A SINGLE LINE, THE SECOND LINE ADDRESS MAY BE OMITTED. LINE GROUPS ARE THE KEY CONSTRUCT IN MOST OF THE EDITING COMMANDS OF THE LANGAUGE SYSTEM.

SYNTAX:

LINEGROUP = LINEADDRESS ["," LINEADDRESS]

OPERATOR COMMANDS

CERTAIN COMMANDS WHICH ARE EXPECTED TO RECEIVE HEAVY USE ARE GIVEN A TERSE "OPERATOR-OPERANDS" STYLE OF SYNTAX. THE REST OF THE COMMANDS ARE EXPRESSED IN A MORE CONVENTIONAL "KEYWORD-PARAMETER LIST" STYLE. IT IS HOPED THAT THIS HYBRID APPROACH WILL AVOID FROUENT TYPING OF VERBOSE COMMANDS, WITHOUT INTRODUCING A LARGE NUMBER OF UNREADABLY TERSE BUT LITTLE USED OPERATOR SYMBOLS.

EACH OPERATOR IS DESCRIBED BELOW, ALONG WITH EXAMPLES OF ITS USE. IN GENERAL, THE OPERANDS ARE POINTER NAMES, LINE ADDRESSES, OR LINE GROUPS. IN CERTAIN CASES, AN OPERAND MY BE OMITTED. THE EXACT INTERPRETATION OF THIS DEPENDS ON THE OPERATOR, BUT IT ALWAYS MEANS, INTUITIVELY, "THE NEXT LINE(S) TYPED ON THE USER'S TERMINAL."

IN THE FEW CASES IN WHICH COMMANDS MAY CONTAIN MULTIPLE OPERATORS, EVALUATION IS DONE LEFT-TO-RIGHT AND THE ONLY PRIORITY RULE IS THAT UNARY OPERATORS ARE EVALUATED BEFORE BINARY OPERATORS.

THE "=" OPERATOR (BINARY)

THE RIGHT OPERAND OF THIS OPERATOR IS ALWAYS A LINE ADDRESS. THE LEFT OPERAND IS EITHER A POINTER NAME, OR MAY BE OMITTED. IN THE FORMER CASE, THE VALUE OF THE LINE ADDRESS IS ASSIGNED TO THE POINTER. IN THE LATTER, IT IS TYPED OUT ON THE TERMINAL.
NO VALUE IS PRODUCED BY THIS OPERATOR.

EXAMPLES

COMMAND	MEANING
A=.	REMEMBER TO CURRENT LINE WITH POINTER A
B=B+"XYZ"	ADVANCE POINTER B TO NEXT LINE WITH "XYZ"
=*	TYPE OUT THE ADDRESS OF THE CURRENTLY ACTIVE STATEMENT LINE.
.=	NOT MEANINGFUL (NO RIGHT OPERAND)

THE "←" OPERATOR (BINARY)

BOTH RIGHT AND LEFT OPERANDS OF THIS OPERATOR MAY BE LINE GROUPS, OR ONE (BUT NOT BOTH) MAY BE OMITTED. THE EFFECT IS TO REPLACE THE LEFT OPERAND WITH THE CONTENTS OF THE RIGHT OPERAND. THE VALUE PRODUCED IS THE SAME AS THE RIGHT OPERAND.

EXAMPLES

COMMAND	MEANING
A←B, B+1	REPLACE LINE AT A WITH TWO LINES AT B
←*	TYPE OUT THE ACTIVE LINE
.←	TYPE IN LINES TO REPLACE CURRENT LINE
A←B←C	REPLACE LINE AT A WITH LINE AT B. THEN REPLACE LINE AT B WITH LINE AT C.
←	NOT MEANINGFUL (BOTH OPERANDS MISSING)

THE "@" OPERATOR (UNARY)

THIS OPERATOR IS AMBIDEXTROUS; IT CAN BE EITHER PREFIX OR POSTFIX. ITS OPERAND IS A LINE ADDRESS, WHICH MAY BE OMITTED. IF THE OPERATOR IS PREFIX, IT PRODUCES AS VALUE THE EMPTY LINE GROUP PRECEDING ITS ARGUMENT. IF THE OPERATOR IS POSTFIX, IT PRODUCES THE EMPTY LINE GROUP FOLLOWING ITS ARGUMENT. IF THE OPERAND IS OMITTED, IT PRODUCES A "NON-SPECIFIC" EMPTY LINE GROUP. ("THE NEXT EMPTY LINE TYPED ON THE TERMINAL") THE MAIN USE OF THIS OPERATOR IS TO GREATLY EXTEND THE UTILITY OF THE "←" OPERATOR.

EXAMPLES

COMMAND	MEANING
.←@	DELETE THE CURRENT LINE
/F/@←	TYPE IN LINES TO BE APPENDED AFTER LINE 1 OF FUNCTION F.
@.←A, A+1←@	MOVE THE TWO LINES AT A TO JUST BEFORE THE CURRENT LINE.
@←.	NOT MEANINGFUL (@ WITH NO OPERAND DOES NOT REFER TO A SPECIFIC EMPTY LINE GROUP.)

DIGRESSION: SETTING OF . BY THE "←" OPERATOR

THE "←" OPERATOR SETS THE CURRENT LINE POINTER AS A SIDE EFFECT. THE LINE TO WHICH "." SHOULD POINT IS DETERMINED BY THE FOLLOWING CONFUSING PROCEDURE, WHOSE ONLY APPARENT VIRTUE IS THAT IT SEEMS TO DO WHAT IS MOST USEFUL MOST OF THE TIME.

1. IF THE LEFT OPERAND OF "←" WAS OMITTED, LEAVE "." UNCHANGED.
2. OTHERWISE, IF THE RIGHT OPERAND OF "←" WAS NOT AN EMPTY LINE GROUP, POINT "." TO THE LAST LINE OF THE COPY OF IT WHICH REPLACED THE LEFT OPERAND.
3. OTHERWISE, IF THE LEFT OPERAND DID NOT INCLUDE THE FIRST LINE OF THE BLOCK, POINT "." TO THE LINE PRECEDING THE LEFT OPERAND.
4. OTHERWISE, IF THE LEFT OPERAND DID NOT INCLUDE BOTH THE FIRST AND LAST LINES OF THE BLOCK, POINT "." TO THE LINE FOLLOWING THE LEFT OPERAND.
5. OTHERWISE, LEAVE "." UNCHANGED.

THE "↑" OPERATOR (UNARY)

THIS POSTFIX OPERATOR TAKES AS ITS OPERAND A LINE GROUP. THE EFFECT IS TO LINE-EDIT EACH LINE IN THE GROUP, OR UNTIL A CARRIAGE-RETURN-ESCAPE (CTRL-U) IS RECEIVED, IN WHICH CASE THE REMAINDER OF THE GROUP IS RETAINED UNCHANGED.

EXAMPLE: .↑ MEANS LINE-EDIT THE CURRENT LINE

THE "!" OPERATOR (UNARY)

THIS IS ANOTHER AMBIDEXTROUS OPERATOR, WHICH TAKES AS OPERAND A LINE ADDRESS. IF THE OPERATOR IS PREFIX, IT SETS A DYNAMIC BREAKPOINT ON THE LINE, PROVIDED THAT THE LINE IS ELIGIBLE. (I.E. IT IS A STATEMENT LINE, WHICH DOES NOT ALREADY HAVE A DYNAMIC BREAKPOINT SET.) THIS IS EXACTLY EQUIVALENT TO USING THE NORMAL EDITING COMMANDS TO INSERT "BREAK;" AT THE START OF THE LINE, EXCEPT THAT:

A) IT IS EASIER TO TYPE
B) IT TAKES MUCH LESS CPU TIME
C) IT ALLOWS RESUMPTION OF EXECUTION, EVEN IF THE FUNCTION CONTAINING THE LINE WAS ALREADY ACTIVE.

IF "!" IS USED AS A POSTFIX OPERATOR, IT CLEARS A DYNAMIC BREAKPOINT FROM A STATEMENT LINE ON WHICH ONE WAS ALREADY SET.