TECHNICAL MANUAL NO 10

AID - ALGEBRAIC INTERPRETIVE DIALOGUE

Sarah Barry

UNIVERSITY OF QUEENSLAND
COMPUTER CENTRE

UNIVERSITY OF QUEENSLAND

COMPUTER CENTRE

TECHNICAL MANUAL NO 10

AID - ALGEBRAIC INTERPRETIVE DIALOGUE

Sarah Barry

MNT-10
7Dec70

This manual has been authorized by the Director of the Computer Centre.

ABSTRACT

This manual has been adapted from the PDP-10 AID manual. AID itself is an adaptation of JOSS, a computing service program developed by the RAND corporation of America.

MANUAL STATUS

First Edition 7 December 1970

CONTENTS

Appendix Page

1. INTRODUCTION TO AID

AID is available on the University of Queensland PDP-10 system and provides each user with a personal computing service, interacting with the user and responding to commands expressed in a simple language via the user's Teletype. AID is easy and convenient to use in solving both simple and complex numerical problems.

AID allows the user to create external files for storage of subroutines and data for subsequent recall and use. These files are stored on disk.

AID runs in approximately 11K of core memory (with 1K of user data area) and expands to 14K of core (with 4K of user data area) as required.

1.1 INITIATING AID

To initiate AID, the user must first log into the operating system. This is accomplished simply by typing ↑C (hold down the CTRL key while striking C) and then logging in by using the LOGIN command (see the System User's Guide MNT-8).

When access to the operating system is gained, and the system has responded with a period (.), the user types

.AID<cr>

When AID is loaded into core, it responds with the message

AID (revision date) AT YOUR SERVICE ...
*

The asterisk (*) indicates that AID is ready to accept a command from the user.

1.2 TERMINATING AID

AID is terminated and control is returned to the operating system by typing

*↑C<cr> (Hold down the CTRL key and strike C)

1.3 AID LANGUAGE

Appendix B contains all AID commands and functions. Each command occupies a single line and is terminated by a return. This is echoed by the system as a carriage return and a line feed. A period at the end of a command is optional. A command can be entered as a *direct command* (to be executed immediately) or as an *indirect command* (to be stored for later execution). *Variables* in commands are represented by single alphabetic letters, A through Z, called *identifiers*. Entire *routines* can be stored as a series of indirect commands to be executed in a specific order. An *expression* is defined as one number or identifier (or a combination of numbers and/or identifiers and/or AID functions) that is reducible to a number when AID is called upon to use it. The *standard mathematical operators* can be expressed in AID as follows:

! !	absolute value	(equivalent to the mathematical symbol $\|\|$)
[ ]	brackets	
( )	parentheses	(brackets and parentheses can be used interchangeably in pairs)
+	addition	
-	subtraction	
*	multiplication	
/	division	
↑	exponentiation	($X^3 \equiv$ X↑3, the up arrow for exponentiation is shift N on Models 33 and 35 Teletype)

The order of precedence for these operations is conventional:

(a) ! !, [ ] and ( ) from the innermost pair to the outmost pair

(b) ↑ (exponentiation)

(c) * (multiplication) and / (division) from left to right within each term

(d) + (addition) and - (subtraction).

<u>examples</u>:

(i) A/3*C $= \left(\frac{A}{3}\right)C$ <u>not</u> $\frac{A}{3C}$ (left-to-right rule)

(ii) X/Y↑3 $= \frac{X}{Y^3}$ <u>not</u> $\left(\frac{X}{Y}\right)^3$ (order of precedence)

Boolean expressions composed of arithmetic statements using the operators

= (equal to)

($\neq$ not equal to)

<= ($\leqslant$ less than or equal to)

>= ($\geqslant$ greater than or equal to)

< (less than)

> (greater than)

and the negation

not

and connected in turn by logical operators

and

or (inclusive)

are handled by AID.

1.3.1 Rules of Form

(a) Only one step (command) may be typed per line and only one line may be used for each step.

(b) Each step begins with a verb and terminates with a return. A period at the end of a step is optional.

(c) Words, variables (identifiers), and numerals can neither abut each other nor contain embedded spaces; spaces cannot appear between an identifier (when it appears in an array, a formula, or a function) and its associated group operators and arguments. Otherwise, spaces can be used freely.

examples:

	Step Number	Verb	Arguments	Modifiers
(i)	1.23	TYPE	A, A↑2, A↑3	IN FORM 1 IF A>Ø.
(ii)	1.4	DO	PART 2	FOR C=5(1Ø)1ØØ.

1.3.2 Arithmetic Accuracy and Notations

All results are rounded to the nine most significant decimal digits.

All results with a value of less than 10^6 and equal to or greater than .001 are typed by AID in *fixed point* notation.

examples:

```
(i)        *TYPE 1/3+2<cr>
                1/3+2 =          2.33333333

(ii)       *TYPE 1ØØ**3<cr>
                100↑ 3 =         1*1Ø↑6

(iii)      *TYPE 1/4*1<cr>
                1/4*1 =           .25

(iv)       *TYPE COS(2.5)<cr>
             COS(2.5) =          -.8Ø1143616
```

All other results are typed in *scientific* notation.

examples:

```
(i)        *TYPE 365*24*6Ø*6Ø <cr>
           365*24*6Ø*6Ø =        3.1536*1Ø↑7         (Read as 3.1536 times
                                                      10 raised to the 7th
                                                      power)

(ii)       *TYPE (.ØØØ5)*  (17)*(.Ø1) <cr>
           (.ØØØ5)*  (17)*(.Ø1) = 8.5*1Ø↑(-5)        (Read as 8.5 times 10
                                                      raised to the minus
                                                      5th power)
```

1.4 USE OF TELETYPES

A Teletype is the link between the user and AID. The PDP-10 system is equipped with model 33 and model 35 Teletypes. Models 33 and 35 have upper case letters only, typed without use of the SHIFT key. Some of special characters occupy what are normally the upper case positions on the letter keys.

Appendix C lists the AID character set, the corresponding standard mathematical symbols, and the method used to obtain each character on the Teletype.

1.5 CORRECTION OF TYPING ERRORS

If the user should make an error while typing a command to AID, he can correct it by one of two methods:

(a) he can strike the RUBOUT key once for each character to be erased and then type the correct data, or

(b) he can type ↑U (press CTRL key while typing U) to delete the entire line, and type the line again.

examples:

(i)	*TYPE VEC\CEV\"VECTOR CALCULATION"<cr>	User omitted the quotation mark before the V; he erases C, E, and V by striking the RUBOUT key three times (deleted characters are printed between \), types the missing quotation mark, and continues.
(ii)	*TYPE"VECTOR CALCULATION↑U TYPE "VECTOR CALCULATION"<cr>	User realizes that he has omitted a space between the TYPE and the quotation mark; he decides to delete the line and retype it.

2. STEPS AND PARTS

A user requests AID functions by typing single-line commands called *steps*. The user can enter a step whenever AID responds with an asterisk (*) typeout. Each step is terminated by a return, <cr>.

Steps can be entered in two ways:

(a) as *direct* steps, or

(b) as *indirect* steps.

2.1 DIRECT STEPS

A direct step is interpreted and executed by AID immediately (following the terminating return typed by the user).

example:

```
*TYPE 2+2<cr>
         2+2 =        4          User types direct step.
*                                AID responds immediately
                                 by interpreting and
                                 executing the step.
```

Direct steps are performed only once each time they are typed, and must be retyped each time the user desires to execute them.

Should the user type a direct command incorrectly, then AID, when it attempts to interpret it, will respond with the message

```
EH?
*
```

2.2 INDIRECT STEPS

An indirect step is entered by preceding the step with a numeric label containing both an integer and a decimal portion (for example 1.1, 2.53). By preceding a step with a numeric label, the user signals to AID that the step is not to be executed immediately, but is to be stored for later execution as part of a routine. AID files away labelled steps in sequence according to the numeric value of the label or *step number*. Thus, a step number can be used to indicate that a step is to be inserted into, deleted from, or substituted in a series of previously entered indirect steps. Step numbers may contain a maximum of nine significant digits.

example:

*1.1 TYPE "X VALUES"<cr>	User types in a 3-step routine for later execution
*1.2 TYPE X<cr>	
*1.3 TYPE X*2, X/2, X↑2<cr>	
*1.15 SET X=3<cr>	User *inserts* a step between steps 1.1 and 1.2 by assigning it a number which falls between these two step numbers.
*1.3 TYPE X*2, X/2, X↑2, X↑3<cr>	User changes step 1.3 by *substituting* a new step having the same step number.
*DELETE STEP 1.2<cr>	User *deletes* step 1.2
*	

When indirect commands are entered, AID merely checks the validity of the step number; the validity of the command is not checked until it is called upon for execution.

2.3 PARTS

Steps are organized into *parts* according to the integer portion of their step numbers. All steps with step numbers containing the same value in their integer portion belong to the same *part*. Thus, all of the steps in the previous example can be referred to as PART 1.

example:

*TYPE PART 1<cr>	User requests AID to type all steps in PART 1.

```
1.1 TYPE "X VALUES".
1.15 SET X=3.
1.3 TYPE X*2, X/2, X↑2, X↑3.
```

*DO PART 1<cr>	User requests AID to interpret and execute (i.e. DO) all steps in PART 1.

```
X VALUES
          X*2 =       6
          X/2 =       1.5
          X↑2 =       9
          X↑3 =      27
*
```

Steps and parts are units that may be entered, changed, deleted, typed out, executed, or filed in (and later recalled from) a file stored on disk. In addition, they are available in core storage as stored routines for repetitive execution.

examples:

(i) TYPE STEP 1.1

(ii) TYPE PART 1

(iii) DO STEP 2.3

(iv) DO PART 4

(v) FILE STEP 3.65 AS ITEM 4

(vi) FILE PART 3 AS ITEM 2

All steps or parts can be referred to collectively (except by DO).

examples:

(i) TYPE ALL STEPS

(ii) TYPE ALL PARTS

(iii) FILE ALL STEPS AS ITEM 8

(iv) FILE ALL PARTS AS ITEM 9

3. IDENTIFIERS

An identifier or variable is used in expressions to represent a variable quantity. In AID, identifiers are represented by single alphabetic characters to which arithmetic or logical values have been assigned. On Teletype models 33 and 35, 26 unique identifiers are available.

3.1 DEFINING AN IDENTIFIER BY A VALUE

A *fixed value* may be assigned to an identifier by typing

SET identifier = value

When this command is typed as a direct command, the verb (SET) may be omitted

identifier = value

In a SET command, the single-character identifier on the left of the equals sign (=) is not a number, but an identifier being defined (or redefined). The value or expression on the right of the equals sign is a numeric value (or truth value) and must always, if a numeric expression, be immediately reducible to a number.

examples:

(i)	X=1Ø	
(ii)	SET X=3.5	
(iii)	Y=COS(25)+2	COS is a standard function provided by AID (see Chapter 4).
(iv)	SET A=SQRT(2Ø)+5	SQRT is also a standard AID function.
(v)	M=FALSE	M is set equal to the value FALSE.

The SET command is a convenient way to shorten a lengthy expression by using identifiers to represent its parts.

example:

$$\frac{\left(5 + \frac{34}{73}\right)^2 + \frac{42 - \sqrt{50}}{19}}{\left(5 + \frac{34}{73}\right)}$$

This expression can be simplified and solved as follows.

```
*A=5+34/73<cr>
*B=[42-SQRT(5Ø)]/19<cr>
*TYPE (A↑2+B)/A<cr>
   (A↑2+B)/A =         5.8Ø2Ø9589
*
```

Common algebraic functions (e.g. SQRT, COS, SIN, LOG) are provided in AID for use in expressions (see section 4.2).

example:

Define the value of pi (π)

```
*P=ARG(-1,Ø)<cr>
```

ARG is the AID function of a rectangular coordinate point (see Table 4-2).

```
*TYPE P<cr>
               P =      3.14159265
```

Calculate the area of a circle having a radius of 36.

```
*TYPE P*36↑2 <cr>
          P*36↑2 =    4Ø71.5Ø4Ø7
*
```

An identifier may also be set to a value, to be typed in by the user prior to execution of the associated routine. This is accomplished by using the DEMAND command, which may be used indirectly only. Execution of a DEMAND command causes a typeout of the specified variable, which is followed by the value to be used, typed by the user.

example:

```
        *1.1 DEMAND X<cr>
        *1.2 DEMAND Y<cr>
        *1.3 TYPE X*Y, (X↑2)*(Y↑2)<cr>
        *DO PART 1<cr>

                    X = *4<cr>                AID requests value for X.
                                              User responds by typing in 4.

                    Y = *6<cr>                AID requests value for Y.
                                              User responds by typing in 6.

                  X*Y =        24
         (X↑2)*(Y↑2) =        576
        *
```

An identifier may be set to a range of values by the 'DO...FOR identifier = first-value (increment) last-value' command (see section 5.5). When this form of the DO command is used, the series of steps is executed repetitively for each requested value, beginning with 'first-value' and incrementing it by 'increment' following each repetition until 'last-value' is reached. The range given for a variable can be greatly expanded beyond this simple format (see section 4.5). For example

DO PART 1 FOR X = 1,2,3(2)25(I)2↑T(K)2ØØ,5ØØ.

In this example, PART 1 is performed for X = 1, 2, 3, then in increments of Z up through 25, then in increments of I up through the value of 2^T, then in increments of K up through 200, and finally for X = 500.

example:

```
*1.1 TYPE X, X↑2, X↑3<cr>
*DO PART 1 FOR X=2(2)1Ø<cr>
```

Directs AID to perform step 1.1 for values of X, beginning with a value of 2 and incrementing this value by 2 until 10 is reached in a series of repetitive executions.

```
      X =      2
    X↑2 =      4
    X↑3 =      8
      X =      4
    X↑2 =     16
    X↑3 =     64
      X =      6
    X↑2 =     36
    X↑3 =    216
      X =      8
    X↑2 =     64
    X↑3 =    512
      X =     1Ø
    X↑2 =    1ØØ
    X↑3 =   1ØØØ
*
```

AID types out results.

3.2 DEFINING AN IDENTIFIER BY A FORMULA

3.2.1 Arithmetic Formulas

AID may be told how to calculate the value of an identifier rather than associating the identifier with a fixed value. This is done with the LET command. The use of LET causes the identifier on the left of the equals sign to be set to the *formula* (not necessarily a numeric value) on the right of the equals sign (=).

LET identifier = formula

example:

```
*LET D=SQRT(A)+B+C<cr>
*TYPE FORMULA D<cr>
            D:   SQRT(A)+B+C
*
```

Note that AID associates a formula, not a numeric value, with the identifier D.

In the above example, the formula for D is an expression reducible to a number, but this value is not calculated until D is called for. However, before D can be calculated, the user must supply values for all variables in the formula associated with D.

```
*TYPE D<cr>
ERROR IN FORMULA D:  A = ???      User has assigned no value
                                  to A.
*A=9<cr>
*B=5<cr>
*C=8<cr>
*TYPE D<cr>
           D =       16
*
```

3.2.2 Boolean Expressions

A second use of the LET command is to define an identifier as being equivalent to the value (true or false) of a *proposition*, i.e. a Boolean expression composed of arithmetic and logical statements using common relational operators (e.g. = >, <), the logical negation NOT and logical operators AND, OR.

```
LET identifier = proposition
```

example:

```
*SET A=TRUE<cr>
*B=FALSE<cr>
*LET C=A AND B<cr>
*TYPE C<cr>
           C =      FALSE
*
```

Propositions are discussed in detail in section 4.4.

3.2.3 User Functions

AID provides many of the common algebraic and geometric functions (SQRT, square root; COS, cosine; LOG, logarithm; etc.). AID *functions* are specified in expressions by using the appropriate function mnemonic (SQRT for square root).

A third use of the LET command is to equate an identifier to a *user-defined function*. Once defined, a user function can be used the same as an AID function.

example:

*LET A(B,C) = (B↑2) + (2*B*C) + (C↑2)<cr>	Defines the user function, A.
*TYPE A(4,1Ø)<cr>	
A(4,1Ø) = 196	
*	

Note that in the function A(B,C), B and C are *dummy arguments* and do not conflict with *variables* of the same letter outside of the formula (B and C may be used as identifiers elsewhere).

Both AID functions and user-defined functions are discussed further in Chapter 4.

3.3 IDENTIFIER REFERENCES

In addition to an identifier in a formula referring to its associated value or formula, it can also be used to delete, type, or file that value or formula.

examples:

(i)	DELETE A	Delete A and its associated value.
(ii)	DELETE FORMULA B	Delete B and its associated value.
(iii)	TYPE C	Type the value of C.
(iv)	TYPE FORMULA D	Type the formula associated with D.
(v)	FILE E AS ITEM 1	Store E and its associated value on the currently open file as item 1 (see section 5.7 on FILE).

All current identifiers and their associated values or formulas may be referred to collectively.

examples:

(i)	TYPE ALL VALUES
(ii)	TYPE ALL FORMULAS
(iii)	DELETE ALL VALUES
(iv)	DELETE ALL FORMULAS
(v)	FILE ALL VALUES AS ITEM 3
(vi)	FILE ALL FORMULAS AS ITEM 4

3.4 INDEXED IDENTIFIERS

Values may be organized into vectors and arrays by using indexed letters for identifiers. The letters may then be used to refer to the arrays. Identifiers defined by formulas may not be indexed. The index or subscript is enclosed in parentheses immediately following the identifier.

example:

```
*X(1) = 12<cr>
*X(2) =  4<cr>
*X(3) =  6<cr>
*TYPE X(1), X(2), X(3), X(1)*X(2)*X(3)<cr>
        X(1) =        12
        X(2) =         4
        X(3) =         6
X(1)*X(2)*X(3) =     288
*TYPE X<cr>
        X(1) =        12
        X(2) =         4
        X(3) =         6
*
```

X refers to all indexed Xs; thus, a nonindexed identifier cannot coexist with the same identifier indexed.

Multiple subscripts may be specified for an identifier to create a multidimensional array. An identifier may be indexed by one to ten subscripts, and each subscript may have an integer value in the range -250 through +250.

examples:

(i) X(1) = 6

(ii) A(1,2) = 1Ø

(iii) C(1ØØ,5Ø,67) = 13Ø

An individual identifier may be used in only one way at any one time and redefinition deletes any previous definitions. Thus, the definition of an identifier with n dimensions deletes all definitions of the same identifier having other than n dimensions.

example:

```
*X=5<cr>
```

The identifier X (unindexed, of 0 dimension) is defined as equal to 5.

```
*TYPE X<cr>
                X =          5
```

```
*X(1)=1Ø<cr>
*X(2)=2Ø<cr>
*TYPE X<cr>
             X(1) =         1Ø
             X(2) =         2Ø
```

The identifier X is now redefined with one dimension (subscript); the unindexed X is deleted.

```
*X(1,1)=33<cr>
```

The identifier X is now redefined as describing a two-dimensional array; Xs having other dimensions are deleted.

```
*TYPE X<cr>
           X(1,1) =         33
*X(1,2)=44<cr>
*X(2,1)=55<cr>
```

Additional X values having the same number of subscripts as the previously defined X are entered; no deletions occur.

```
*TYPE X<cr>
           X(1,1) =         33
           X(1,2) =         44
           X(2,1) =         55
```

Undefined elements of the X array in this example can be set to a value of zero by the use of the command LET X BE SPARSE

```
*TYPE X(2,2)<cr>
X(2,2) = ???
*LET X BE SPARSE<cr>
*TYPE X(2,2)<cr>
           X(2,2) =          Ø
*TYPE X<cr>
           X(1,1) =         33
           X(1,2) =         44
           X(2,1) =         55
X IS SPARSE
*
```

4. ARITHMETIC OPERATORS, FUNCTIONS, PROPOSITIONS, AND ITERATIONS

4.1 ARITHMETIC OPERATORS

As discussed in section 1.3 all standard arithmetic operators can be expressed in AID. These are presented in Table 4-1, in their order of precedence.

Standard Designation	AID Symbology	Meaning
$\|X\|$	!X!	Absolute value of X
[ ]	[ ]	First level grouping
( )	( )	Second level grouping
X^e	X↑E	The value 'X' raised to the power of 'e'
A.B, (A)(B), or A×B	A*B	Multiply A times B
A/B or $\frac{A}{B}$	A/B	Divide A by B
A + B	A + B	Add A to B
A - B	A - B	Subtract B from A

Table 4-1 AID Arithmetic Operators

Note that within nested pairs of parentheses the order of evaluation is from the innermost pair outward.

examples:

(i)

```
*X=-5<cr>
*Y=+2<cr>
*TYPE X+Y<cr>
          X+Y =        -3
*TYPE !X+Y!<cr>
        !X+Y! =         3
*TYPE X+Y↑2-15<cr>
     X+Y↑2-15 =       -16
*TYPE (X+Y)↑2-15<cr>
   (X+Y)↑2-15 =        -6
*TYPE (X+Y)↑(2-15)<cr>
(X+Y)↑(2-15) =         -6.27225472*1Ø↑(-7)
*
```

(ii)

```
*X=SQRT(16) + SQRT(9)<cr>
*Y=SQRT[16 + SQRT(9)]<cr>
*Z=SQRT[SQRT(16 + 9)]<cr>
*TYPE X<cr>
           X =          7
*TYPE Y<cr>
           Y =          4.35889894
*TYPE Z<cr>
           Z =          2.236Ø6798
*
```

(iii) Computing simple interest

r = rate of interest per year (in %)
t = time (in years)
p = principal

$$i = \frac{(p)(r)(t)}{100}$$

```
*LET I = (P*R*T)/1ØØ<cr>
*P=1ØØØ<cr>
*R=6<cr>
*T=3<cr>
*TYPE I<cr>
           I =        18Ø
*
```

(iv) Computing total accumulated principal and compound interest

a = accumulated principal and interest, compounded annually.
r, t, and p are the same as in example (iii)
$a = p\ (1 + r/100)^t$

```
*LET A=P*(1 + R/1ØØ)↑T<cr>
*P=1ØØØ<cr>
*R=6<cr>
*T=3<cr>
*TYPE A<cr>
              A =       1191.Ø16
*
```

(v) Formula for a catenary curve

$$y = \frac{a}{2}\left(e^{\frac{x}{a}} + e^{\frac{-x}{a}}\right)$$

where a is a constant, and
e is Euler's number

```
*LET M = X/A<cr>
*LET N = Ø-M<cr>                              (optional)
*LET E = 2.71828183<cr>
*LET Y = (A/2)*[(E↑M) + (E↑(Ø-M))]<cr>

                 or

*LET Y = (A/2)*[(E↑M) + (E↑N)]<cr>

*1.1 TYPE Y<cr>
*A=-3<cr>
*DO PART 1 FOR X=1(1)5<cr>                    Do Part 1 for Y with
              Y =       -3.1682156            values of X beginning
              Y =       -3.69172674           with 1 and incremented
              Y =       -4.62924191           by 1 until 5 is
              Y =       -6.Ø8589753           reached.
              Y =       -8.225Ø4851
*
```

4.2 AID FUNCTIONS

Many common algebraic and geometric functions are provided by AID for use in expressions. Two of the most commonly used functions are

SQRT	Square root
LOG	Natural logarithm

examples:

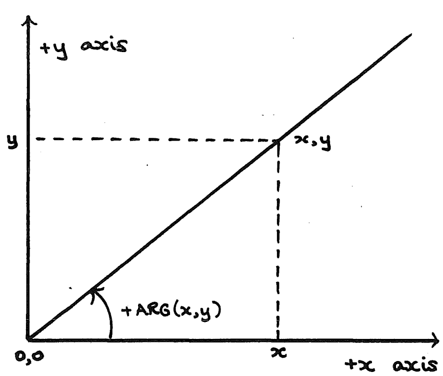
(i) SQRT(1Ø)

(ii) LOG(X*Y)

Note that the argument for a function is enclosed in parentheses and immediately follows the function mnemonic.

Table 4-2 lists AID functions in alphabetic order. The symbols x and y represent any expression reducible to a number and are the arguments of the function. The variable i is a dummy variable and does not affect any real identifier denoted by the same alphabetic character.

Table 4-2 AID Functions

Function	Description
ARG (x,y) 	The ARGUMENT function takes two arguments (x,y) and computes the angle between the +x axis of the x, y plane and the line joining point 0,0 and point x,y. The result is in radians ARG(x,y) The value of arg (0,0) is 0. The range is 0 through 2π, or $-\pi$ through π. 2π radians are equivalent to 360 degrees.
COS (x)	The COSINE function requires one argument, assumed to be in radians. $\lvert x \rvert$ must be < 100.

Function	Dex
DP(x)	The DIGIT PART function. DP(13456.5432) = 1.34565432
EXP(x)	The EXPONENTIAL function: e^x, where e is Euler's number (2.718281828). The argument (x) must fulfil the requirement that $e^x<10^{100}$ (i.e. x must be less than 230.2585). If $e^x<10^{-99}$, the result is 0.
FIRST(i=range...:i proposition)	The FIRST function requires two arguments: (a) an iterative clause (see section 4.5) and (b) a proposition containing i as an index. The result is the first value of index i to satisfy the proposition.
FP(x)	FRACTION PART function. FP(13456.5432) = .5432
IP(x)	INTEGER PART function. IP(13456.5432) = 13456
LOG(x)	NATURAL LOGARITHM function. The argument (x) must be greater than zero.

Function	Description
MAX(i=range...:...i expression...)	*The MAXIMUM function requires two arguments: (a) an iterative clause (see section 4.5), and (b) an expression containing a function of i. The expression is computed iteratively for each value of i, and the result (largest value) is typed out.
MIN(i=range...:...i expression...)	*The MINIMUM function requires two arguments: (a) an iterative clause (see section 4.5), and (b) an expression containing a function of i. The expression is computed iteratively for each value of i, and the result (smallest value) is typed out.
PROD(i=range...:...i expression...)	*The PRODUCT function requires the same two types of arguments as the MAXIMUM, MINIMUM and SUM functions. The expression is computed iteratively for each value of i, and the result (product of all the iterations) is typed out.
SGN(x)	The SIGNUM function. The value of a signum function of an argument greater than zero is +1, of an argument equal to zero is 0, of an argument less than zero is -1.

Function	Description
SIN(x)	The SINE function requires one argument, assumed to be in radians. $\lvert x \rvert$ must be < 100.
SQRT(x)	The SQUARE ROOT function. The argument (x) must be equal to or greater than zero.
SUM(i=range...:...i expression...)	*The SUM function requires the same two types of arguments as the MAXIMUM, MINIMUM, and PRODUCT functions. The expression is computed iteratively for each value of i, and the result (sum of all iterations) is typed out.
TV(proposition)	The TRUTH VALUE function requires one argument, a proposition, and converts this argument into a numeric value: 1, if the proposition is true; 0, if the proposition is false.
XP(x)	The EXPONENT PART function. XP(13456.5432) = 4 i.e. 13456.5432 = 1.34565432*1Ø↑4

*
The iterative clause and i function can, in all of these cases, be replaced by a simple series of values for i.

example:

MAX(5,-4.3,y,x↑2)

See section 4.5.

examples:

```
*A=1Ø<cr>
*B=12<cr>
*C=-2.5<cr>
*D=1ØØ<cr>
*E=1.325<cr>
*F=1Ø.435<cr>
*I=25<cr>
*
```

(i)

```
*LET Z = SUM(I=Ø(1Ø)1ØØ:I*2)<cr>
*TYPE Z<cr>
            Z =      11ØØ
*
```

The I in (i) is a dummy variable and in no way relates to the I in (ii). The latter is an identifier and refers to the variable defined above.

(ii)

```
*TYPE SUM(A, B, C, D, E, F, I, Z)<cr>
SUM(A, B, C, D, E, F, I, Z) = 1256.26
*
```

The sum of all these values.

(iii)

```
*LET Z = PROD(I=1(1)5:I↑2)<cr>
*TYPE Z<cr>
            Z =     144ØØ
*
```

The product of each value of the expression.

(iv)

```
*LET Z = MAX(I=-15(1)15:(I↑2)-(-5*I))<cr>
*TYPE Z<cr>
            Z =       3ØØ
*
```

The maximum value of the expression over all the range of values.

(v)

```
*LET Z = MIN(I=-15(1)15:(I↑2)-(-5*I))<cr>
*TYPE Z<cr>
            Z =        -6
*
```

The minimum value of the expression over all the range of values.

(vi)

```
*TYPE MIN(A,B,C,D,E,F,I)<cr>
MIN(A,B,C,D,E,F,I) = -2.5
*
```

The minimum value of each of these values.

(vii)

```
*TYPE ARG(-1,Ø)<cr>
   ARG(-1,Ø) =         3.14159265
*
```

The angle in radians between the x axis and the point (-1,Ø).

(viii)

```
*TYPE ARG(C,A)<cr>
    ARG(C,A) =          1.81577499
*
```

The angle in radians between the x axis and the point (c,a).

(ix)

```
*TYPE SIN(1Ø)<cr>
    SIN(1Ø) =          -.544Ø21111
*
```

The sine of 10 radians.

(x)

```
*TYPE COS(A)<cr>
    COS(A) =           -.839Ø71529
*
```

The cosine of A where A is assumed to be in radians.

(xi)

```
*TYPE SIN((A*E)-I)<cr>
SIN((A*E)-I) =          .728664976
*
```

The sine of ((A*E)-I) where this is assumed to be in radians.

(xii)

```
*TYPE EXP(.346)<cr>
   EXP(.346) =          1.4134Ø262
*
```

The exponential of .346.

(xiii)

```
*TYPE DP(E+F)<cr>
     DP(E+F) =          1.176
*
```

The digit part of (E+F).

(xiv)

```
*TYPE FP(E+F)<cr>
     FP(E+F) =           .76
*
```

The fraction part of (E+F).

(xv)

```
*TYPE IP(E+F)<cr>
     IP(E+F) =        11
*
```

The integer part of (E+F).

(xvi)

```
*TYPE LOG(65Ø)<cr>
    LOG(65Ø) =          6.47697236
*
```

The natural logarithm of 650.

(xvii)

```
*TYPE LOG(E+F)<cr>
    LOG(E+F) =          2.4647Ø394
*
```

The natural logarithm of (E+F).

(xviii)

```
*TYPE SGN(D-(B↑2)+I)<cr>
SGN(D-(B↑2)+I) =      -1
*
```

The signum of (D-(B↑2)+I).

(xix)

```
*M=-5<cr>
*N=3<cr>
*TYPE TV((M>=N) OR (M=Ø) OR (M<Ø) AND (M>-4))<cr>
TV((M>=N) OR (M=Ø) OR (M<Ø) AND (M>-4)) = Ø
*
```

The truth value function.

(xx)

```
*TYPE SQRT(A+B+C+D+E)<cr>
SQRT(A+B+C+D+E) =     1Ø.992Ø426
*
```

The square root of the sum of these values.

(xxi)

```
*1.1 LET A(X) = X↑2-2Ø<cr>
*DO STEP 1.1 FOR X=1(1)3Ø<cr>
```

Set up a table (or array) of 30 items calculated according to the formula given in step 1.1.

```
*TYPE A(25)<cr>
      A(25) =       6Ø5
*TYPE FIRST(I=1(1)3Ø:A(I)=Ø)<cr>
FIRST(I=1(1)3Ø:A(I)=Ø) = ???
```

No such value found in table.

```
*TYPE FIRST(I=1(1)3Ø:A(I) 7ØØ)<cr>
FIRST(I=1(1)3Ø:A(I) 7ØØ) = 27
*TYPE A(27)<cr>
      A(27) =       7Ø9
*
```

4.3 USER-DEFINED FUNCTIONS

Functions not included in AID can easily be defined for repetitive use.

As discussed in section 3.2.3 the LET command is used to equate an identifier to some user-defined function. Following this function identifier, up to ten dummy arguments (enclosed as a group in parentheses) can be specified; these are replaced by actual arguments when the function is to be used. Dummy arguments are also represented by single alphabetic characters, but the use of a letter as a dummy in no way affects the use of that same letter as an identifier. Following the dummy arguments, an equals sign and the expression representing the user function are typed.

f(a,b,c,...) = expression

where 'f' is the function identifier (any single alphabetic character)
'(a,b,c,...)' are dummy arguments (also single alphabetic characters)
'expression' is the arithmetic expression representing the user function.

Arguments supplied for functions can themselves be functional.

4.3.1 Examples of User-Defined Functions

example:

```
*LET A(B,C) = SQRT(B*C) + B↑2 + C↑2<cr>
```

Define the user function A, with two dummy arguments B and C, as being equivalent to the formula

SQRT(B*C)+B↑2+C↑2

```
*TYPE A(12Ø.555,32.Ø76)<cr>
A(12Ø.555,32.Ø76) = 15624.5624
```

Use the newly-defined function by specifying two actual arguments in place of the dummy arguments, B and C.

```
*TYPE A<cr>
      A(B,C):   SQRT(B*C) + B↑2 + C↑2
```

Note that A is equated to the formula, not a value, since A alone is not an expression.

```
*TYPE FORMULA A<cr>
      A(B,C):   SQRT(B*C) + B↑2 + C↑2
```

Same typeout.

```
*TYPE A(B,C)<cr>
B = ???
```

No values have been specified for the identifiers (not dummy arguments), B and C.

```
*B=(4↑6)/9<cr>
*C=5.23<cr>
*TYPE A(B,C)<cr>
      A(B,C) =    2Ø72Ø2.264
*
```

Many common functions can be defined as user functions, as shown below.

examples:

(i) Tangent function

```
*LET T(A) = SIN(A)/COS(A)<cr>
*TYPE T(1Ø)<cr>
       T(1Ø) =           .64836Ø828
*
```

(ii) Arc cosine function

```
*LET F(A) = ARG(A, SQRT(1-A↑2))<cr>
*TYPE F(.1Ø)<cr>
      F(.1Ø) =         1.47Ø62891
*
```

(iii) Arc cotangent function

```
*LET C(A) = ARG(A,1)<cr>
*TYPE C(1Ø)<cr>
        C(1Ø) =            .Ø996686522
*
```

(iv) Arc cosecant function

```
*LET S(A) = ARG(SQRT(1-1/A↑2), 1/A)<cr>
*TYPE S(1Ø)<cr>
        S(1Ø) =            .1ØØ167421
*
```

(v) Arc secant function

```
*LET K(A) = ARG(1/A,SQRT(1-1/A↑2))<cr>
*TYPE K(1Ø)<cr>
        K(1Ø) =            1.47Ø62891
*
```

(vi) Arc sine function

```
*LET N(A) = ARG(SQRT(1-A↑2),A)<cr>
*TYPE N(.1Ø)<cr>
        N(.1Ø) =           .1ØØ167421
*
```

(vii) Arc tangent function

```
*LET T(A) = ARG(1,A)<cr>
*TYPE T(1Ø)<cr>
        T(1Ø) =            1.47112767
*
```

(viii) Logarithm to base 10

```
*LET L(A) = LOG(A)/LOG(1Ø)<cr>
*TYPE L(25.38)<cr>
     L(25.38) =            1.4Ø449162
*
```

(ix) Derivative of a function of a variable

```
*LET D(A) = (F(A + .ØØ5)-F(A - .ØØ5))/.Ø1<cr>
*LET F(A) = 3*A↑3-4*A↑2+2*A+5<cr>
*TYPE D(4)<cr>
        D(4) =         114
*
```

4.4 PROPOSITIONS

As discussed in section 3.2.2, propositions are Boolean expressions composed of arithmetic or logical statements using the relational operators.

= (equal)
(not equal)
> (greater than)
< (less than)
>= (greater than or equal to)
<= (less than or equal to)

the negation

not

and the logical operators

and
or

A proposition has either of two possible values: *true* or *false*.

example:

```
*X=TRUE<cr>
*Y=FALSE<cr>
*LET Z=(X) AND (Y) OR (X) AND (100>SQRT(959))<cr>
*TYPE Z<cr>
           Z =      TRUE
*
```

The order of execution within a proposition is:

(a) evaluation of expressions
(b) ( ) Within nested pairs of parentheses, the order of evaluation is from the innermost pair outward.
(c) relational operations
(d) NOT
(e) AND
(f) OR

A series of relational operations is assumed to be an AND chain if no logical operator intervenes.

A=B>C<D is equivalent to A=B AND B>C AND C<D

The truth value (TV) function (see section 4.2) converts the value of a proposition to a numeric value (true = 1, false = 0) and allows it to be used as an expression, since it is then reducible to a numeric value.

example:

```
*SET X=TRUE<cr>
*LET Y= (X) AND (SQRT(1ØØ)>SQRT(3Ø*5-2Ø))<cr>
*TYPE TV(Y)<cr>
       TV(Y) =          Ø
*TYPE 24 + TV(X)<cr>
 24 + TV(X) =          25
*
```

4.4.1 Conditional Expressions

A conditional expression allows an expression (e.g. a variable) to have different values depending upon which of a number of conditions is true. It is composed of a series of clauses separated by semicolons, with each clause made up of a proposition followed by a colon followed by an expression. The entire conditional expression must be enclosed by parentheses.

```
(proposition:expression;  proposition:expression;  ...)
```

example:

Express the function:

If x>0, C(x) = x↑2;
if x=0, C(x) = 0;
if x<0, C(x) = x.

```
*LET C(X) = (X>Ø:X↑2;X=Ø:Ø;X<Ø:X)<cr>
*TYPE C(5), C(-1Ø), C(Ø), C(1Ø)<cr>
        C(5) =        25
      C(-1Ø) =       -1Ø
        C(Ø) =         Ø
       C(1Ø) =       1ØØ
*
```

If the last expression is to be true for all cases that do not satisfy any of the stated conditions, the expression can be typed without a preceding proposition. For example, in the case above, the user could have typed:

```
LET C(X) = (X>Ø:X↑2;X=Ø:Ø;X)
```

Note that every possible combination of the variable must be provided for, either by explicitly stating a conditional expression and a proposition for it, or by simply specifying a terminating expression to be executed for all cases that do not satisfy any of the explicitly stated propositions. If this provision is not made, and an unprovided-for condition occurs, AID responds with the message.

```
ERROR IN FORMULA X
```

A conditional expression can be used to perform a table lookup for all items whose values satisfy one or more conditions.

example:

```
*1.1 SET A(X) = X↑2 + X*.5 - 5*X<cr>
*DO STEP 1.1 FOR X = 1(1)35<cr>
*TYPE A(2Ø)<cr>
        A(2Ø) =       31Ø
*TYPE A(3)<cr>
         A(3) =        -4.5
```

Generates a 35-item table

```
*LET F(X) = (X<Ø:X;X>7ØØ:X;FP(X)>Ø AND X>3ØØ:X; ← )<cr>
```

Find all values that are
(a) less than zero
(b) greater than 700, or
(c) greater than 300 and have a fractional part that is nonzero.

If X is none of these, perform a line feed, carriage return (indicated by the ← symbol).

```
*1.1 TYPE F(A(I))<cr>
*DO STEP 1.1 FOR I = 1(5)35<cr>
     F(A(I)) =        -3.5



     F(A(I)) =       346.5

     F(A(I)) =       821.5
     F(A(I)) =      1Ø67.5
```

Test every fifth item in the table.

Values in tested items that do not satisfy any of the three propositions result in line feed/ carriage return (because of the terminating ← symbol in the conditional expression).

```
*LET E(X) = (FP(X/2)=Ø:X; ← )<cr>
*1.1 TYPE E(A(I))<cr>
```

Find all even-numbered values in the table.

*DO STEP 1.1 FOR I = 1(1)35<cr> Test every item in the table.

E(A(I)) = -2

E(A(I)) = 28

E(A(I)) = 9Ø

E(A(I)) = 184

E(A(I)) = 31Ø

E(A(I)) = 468

E(A(I)) = 658

E(A(I)) = 88Ø

4.5 ITERATIVE CLAUSES

Iterative clauses are used with the DO command and with the functions FIRST, MAX, MIN, PROD, and SUM. In both cases, the iterative clause specifies a range of values to be acted upon by the command or function.

4.5.1 Series of Values

One format of an iterative clause lists the individual values that make up the range:

$n, n_1, n_2, n_3, \ldots$

for example,

DO PART m FOR x = range

DO PART 1 FOR A = 1, M, 1ØØ, 5Ø, -25, x↑3

PART 1 will be executed for each of the individual values of A.

TYPE SUM(x=range)

TYPE SUM(A = -4.6,M*N, 24Ø.5, C)

The SUM function is performed on all values listed and the result (the SUM of all values) typed out.

4.5.2 Incrementation

The range of values for a variable can also be expressed as a first value, an incremental value, and an ending value. As a result, the variable values range from the first value upward in steps of the specified increment until the ending value is reached. The ending value is always taken as the last value in the range, even though the incremental steps may not hit it exactly.

The general form of an incremental iterative clause is:

x = first-value(increment)ending-value

For example:

DO STEP m.n FOR x = range

DO STEP 2.3 FOR A = 1(2)12

STEP 2.3 will be executed for each individual value of A:

1,3,5,7,9,11,12

TYPE SUM(x=range)

TYPE SUM(A = -5Ø(B)C)

The SUM of all values of A, as indicated by the range, is calculated. This range begins with -50 and continues in increments of B until C is reached.

4.5.3 Combinations

A range can be expressed as a combination of value series and increments.

X = A(B)C,D,E(F)G(H)I,J,K	The range of X values begins with A, continues in increments of B through C, then D, E, then in increments of F until G is reached, then in increments of H until I is reached, then J and K.

For example,

DO PART 3 FOR W = 2Ø(Y)5Ø(5Ø)5ØØ, 1ØØØ(1ØØ)Z

PART 3 will be performed for all values of W, beginning with 20, continuing in increments of Y through 50, then continuing in increments of 50 through 500, and from 1000 in increments of 100 through Z.

TYPE SUM(W = A(3Ø)B, C, 8ØØ(D)15ØØ)	The SUM function will be applied to all values of W, beginning with A and continuing in increments of 30 through B, then C, followed by 800 through 1500 in increments of D.

5. AID VERBS

This chapter contains a summary of AID verbs, their command formats, optional features, and examples of usage. Some of these verbs (e.g. TYPE, DO) have appeared frequently in examples in previous chapters; others (e.g. LET, SET) have already been described extensively and are included here only as a review.

Some verb descriptions include diagnostic messages which are associated with a specific command or group of commands. A complete list of diagnostic messages can be found in Appendix D.

5.1 CANCEL

5.1.1 Description

The CANCEL command cancels a currently stopped (interrupted) process, if the user does not desire to resume execution. An interrupted process is automatically cancelled whenever the user types a direct DO command to initiate another part or step. The CANCEL command does not however, delete any commands, formulas, variables, etc., associated with the interrupted process. It does release any core memory currently assigned to the interrupted execution.

CANCEL is the antithesis of the GO command.

The CANCEL command can be given directly only.

example:

```
*1.1 LET X = ...<cr>
 .
 .
 .
*2.1Ø TYPE ...<cr>
*DO PART 1<cr>
ERROR AT STEP 2.5:  ILLEGAL SET OF VALUES FOR ITERATION.
*CANCEL<cr>                        User does not desire to correct
                                   step and resume execution.
*
```

5.1.2 Parenthetical CANCEL

Typing

(CANCEL)

cancels any currently stopped process that was initiated by a parenthetical DO (see section 5.5.5).

5.1.3 Diagnostic messages

(a) DON'T GIVE THIS COMMAND INDIRECTLY

The CANCEL command can be given directly only (with no step number preceding it).

5.2 DELETE

5.2.1 Description

The DELETE command erases a step, part, form, value, or formula from core storage and frees that storage for some other use. This command should be used frequently to delete routines, tables, and other items which are no longer needed. By doing this, unnecessary waste of storage and possible storage overflow can be avoided.

DELETE a
    Delete identifier a and its associated value(s) from core storage. If identifier a is a subscripted variable, the entire a array is deleted.

DELETE a(b,...)
    Delete the particular array item, a(b,...), and its associated value from core storage.

DELETE STEP m.n
    Delete the step numbered m.n.

DELETE PART m
    Delete PART m (all steps having numbers whose integer portion is m).

DELETE FORMULA a
    Delete the formula associated with a.

DELETE FORM m
    Delete FORM m.   (See section 5.8 on FORM and section 5.20 on TYPE for an explanation of forms.)

```
           VALUES
           STEPS
DELETE ALL PARTS
           FORMULAS
           FORMS
```
    Delete all entries of the named type.

DELETE ALL
    Delete all entries.

Several individual DELETE commands can be combined into one. For example,

    DELETE X, FORM 3, FORMULA B, ALL PARTS.

example:

```
*1.1 TYPE "STEP A"<cr>              Type entries into core storage.
*1.2 TYPE A+B IN FORM 1<cr>
*1.3 TO STEP 2.1<cr>
*2.1 TYPE A↑2+B↑2<cr>
*2.2 TYPE "END"<cr>
*LET A = 1Ø;<cr>
*LET A = 1Ø+B<cr>
*B=25<cr>
*FORM 1:<cr>
*←←←←←.←←<cr>
*DO PART 1<cr>
STEP A                              Test routine.
   6Ø.ØØ
     A↑2+B↑2 =      185Ø
END
*DELETE B<cr>                       Delete identifier B and its
                                    associated value.
*DO PART 1<cr>                      Attempt to use B.
STEP A
ERROR AT STEP 1.2 (IN FORMULA A):  B = ???
*DELETE STEP 2.1<cr>                Delete STEP 2.1.
*DO STEP 2.1<cr>                    Attempt to execute it.
I CAN'T FIND THE REQUIRED STEP.
*DELETE A<cr>                       Delete formula A.
*TYPE A<cr>                         Attempt to type it.
A = ???
*DELETE ALL<cr>                     Delete remaining entries.
*TYPE ALL<cr>                       Test that all have been deleted.
*
```

5.3 DEMAND

5.3.1 Description

DEMAND causes AID to type out a request for a user-supplied value during execution of a routine. The DEMAND command can be given indirectly only.

DEMAND a

AID types out a request for the value of a.

```
*1.1 DEMAND A<cr>
*1.2 TYPE ...<cr>
 .
 .
 .
*DO PART 1<cr>
              A = *value-of-A<cr>
 .
 .
 .
```

DEMAND a(b,...)

AID types out request for the value of the subscripted variable a(b,...).

```
*1.1 DEMAND M(3,5,7)<cr>
*1.2 TYPE ...<cr>
 .
 .
 .
*DO PART 1<cr>
     M(3,5,7) =    *value-of-M(3,5,7)<cr>
 .
 .
 .
```

DEMAND a AS "any text"

AID types "any text" to request a value for a.

```
*1.1 DEMAND P AS "NUMBER OF SAMPLES WANTED"<cr>
*1.2 TYPE ...<cr>
 .
 .
 .
*DO PART 1 <cr>
NUMBER OF SAMPLES WANTED=    *value-of-P<cr>
 .
 .
 .
```

DEMAND a(b,...) AS "any text"
AID types "any text" to request a value for the subscripted variable a(b,...).

```
*1.1 DEMAND Y(3) AS "MAXIMUM SPEED"<cr>
*1.2 TYPE ...<cr>
 .
 .
 .
*DO PART 1<cr>
     MAXIMUM SPEED=          *value-of-Y(3)<cr>
 .
 .
 .
```

Depending upon the use of the variable specified, values requested by a DEMAND command can be entered in the form of

(a) a numeric expression (e.g. a number in fixed or floating point notation, or an identifier representing a numeric value),

(b) a formula, or

(c) a Boolean value (true or false).

Only one variable can be specified in each DEMAND command.

examples:

(i)

```
*X=25<cr>
*Y=5Ø.25<cr>
*Z=16.4<cr>
*1.1 TYPE "CONVERSION OF POUNDS TO KILOGRAMS"<cr>
*1.2 DEMAND A<cr>
*1.3 TYPE A, B IN FORM 1<cr>
*1.4 TO STEP 1.2<cr>
*LET B = .45359*A<cr>
*FORM 1:<cr>
*←←←←←.←←←← POUNDS = ←←←←←.←←←← KILOGRAMS<cr>
*DO PART 1<cr>
CONVERSION OF POUNDS TO KILOGRAMS
           A = *25.8<cr>
   25.8ØØØ POUNDS =    11.7Ø26 KILOGRAMS
           A = *1ØØ.543<cr>
  1ØØ.543Ø POUNDS =    45.6Ø53 KILOGRAMS
           A = *5567.98<cr>
 5567.98ØØ POUNDS =  2525.58Ø1 KILOGRAMS
           A = *<cr>     Return by user terminates iterations.
I'M AT STEP 1.2.
*1.2 DEMAND A AS "POUNDS"<cr>
*DO PART 1<cr>
CONVERSION OF POUNDS TO KILOGRAMS
           POUNDS = *25.8<cr>
   25.8ØØØ POUNDS =    11.7Ø26 KILOGRAMS
           POUNDS = *<cr>
I'M AT STEP 1.2.
*
```

```
(ii)        *LET A = M AND N<cr>
            *LET B = M OR N<cr>
            *1.1 DEMAND M<cr>
            *1.2 DEMAND N<cr>
            *1.3 TYPE TV(A) IN FORM 1<cr>
            *1.4 TYPE TV(B) IN FORM 2<cr>
            *1.5 TO STEP 1.1<cr>
            *FORM 1:<cr>
               LOGICAL AND:   ←<cr>
            *FORM 2:<cr>
               LOGICAL OR:    ←<cr>
            *DO PART 1<cr>
                      M = *TRUE<cr>
                      N = *FALSE<cr>
              LOGICAL AND:  Ø
              LOGICAL OR:   1
                      M = *FALSE<cr>
                      N = *FALSE<cr>
              LOGICAL AND:  Ø
              LOGICAL OR:   Ø
                      M = *NOT TRUE<cr>
                      N = *NOT FALSE<cr>
              LOGICAL AND:  Ø
              LOGICAL OR:   1
                      M = *<cr>          Return terminates iterations.
            I'M AT STEP 1.1.
            *
```

(iii)

```
*1.1 DO PART 2 FOR B = 1(1)3<cr>
*1.2 TYPE A<cr>
*2.1 DO PART 3 FOR C = 1(1)5<cr>
*3.1 DEMAND A(B,C)<cr>
*3.2 SET A(B,C) = SQRT(A(B,C))<cr>
*DO PART 1<cr>
      A(1,1) = *30<cr>
      A(1,2) = *65<cr>
      A(1,3) = *4<cr>
      A(1,4) = *50<cr>
      A(1,5) = *43.55677<cr>
      A(2,1) = *32<cr>
      A(2,2) = *1<cr>
      A(2,3) = *45.99<cr>
      A(2,4) = *29<cr>
      A(2,5) = *22.3333<cr>
      A(3,1) = *56.77<cr>
      A(3,2) = *66.7777<cr>
      A(3,3) = *99<cr>
      A(3,4) = *100<cr>
      A(3,5) = *1234.33<cr>
      A(1,1) =        5.47722558
      A(1,2) =        8.06225775
      A(1,3) =        2
      A(1,4) =        7.07106781
      A(1,5) =        6.5997553
      A(2,1) =        5.65685425
      A(2,2) =        1
      A(2,3) =        6.78159273
      A(2,4) =        5.38516481
      A(2,5) =        4.7258121
      A(3,1) =        7.53458692
      A(3,2) =        8.17176236
      A(3,3) =        9.94987437
      A(3,4) =       10
      A(3,5) =       35.1330329
*
```

5.4 DISCARD

5.4.1 Description

DISCARD deletes an item from the disk file currently in use.

DISCARD ITEM m (code)

Erase item m (where m can be in the range 1 through 25) from the currently open disk file and make the item available for some other use. Core storage is not affected in any way. (code) is optional for documentation purposes only and is ignored by AID; however, code, if used, cannot exceed five characters in length.

example:

```
*DISCARD ITEM 2Ø<cr>
DONE.
```

Item 20 of the disk file currently in use has been cleared successfully, as evidenced by the AID message, DONE. Item 20 can now be used for storing some other data, via the FILE command.

5.4.2 Diagnostic Messages

(a) I CAN'T FIND THE REQUIRED ITEM

The specified item cannot be found in the currently open file. Either the wrong file is open, or the item number is incorrect.

(b) ITEM NUMBER MUST BE POSITIVE INTEGER <= 25

An invalid item number was given.

(c) YOU HAVEN'T TOLD ME WHAT FILE TO USE

A DISCARD command was attempted before an external storage file was opened via a USE command.

5.5 DO

5.5.1 Description

The DO command executes an indirect step or part. DO is completed when either
(a) in a noniterative operation, the last step in the sequence has been completed, or
(b) in an iterative operation the last iteration has been completed.

(Steps are always executed according to the numerical sequence of their step numbers, regardless of the order in which the steps were originally entered).

If the DO command is a direct step, control returns to the user at the completion of the DO; if the DO command is indirect, control returns to the step following the DO. If the step or part being executed contains imbedded DO or TO commands, they are executed normally.

DO STEP m.n

Execute STEP m.n and return control as described above.

DO STEP m.n, p TIMES

Execute STEP m.n the number of times specified by integer p and return control as described above. Note that a comma must immediately follow the step number.

DO STEP m.n FOR x = range

Execute STEP m.n iteratively for each specified value of x as indicated by the range (see section 4.5). When the range is satisfied, control is returned as described above.

examples:

(i) DO STEP 1.2 FOR X = 1(1)5

Execute STEP 1.2 iteratively, beginning with an initial X value of 1 and incrementing X by 1 prior to each iteration until the maximum value of 5 is reached.

(ii) DO STEP 5.25 FOR A = -1Ø.25(.25)4.5Ø

Execute STEP 5.25 iteratively, beginning with an initial A value of -10.25 and incrementing A by the value .25 each time until the maximum value of 4.50 is reached.

(iii) DO STEP 1.3 FOR M = 1,-2.5,1ØØ,-43.666

Execute STEP 1.3 for each of the four specified M values.

(iv) DO STEP 1Ø.6 FOR P = 2ØØ,-3Ø.667,-2.3(.1)1.9,5.75

Execute STEP 10.6 for three values of P (200,-30.667, and 5.75) and for a range of values of P (-2.3 through 1.9, in increments of .1).

(v) DO STEP 1.3 FOR M = 1(4)26(5)5Ø(25)155

Perform STEP 1.3 iteratively for M = 1 to 26 in increments of 4, 26 to 50 in increments of 5, and 50 to 150 in increments of 25.

Thus, the values of M will be 1,5,9,13,17,21,25,26,31,36,41,46, 50,75,100,125,150, and 155 for the 18 iterations of step 1.3.

DO PART m

Execute PART m (all steps having the value m as the integer portion of their step number). All steps are executed in numeric sequence; any jump (via a DO or TO) to a step which is outside PART m is handled correctly. Control is returned as described above.

DO PART m, p TIMES

Execute PART m the number of times specified by integer p and return control as described above.

DO PART m FOR x = range

Execute PART m iteratively for each specified value of x in the same manner as described under "DO STEP m.n FOR x = range".

The FOR clause of a DO command is interpreted only once, at the point where the DO command is encountered; therefore, if a variable specified within the FOR clause is changed during execution of the DO-initiated routine, the change has no effect on the performance of the FOR clause. The number of iterations performed and the setting of the variable at the beginning of each iteration is the same as if no modification of the variable were performed by the routine.

example:

```
*1.1 DO PART 2 FOR X = 1(1)5<cr>
*2.1 TYPE X<cr>
*2.2 SET X = X+1ØØ<cr>
*2.3 TYPE X<cr>
*DO PART 1<cr>
             X =        1
             X =      1Ø1
             X =        2
             X =      1Ø2
             X =        3
             X =      1Ø3
             X =        4
             X =      1Ø4
             X =        5
             X =      1Ø5
*
```

Note that, when the FOR clause is used, the end-range value is hit exactly, For example, given the DO command

DO PART 1 FOR X = 1(3)14.5

iterations will be performed for X = 1, 4, 7, 10, 13, and 14.5.

5.5.2 IF Clause

The IF clause (see section 5.10) when appended to a DO command is also interpreted only once (when the DO command is encountered) and has no effect once execution of the DO has begun. Thus, even though the DO-initiated routine might perform some action which would make the IF condition no longer satisfied, once execution has begun it continues to its normal termination.

example:

```
*X=2Ø<cr>
*1.1 DO PART 2, 3 TIMES IF X>Ø<cr>
*2.1 SET X = X-5Ø<cr>
*2.2 TYPE X<cr>
*DO PART 1<cr>
             X =      -3Ø        At the start, X = 20
             X =      -8Ø        X is now <0, but iteration
             X =     -13Ø        continues.
*
```

5.5.3 Parenthetical DO

The parenthetical DO command is used to initiate execution of a step or part, while another process is waiting to continue after a STOP or other type of interrupt, without cancelling that other process. A normal DO command automatically cancels any currently stopped process.

The parenthetical DO command includes all the options of the normal DO command. Its general format is:

(DO ...)

examples:

(i) (DO PART 3)

(ii) (DO STEP 1.4 FOR X = 5(5)25)

The parenthetical DO command is commonly used to execute a step or part to test its validity; thus, this command is primarily a debugging aid. Any stopped process which was originally initiated by a parenthetical DO can be cancelled by a parenthetical CANCEL command. Examples of parenthetical DOs can be found under section 5.9.

examples:

(i)

```
*1.1 TYPE "A"<cr>
*1.2 TYPE "D"<cr>
*1.3 TYPE "F"<cr>
*1.4 TYPE "J"<cr>
*1.25 TYPE "E"<cr>
*2.1 TYPE "B"<cr>
*2.2 TYPE "C"<cr>
*3.1 TYPE "G"<cr>
*3.2 TYPE "H"<cr>
*3.3 TYPE "I"<cr>
*DO PART 1<cr>
A
D
E
F
J
*1.15 DO PART 2<cr>
*DO PART 1<cr>
A
B
C
D
E
F
J
*1.35 TO PART 3<cr>
*DO PART 1<cr>
A
B
C
D
E
F
G
H
I
*
```

Note that no return is made to STEP 1.4 (the TO command does not return control).

(ii)

```
*LET I = SQRT(A↑2 + B↑2)<cr>
*1.1 DO PART 2 FOR B = 1(3)9<cr>
*2.1 TYPE A, B, I IN FORM 1<cr>
*FORM 1:<cr>
*   A = ←←←   B = ←←←   C = ←←←←←.←←←←<cr>
*DO PART 1 FOR A = 1(2)12<cr>
   A =   1  B =   1  C =     1.4142
   A =   1  B =   4  C =     4.1231
   A =   1  B =   7  C =     7.0711
   A =   1  B =   9  C =     9.0554
   A =   3  B =   1  C =     3.1623
   A =   3  B =   4  C =     5.0000
   A =   3  B =   7  C =     7.6158
   A =   3  B =   9  C =     9.4868
   A =   5  B =   1  C =     5.0990
   A =   5  B =   4  C =     6.4031
   A =   5  B =   7  C =     8.6023
   A =   5  B =   9  C =    10.2956
   A =   7  B =   1  C =     7.0711
   A =   7  B =   4  C =     8.0623
   A =   7  B =   7  C =     9.8995
   A =   7  B =   9  C =    11.4018
   A =   9  B =   1  C =     9.0554
   A =   9  B =   4  C =     9.8489
   A =   9  B =   7  C =    11.4018
   A =   9  B =   9  C =    12.7279
   A =  11  B =   1  C =    11.0454
   A =  11  B =   4  C =    11.7047
   A =  11  B =   7  C =    13.0384
   A =  11  B =   9  C =    14.2127
   A =  12  B =   1  C =    12.0416
   A =  12  B =   4  C =    12.6491
   A =  12  B =   7  C =    13.8924
   A =  12  B =   9  C =    15.0000
*
```

(iii)

```
*LET A = (B↑2)/4*SQRT(3)<cr>
*1.1 TYPE A, 2*A, ←<cr>
*1.2 STOP IF A>1ØØ<cr>
*2.1 TYPE B<cr>
*DO PART 1 FOR B=1Ø(25)1ØØ<cr>
           A =       43.3Ø127Ø3
         2*A =       86.6Ø254Ø6

           A =      53Ø.44Ø561
         2*A =     1Ø6Ø.88112

STOPPED BY STEP 1.2.
*TYPE A<cr>
           A =      53Ø.44Ø561
*1.Ø STOP IF A>1ØØ<cr>
*DO PART 1 FOR B=1Ø(25)1ØØ<cr>
           A =       43.3Ø127Ø3
         2*A =       86.6Ø254Ø6

STOPPED BY STEP 1.
*
```

5.5.4 Diagnostic Messages

(a) I CAN'T FIND THE REQUIRED STEP.

An incorrect step number has been specified; no such step number exists.

(b) I CAN'T FIND THE REQUIRED PART.

An incorrect part number has been specified; no such part number exists.

5.6 DONE

5.6.1 Description

The DONE command skips execution of the remaining steps of a part during the current iteration. This command can be given indirectly only. It is usually given conditionally.

DONE (unconditional)

> Normally used only as a temporary step (during the testing of a routine) when performing a partial execution.

DONE IF ... (conditional)

> Used to skip execution of the remaining steps of a part when certain conditions (specified in the IF clause) are met.

example:

```
*LET A = B↑2 + 2*B + 1Ø<cr>
*LET C = A↑2 + 2*A*B + B↑2<cr>
*1.1 TYPE A, B, C IN FORM 1<cr>
*1.2 TYPE A*B<cr>
*1.3 TYPE A*C, ←<cr>
*FORM 1:<cr>
*←←←←←.←←←← ←←←←←.←←←← ←←←←←.←←←←<cr>
*DO PART 1 FOR B= 1(1)4<cr>
   13.ØØØØ       1.ØØØØ   196.ØØØØ
         A*B =        13
         A*C =      2548

   18.ØØØØ       2.ØØØØ   4ØØ.ØØØØ
         A*B =        36
         A*C =      72ØØ

   25.ØØØØ       3.ØØØØ   784.ØØØØ
         A*B =        75
         A*C =     196ØØ

   34.ØØØØ       4.ØØØØ  1444.ØØØØ
         A*B =       136
         A*C =     49Ø96

*1.15 DONE<cr>                              Insert temporary premature
*DO PART 1 FOR B=1(1)4<cr>                  termination step following 1.1.
   13.ØØØØ       1.ØØØØ   196.ØØØØ
   18.ØØØØ       2.ØØØØ   4ØØ.ØØØØ
   25.ØØØØ       3.ØØØØ   784.ØØØØ
   34.ØØØØ       4.ØØØØ  1444.ØØØØ
*1.15 DONE IF B>2<cr>                       Change unconditional DONE
*DO PART 1 FOR B=1(1)4<cr>                  to conditional.
   13.ØØØØ       1.ØØØØ   196.ØØØØ
         A*B =        13
         A*C =      2548

   18.ØØØØ       2.ØØØØ   4ØØ.ØØØØ
         A*B =        36
         A*C =      72ØØ

   25.ØØØØ       3.ØØØØ   784.ØØØØ
   34.ØØØØ       4.ØØØØ  1444.ØØØØ
*
```

5.6.2 Diagnostic Messages

(a) DON'T GIVE THIS COMMAND DIRECTLY

The DONE command must only be given indirectly (preceded by a step number).

5.7 FILE

5.7.1 Description

FILE stores an item in the disk file currently in use. Core storage is not affected in any way.

$$\text{FILE} \left\{ \begin{array}{l} \text{a} \\ \text{a(b,...)} \\ \text{FORM m} \\ \text{STEP m.n} \\ \text{PART m} \\ \text{FORMULA f} \\ \text{ALL STEPS} \\ \text{ALL PARTS} \\ \text{ALL FORMULAS} \\ \text{ALL FORMS} \\ \text{ALL VALUES} \\ \text{ALL} \end{array} \right\} \text{AS ITEM n (code)}$$

Stores the specified information as ITEM n (where n can be in the range 1 through 25) in the currently open disk file. (code) is optional for documentation only and has no meaning to AID; however, code, if used, must not exceed five characters in length.

example:

```
*FILE ALL PARTS AS ITEM 5<cr>
DONE.
*
```

All parts existing in core storage are stored on the currently open disk file as ITEM 5. Successful execution of the command is evidenced by the AID response, DONE. Item contents can be retrieved by the RECALL command.

5.7.2 Diagnostic Messages

(a) ITEM NUMBER MUST BE POSITIVE INTEGER <=25.

An invalid item number was given.

(b) PLEASE DISCARD THE ITEM OR USE A NEW ITEM NUMBER.

The specified item is already occupied; no change in either immediate or external storage occurs.

(c) PLEASE LIMIT ID'S TO 5 LETTERS AND/OR DIGITS.

Code exceeds five characters in length.

(d) YOU HAVEN'T TOLD ME WHAT FILE TO USE.

A FILE command was attempted before an external storage file was opened via a USE command.

5.8 FORM

5.8.1 Description

FORM is used to edit typeouts of results for purposes of readability. For example:

(a) to specify that results be typed in a specific notation, (either scientific or fixed point),

(b) to specify that multiple results are to be printed on a single line, usually to conserve space,

(c) to intersperse text with results, and

(d) to produce report-type headings.

The elements that can be typed in a form are:

(a) Numeric values, including variables, $ (line counter), TIME, TIMER, and SIZE.

(i) TYPE -23.466 IN FORM 1

(ii) TYPE A, B, C IN FORM 2

(iii) TYPE $ IN FORM 3

(iv) TYPE TIMER IN FORM 4

(b) Propositional values (TRUE and FALSE). Both of these values must be provided with an integral form field containing at least five character positions.

example:

TYPE F IN FORM 5 (where F is a proposition)

(c) ← (indicating a blank field).

example:

TYPE A, B, ←, F IN FORM 6.

Forms are entered as two lines.

example:

```
*FORM n:<cr>                    n identifies the specific form
*actual-format-typed-here<cr>
```

Once a form is defined, it can be used by specifying it in a TYPE command.

TYPE ... IN FORM n

5.8.2 Specific Notations

Fixed-point notation is specified by a series of left arrows, one for each digit position and one for a sign (if any). If less integer places appear in the form than in the result, the error message I CAN'T EXPRESS THE VALUE IN YOUR FORM is typed; if less decimal places appear in the form than in the result, rounding occurs. A period is used to indicate the decimal point position.

```
←←←←.←←←
-345.667
←←←←.←←←←←←
-345.666667
```

At least seven periods must appear in a scientific notation form.

```
.......
-3.3-01
```

Reducing the number of periods in a scientific notation form reduces the number of fraction digits appearing in the result; these digits are dropped after rounding.

5.8.3 Multiple Results on a Single Line

More than one result can be typed on a single line through the use of the FORM command. Such a technique might be used to conserve space, increase output speed, and/or cause results to appear under previously typed column headings.

example:

```
*FORM 1:<cr>
*  ←←.← ←←←←←←←←.← ←←←←←←←←.←← ←←←←←←←.←←←<cr>
*1.1 TYPE A, A↑2, A↑3, A↑4 IN FORM 1<cr>
*DO STEP 1.1 FOR A = 1Ø(.5)15<cr>
  1Ø.Ø      1ØØ.Ø     1ØØØ.ØØ  1ØØØØ.ØØØ
  1Ø.5      11Ø.3     1157.63  12155.Ø63
  11.Ø      121.Ø     1331.ØØ  14641.ØØØ
  11.5      132.3     152Ø.88  1749Ø.Ø63
  12.Ø      144.Ø     1728.ØØ  2Ø736.ØØØ
  12.5      156.3     1953.13  24414.Ø63
  13.Ø      169.Ø     2197.ØØ  28561.ØØØ
  13.5      182.3     246Ø.38  33215.Ø63
  14.Ø      196.Ø     2744.ØØ  38416.ØØØ
  14.5      21Ø.3     3Ø48.63  442Ø5.Ø63
  15.Ø      225.Ø     3375.ØØ  5Ø625.ØØØ
*
```

5.8.4 Interspersing Text with Results

A form can be used to intersperse explanatory text with typed results.

example:

```
*FORM 2:<cr>
*IF A = ←←←.←  THEN A↑2 = ←←←←←.←←←←<cr>
*2.1 TYPE A, A↑2 IN FORM 2<cr>
*DO PART 2 FOR A = 1Ø(.5)12<cr>
IF A =  1Ø.Ø  THEN A↑2 =   1ØØ.ØØØØ
IF A =  1Ø.5  THEN A↑2 =   11Ø.25ØØ
IF A =  11.Ø  THEN A↑2 =   121.ØØØØ
IF A =  11.5  THEN A↑2 =   132.25ØØ
IF A =  12.Ø  THEN A↑2 =   144.ØØØØ
*
```

5.8.5 Report-Type Headings

A form containing only text can be used to generate columnar headings.

example:

```
*FORM 3:<cr>
*    A          A↑2           A↑3           A↑4<cr>
*1.1 TYPE FORM 3<cr>
*1.2 DO STEP 1.3 FOR A = 1Ø(.5)12<cr>
*1.3 TYPE A, A↑2, A↑3, A↑4 IN FORM 1<cr>
*FORM 1:<cr>
*  ←←.←  ←←←←←.←←←←  ←←←←←.←←←←  ←←←←←.←←←←<cr>
*DO PART 1<cr>
    A          A↑2           A↑3           A↑4
  1Ø.Ø     1ØØ.ØØØØ    1ØØØ.ØØØØ   1ØØØØ.ØØØØ
  1Ø.5     11Ø.25ØØ    1157.625Ø   12155.Ø625
  11.Ø     121.ØØØØ    1331.ØØØØ   14641.ØØØØ
  11.5     132.25ØØ    152Ø.875Ø   1749Ø.Ø625
  12.Ø     144.ØØØØ    1728.ØØØØ   2Ø736.ØØØØ
  12.Ø     144.ØØØØ    1728.ØØØØ   2Ø736.ØØØØ
*
```

5.8.6 Diagnostic Messages

(a) FORM NUMBER MUST BE INTEGER AND 1<= FORM<10↑9

Form numbers must be integers in the range 1 through 10^9-1.

(b) I CAN'T EXPRESS THE VALUE IN YOUR FORM.

A value cannot be expressed in the format given (the value is too large).

(c) I CAN'T FIND THE REQUIRED FORM.

The specified form does not exist; the form number is incorrect.

(d) I HAVE TOO MANY VALUES FOR THE FORM.

The TYPE command specifies more elements to be typed than there are fields in the form.

5.9 GO

5.9.1 Description

GO continues execution of a currently stopped (interrupted) process. GO is the antithesis of the CANCEL command.

The GO command is normally used to continue execution after control has been returned to the user via a STOP command. The GO command must be given directly only.

example:

```
*LET Y = P*(R↑2)H<cr>
*P=3.142<cr>
*1.1 DO PART 2 FOR H = .5(.5)3<cr>
*1.2 STOP<cr>
*2.1 TYPE R, H, Y IN FORM 1<cr>
*3.1 TYPE R↑2<cr>
*3.2 DELETE STEP 1.2<cr>
*FORM 1:<cr>
*  ←←←  ←←←.←  ←←←←←.←←←←<cr>
*DO PART 1 FOR R=1(1)3<cr>
ERROR AT STEP 2.1 (IN FORMULA Y):  EH?
*TYPE FORMULA Y<cr>
          Y:    P*(R↑2)H    Multiplication symbol was omitted.
*LET Y=P*(R↑2)*H<cr>        Correct formula.
*GO<cr>                     Execute GO to continue.
    1     .5       1.571Ø
    1    1.Ø       3.142Ø
    1    1.5       4.713Ø
    1    2.Ø       6.284Ø
    1    2.5       7.855Ø
    1    3.Ø       9.426Ø
STOPPED BY STEP 1.2.        STOP command at STEP 1.2 is
                            encountered.
*(DO PART 3)<cr>
          R↑2 =         1
DONE.  I'M READY TO GO FROM STEP 1.2, ALTHOUGH I CAN'T FIND IT.
*GO<cr>                     Execute PART 3 via a parenthetical
    2     .5       6.284Ø   DO;  then GO to continue.
    2    1.Ø      12.568Ø
    2    1.5      18.852Ø
    2    2.Ø      25.136Ø
    2    2.5      31.42ØØ
    2    3.Ø      37.7Ø4Ø
```

```
        3     .5         14.139Ø
        3    1.Ø         28.278Ø
        3    1.5         42.417Ø
        3    2.Ø         56.556Ø
        3    2.5         7Ø.695Ø
        3    3.Ø         84.834Ø
    *GO<cr>
    I HAVE NOTHING TO DO.
    *
```

5.9.2 Diagnostic Messages

(a) DON'T GIVE THIS COMMAND INDIRECTLY.

The GO command can be given directly only (with no step number preceding it).

(b) I HAVE NOTHING TO DO.

When the GO command was given, no process was in a stopped or interrupted status. Control returns to the user and AID waits for a new command.

5.10 IF CLAUSE

5.10.1 Description

The IF clause can be appended to any command (except the short SET command) to make that command conditional; (the command is executed only if the proposition following the word IF is satisfied).

verb (arguments) IF proposition

examples:

(i) 1.1 SET B = 5Ø IF A>1ØØ

Set B equal to 50 if, and only if, A is greater than 100; otherwise leave the value of B undisturbed.

(ii) 3.3 TO PART 5 IF FP(D)=Ø

Transfer control to PART 5 if, and only if, D is an integer; otherwise, continue in sequence.

(iii) 2.9 DO PART 3 IF TV(F)=1

Execute PART 3 if the truth value (TV) of proposition F is equal to 1; otherwise, continue in sequence.

5.11 LET

5.11.1 Description

LET defines arithmetic formulas, Boolean expressions (propositions), and user functions. The formula, expression, or function with which an identifier is associated is re-evaluated each time that identifier appears during execution of a routine.

5.11.2 Arithmetic Formulas

The LET command can be used to tell AID how to calculate the value of an identifier (versus associating the identifier with a fixed value, as with the SET command). LET causes the identifier on the left of the equals sign to be set to the formula on the right of the equals sign.

examples:

(i)
```
LET V = P*(R↑2)*L
SET P = 3.1416
```

(ii)
```
LET L = W*H
```

5.11.3 Boolean Expressions

LET can also be used to equate an identifier to the value (true or false) of a proposition (a Boolean expression) composed of arithmetic and logical statements using common relational operators (e.g. =, <, >), the logical negation (not), and logical operators (and, or).

examples:

(i)
```
LET A = TRUE
```

(ii)
```
LET C = A AND B OR C OR D
```

(iii)
```
LET Y = X AND Y OR (SQRT(1ØØ)<SQRT(Z))
```

5.11.4 User Functions

LET has a third use, that of defining a user function.

examples:

(i)
```
LET A(B,C) = B*C
```

(ii)
```
LET V(R, H) = P(R↑2)*H
```

User functions, once defined, are used in exactly the same manner as AID functions.

examples:

(i)   TYPE A(12,30)

(ii)   LET M = V(F,G)*D

A more complete discussion of these three uses of LET, including examples, can be found in section 3.2.

5.11.5 Special LET Command

LET s BE SPARSE

where s is a subscripted letter.

This command declares undefined array elements to have zero value; such elements require no space in immediate storage.

example:

```
*X(1,2)=55<cr>
*X(1,5)=43<cr>
*X(1,1Ø)=6Ø<cr>
*X(2,4)=77<cr>
*TYPE X(1,1Ø)<cr>
     X(1,1Ø) =      6Ø
*TYPE X(1,3)<cr>
X(1,3) = ???
*LET X BE SPARSE<cr>              Set all undefined x array items
                                  to zero.
*TYPE X(1,3)<cr>
      X(1,3) =       Ø
*TYPE X(2,1)<cr>
      X(2,1) =       Ø
*TYPE X<cr>
      X(1,2) =      55            Only those elements that have
      X(1,5) =      43            been explicitly defined are typed,
     X(1,1Ø) =      6Ø            followed by a message reminding the
      X(2,4) =      77            user that he has defined X as sparse.
X IS SPARSE
*
```

Although an array may be defined as sparse, at least one element in the array must be given an explicit value (so that AID will know the dimensions of the array) before any attempt is made to refer to an item within the array.

example:

```
*LET D BE SPARSE<cr>
*TYPE D(1,3,5)<cr>
D = ???
*TYPE D<cr>
D = ???
*D(2,4,6)=2Ø<cr>
*TYPE D(1,3,5)<cr>
    D(1,3,5) =          Ø
*TYPE D<cr>
    D(2,4,6) =         2Ø
D IS SPARSE
*
```

5.12 LINE

5.12.1 Description

The LINE command advances the Teletype page one line.

LINE

The LINE command is often given conditionally:

LINE IF proposition

example:

```
*1.1 TYPE "VOLUME CALCULATION"<cr>
*1.2 LINE<cr>
*1.3 TYPE A, B, C, A*B*C<cr>
*A=3<cr>
*B=5<cr>
*C=12<cr>
*DO PART 1<cr>
VOLUME CALCULATION
                                     AID advances paper form one line.
           A =        3
           B =        5
           C =       12
       A*B*C =      18Ø
*
```

Note that the steps above perform essentially the same process as the commands:

1.1 TYPE "VOLUME CALCULATION"

1.2 TYPE ←, A, B, C, A*B*C

5.13 PAGE

5.13.1 Description

PAGE advances the Teletype paper form to the top of the next page. The PAGE command can be used in conjunction with the $ symbol, which represents the current line count, (the number of lines printed thus far on the current page. AID allows for a maximum of 54 lines per page.

example:

```
*1.Ø PAGE<cr>
*1.1 TYPE "SQUARE ROOT VALUES FOR 1-1ØØ"<cr>
*1.2 DO PART 2 FOR A = 1(1)1ØØ<cr>
*2.1 TYPE A, SQRT(A)<cr>
*2.2 DO PART 3 IF $>45<cr>
*3.1 PAGE<cr>
*3.2 DO STEP 1.1<cr>
*DO PART 1<cr>

          (skip to new page)
SQUARE ROOT VALUES FOR 1-1ØØ

          (44 lines of typeout)

          (skip to next page)
SQUARE ROOT VALUES FOR 1-1ØØ

          (44 lines of typeout)

          (skip to next page)
SQUARE ROOT VALUES FOR 1-1ØØ

          (remaining 12 lines of typeout)

*
```

5.14 QUIT

5.14.1 Description

QUIT skips execution of the remaining steps of a part and satisfies the DO command for that part by cancelling any further iterations. The QUIT command is usually given conditionally.

QUIT (unconditional)

Normally used only as a temporary step (inserted for the purpose of testing a portion of a routine) when performing a partial execution.

QUIT IF ... (conditional)

Used to skip execution of the remaining steps of a part (and any further iterations of the part by the current DO command) when certain conditions are present.

example:

```
*LET A=B↑2 + 2*B + 1Ø<cr>
*LET C = A↑2 + 2 + A + B + B↑2<cr>
*1.1 TYPE A, B, C<cr>
*1.2 TYPE A*B<cr>
*1.3 TYPE A*C<cr>
*DO PART 1 FOR B=5(5)15<cr>
                 A =       45
                 B =        5
                 C =     21Ø2
               A*B =      225
               A*C =    9459Ø
                 A =      13Ø
                 B =       1Ø
                 C =    17142
               A*B =     13ØØ
               A*C =        2.22846*10↑6
                 A =      265
                 B =       15
                 C =    7Ø732
               A*B =     3975
               A*C =        1.874398*10↑7
*1.15 QUIT IF A > 1ØØ<cr>
*DO PART 1 FOR B=5(5)15<cr>
                 A =       45
                 B =        5
                 C =     21Ø2
               A*B =      225
               A*C =    9459Ø
                 A =      13Ø
                 B =       1Ø
                 C =    17142

*
```

5.15 RECALL

5.15.1 Description

RECALL reads an item, previously stored by a FILE command, from the currently open disk file into core storage. The contents of the item then exist both on the disk file and in core. All steps, identifiers, forms, etc., which were in core before the RECALL command was given remain unchanged, with the exception of those which are redefined by the recalled item.

RECALL ITEM m (code)

This reads in ITEM m (where m can be in the range 1 through 25) from the currently open disk file. (code) is optional for documentation purposes only and is ignored by AID; however, code, if used, cannot exceed five characters in length.

example:

```
*RECALL ITEM 23<cr>
DONE.
*
```

The contents of ITEM 23 of the currently open file are read into core storage. Successful execution of the RECALL command is evidenced by the AID response, DONE.

5.15.2 Diagnostic Messages

(a) I CAN'T FIND THE REQUIRED ITEM

The specified item cannot be found in the currently open file. Either the wrong file is open, the item number is incorrect, or the item was never filed.

(b) ITEM NUMBER MUST BE POSITIVE INTEGER<=25.

An invalid item number was given.

(c) PLEASE LIMIT ID'S TO 5 LETTERS AND/OR DIGITS.

Code exceeds five characters in length.

(d) YOU HAVEN'T TOLD ME WHAT FILE TO USE.

A RECALL command was attempted before a disk file was opened via a USE command.

5.16 RESET TIMER

5.16.1 Description

Resets TIMER to zero.

RESET TIMER

TIMER is a counter used by AID to keep track of the amount of central processor time spent by the user in running AID. This cumulative running time can be obtained at any point by typing the request TYPE TIMER. Each time the user wishes to reset the timer and to begin timing a new operation, he types RESET TIMER.

5.17 SET

5.17.1 Description

SET defines an identifier as equivalent to a fixed value. This value is calculated once and the result is then used whenever the identifier appears in a calculation.

SET x = expression or value

If an expression, the expression must be immediately reducible to a numeric value.

When the SET command is typed as a direct command, the verb (SET) may be omitted. This form is called a short SET command.

examples:

(i) SET A = 2Ø

(ii) A = 2Ø

(iii) SET A = SQRT(2Ø)+43.5↑2

(iv) SET D = TRUE

(v) F = FALSE

A more complete discussion of the use of SET commands, including additional examples, can be found in section 3.1.

5.18 STOP

5.18.1 Description

The STOP command temporarily halts the current process at the point where the STOP command appears and returns control to the user. The stopped process can be resumed by typing GO. If the user does not desire to continue the process, he types CANCEL.

The STOP command can be given indirectly only.

STOP (unconditional)

Normally used only as a temporary step (during the testing of a routine) when performing a partial execution.

STOP IF ... (conditional)

Used to halt execution temporarily and return control to the user when certain conditions (specified in the IF clause) are met.

example:

```
*LET B=16-C<cr>
*LET A=3.17568/B<cr>
*1.1 STOP IF B=Ø<cr>                      STOP command prevents attempt
*1.2 TYPE A, B, C IN FORM 1<cr>           to divide by zero in formula A.
*FORM 1:<cr>
*  ..............     ..............     ..............<cr>
*DO PART 1 FOR C = -4(4)24<cr>
   1.58784ØØØ-Ø1     2.ØØØØØØØØ Ø1    -4.ØØØØØØØØ ØØ
   1.98480ØØØ-Ø1     1.6ØØØØØØØ Ø1     Ø
   2.64640ØØØ-Ø1     1.2ØØØØØØØ Ø1     4.ØØØØØØØØ ØØ
   3.96960ØØØ-Ø1     8.ØØØØØØØØ ØØ     8.ØØØØØØØØ ØØ
   7.93920ØØØ-Ø1     4.ØØØØØØØØ ØØ     1.2ØØØØØØØ Ø1
STOPPED BY STEP 1.1.
*TYPE C<cr>
           C =       16
*TYPE B<cr>
           B =       Ø
*
```

5.19 TO

5.19.1 Description

TO discontinues the sequential execution of the part currently being executed and transfers control to another step or part. When the new part is finished, the direct command that initiated the execution is satisfied.

The TO command can be given indirectly only.

```
        PART m
TO
        STEP m.n
```

example:

```
*1.1 DEMAND G<cr>                        Demand gross pay for week.
*1.2 DEMAND T<cr>                        Demand total tax year-to-date.
*1.3 TO PART 2 IF T>=56Ø<cr>             $560 = maximum deduction/year.
*1.4 LET D=G*.Ø46<cr>                    D = current deduction.
*1.5 TO PART 3 IF (T+D)>56Ø<cr>
*1.6 TYPE "DEDUCTIONS"<cr>
*1.7 TYPE D, D+T, ←, ←<cr>
*2.1 TYPE "NO DEDUCTION REQUIRED"<cr>
*2.2 LET D=Ø<cr>
*2.3 TO STEP 1.6<cr>
*3.1 LET D=56Ø-T<cr>
*3.2 TO STEP 1.6<cr>
*DO PART 1, 4 TIMES.<cr>
          G = *125.ØØ<cr>
          T = *34Ø.ØØ<cr>
DEDUCTIONS
          D =        5.75
        D+T =      345.75


          G = *35Ø.ØØ<cr>
          T = *545.ØØ<cr>
DEDUCTIONS
          D =        15
        D+T =        56Ø


          G = *1Ø3.45<cr>
          T = *559.Ø4<cr>
DEDUCTIONS
          D =   .96
        D+T =        56Ø


          G = *3ØØ.ØØ<cr>
          T = *565.ØØ<cr>
NO DEDUCTION REQUIRED
DEDUCTIONS
          D =         Ø
        D+T =       565

*
```

5.19.2 Diagnostic Messages

(a) DON'T GIVE THIS COMMAND DIRECTLY

The TO command must only be given indirectly (preceded by a step number).

5.20 TYPE

5.20.1 Description

Types out the specified information on the user's console.

TYPE

n	where n is numeric value or expression
s	where s is a subscripted variable
s(a,b,...)	
p	where p is a proposition
"any text"	
←	← represents a null item (a blank field when typing in form x, or a blank line)
FORM m	
STEP m.n	
PART m	
FORMULA f	
F(n,...)	function of (n, ...)
F(p)	function (TV) of a proposition
ALL STEPS	
ALL PARTS	
ALL FORMULAS	
ALL FORMS	
ALL VALUES	
ALL	
TIME	current time of day in minutes since midnight
TIMER	processor time used (see RESET TIMER)
SIZE	amount of immediate storage being used
ITEM-LIST	item list of currently open external storage file
USERS	always returns an answer of 0.

Several individual TYPE commands (except for TYPE "any text" or TYPE ITEM-LIST) can be combined into one command.

example:

```
TYPE ALL PARTS, 1243, FORMULA D, FORM 5
```

Each entry, however, is still typed on a separate line.

5.20.2 IN FORM Option

Output editing can be performed by appending the IN FORM ... option to a TYPE command, see section 5.8. Note that only certain types of entries can be typed in forms.

example:

```
*B=2Ø<cr>                          Variables
*C=3Ø<cr>
*D=111.333<cr>
*LET A(B,C)=(B↑2)*(C↑2)<cr>        User functions
*LET F(B)=B/2<cr>
*E(1) = 16<cr>                     Subscripted variables
*E(2) = 25<cr>
*E(3) = 35<cr>
*LET G = H OR I AND J<cr>          Propositions
*H=TRUE<cr>
*I=FALSE<cr>
*J=FALSE<cr>
*LET K = B*C*D<cr>                 Formulas
*LET M = K/2*SQRT(K)<cr>
*FORM 1:<cr>                       Forms
*  .........  POUNDS IS  .........  OUNCES<cr>
*FORM 2:<cr>
*  POUNDS                      OUNCES<cr>
*1.1 TYPE B,C<cr>                  Parts and steps
*1.2 TYPE D, E<cr>
*1.3 TO PART 2<cr>
*2.1 TYPE FORM 2<cr>
*2.2 TYPE D/E(1), D IN FORM 1<cr>
*3.1 TYPE K, M<cr>
*3.2 TYPE A(E(1),E(2))<cr>
*DO PART 1<cr>
          B =       2Ø
          C =       3Ø
          D =      111.333
       E(1) =       16
       E(2) =       25
       E(3) =       35
  POUNDS                      OUNCES
   6.958 ØØ  POUNDS IS   1.113 Ø2  OUNCES
*TYPE E<cr>                        A command to type the values
       E(1) =       16             of a subscripted letter results
       E(2) =       25             in typeouts of all values.
       E(3) =       35
*TYPE A(3,6)<cr>
     A(3,6) =      324
```

```
*TYPE A<cr>
      A(B,C):  (B↑2)*(C↑2)
*TYPE G<cr>
           G =       TRUE
*TYPE TV(G)<cr>
       TV(G) =         1
*TYPE FORM 1<cr>
  .........  POUNDS IS  .........  OUNCES
*TYPE STEP 1.3<cr>
1.3 TO PART 2.
*TYPE FORMULA K<cr>
           K:   B*C*D
*TYPE ALL STEPS<cr>

1.1 TYPE B,C.
1.2 TYPE D, E.
1.3 TO PART 2.

2.1 TYPE FORM 2.
2.2 TYPE D/E(1), D IN FORM 1.

3.1 TYPE K, M.
3.2 TYPE A(E(1),E(2)).

*TYPE ALL FORMULAS<cr>

      A(B,C):  (B↑2)*(C↑2)
        F(B):   B/2
           G:   H OR I AND J
           K:   B*C*D
           M:   K/2*SQRT(K)

*TYPE ALL FORMS<cr>

FORM 1:
  .........  POUNDS IS  .........  OUNCES

FORM 2:
  POUNDS                 OUNCES
```

```
*TYPE ALL VALUES<cr>

          B =       2Ø
          C =       3Ø
          D =      111.333
          H =      TRUE
          I =      FALSE
          J =      FALSE

       E(1) =       16
       E(2) =       25
       E(3) =       35

*TYPE ALL<cr>

1.1 TYPE B,C.
1.2 TYPE D, E.
1.3 TO PART 2.

2.1 TYPE FORM 2.
2.2 TYPE D/E(1), D IN FORM 1.

3.1 TYPE K, M.
3.2 TYPE A(E(1),E(2)).

FORM 1:
  .........  POUNDS IS  .........  OUNCES

FORM 2:
  POUNDS                   OUNCES

      A(B,C):  (B↑2)*(C↑2)
        F(B):   B/2
           G:   H OR I AND J
           K:   B*C*D
           M:   K/2*SQRT(K)

          B =       2Ø
          C =       3Ø
          D =      111.333
          H =      TRUE
          I =      FALSE
          J =      FALSE

       E(1) =       16
       E(2) =       25
       E(3) =       35
```

```
*TYPE TIME<cr>
        TIME:   1149
*TYPE TIMER<cr>                          The total central processor time
       TIMER =         1Ø.76             utilized by the user thus far.
                                              mm = minutes
                                           ss.ss = seconds to the nearest
                                                   hundredth
*TYPE SIZE<cr>                           The number of 'cells' of immediate
        SIZE:   62                       storage currently occupied by the
                                         user's work area.  In a 13K
                                         environment, approximately 1900
                                         such 'cells' are available for
                                         this purpose.
*USE FILE 11Ø<cr>
ROGER.
*FILE ALL FORMULAS AS ITEM 1 (FMULA)<cr>
DONE.
*FILE ALL FORMS AS ITEM 2 (FORM)<cr>
DONE.
*FILE ALL VALUES AS ITEM 3 (VALUE)<cr>
DONE.
*FILE ALL STEPS AS ITEM 1Ø (STEP)<cr>
DONE.
*TYPE ITEM-LIST<cr>
    ITEM-LIST
ITEM         DATE                        DATE is the creation date.
 1          18-FEB-71
 2          18-FEB-71
 3          18-FEB-71
1Ø          18-FEB-71
*
```

5.21 USE

5.21.1 Description

USE makes disk files available for use. The file thus addressed remains open for use until another USE command is given or until the AID program is terminated. Files created by AID are in a special AID (8-bit) character format, not ASCII.

Once a file has been opened by a USE command, all DISCARD, FILE, RECALL, and TYPE ITEM-LIST commands are assumed to refer to that file.

example:

```
*USE FILE 1Ø3<cr>
ROGER.
*
```

Makes file 103, on disk, available for use. Successful execution of the command is evidenced by the AID response ROGER.

5.21.2 Diagnostic Messages

(a) FILE NUMBER MUST BE POSITIVE INTEGER<=2750.

Filename is less than 0 or greater than 2750.

APPENDIX A

A GLOSSARY OF AID TERMS

← Indicates a blank field, when types in conjunction with a FORM; otherwise, a blank line.

$ A special symbol which refers to the line counter kept by AID. This symbol can be used in TYPE commands, or it can be tested within a conditional command to cause a line feed (LINE) or advance to next head of form (PAGE) at appropriate points.

Conditional expression	A series of clauses separated by semicolons, with each clause made up of a proposition followed by a colon and then by an expression. The entire conditional expression must be enclosed in parentheses or brackets.
External storage	See 'File'.
Expression	An arithmetic formula, Boolean proposition, or function.
File	The disk is used for the preservation of user subroutines, values, etc. Files are manipulated by the DISCARD, FILE, and RECALL commands.
Form	A user-specified format for editing output via the TYPE command.
Formula	An arithmetic expression defined by the user via the LET command.
Function	An arithmetic or Boolean function provided by AID (see Table 4-2) or defined by the user via the LET command.

Identifier	A single alphabetic character associated with a variable (via the LET or SET commands) and then used to access the current value of the variable.
Immediate storage	Core storage work area. In a 13K environment, approximately 4K of core is available to the user for his steps, values, tables, etc.
Item-list	The file directory associated with an external storage file. The user can obtain a listing of the directory of the currently open file by typing the command TYPE ITEM-LIST.
Part	A series of indirect steps, the step numbers of which have the same integral value.
SIZE	A noun which can be specified in a TYPE command to obtain a typeout of the amount of core 'cells' used. (1 cell = approx. 2 core words).
Step	Any one-line command typed by the user. A step can be direct (executed immediately), in which case no step number precedes it, or indirect (to be executed later), in which case it is preceded by a step number.
TIME	A noun which can be specified in a TYPE command to obtain a typeout of the time of day (in 24-hour format).
TIMER	A noun which can be specified in a TYPE command to obtain a typeout of the amount of central processor time spent thus far by the user during the current AID run.
Variable	An element defined by either the LET or SET commands which is associated with some value. It is referred to by an associated tag called an identifier.

APPENDIX B

AID COMMAND SUMMARY

Command Format Symbology

l = letter
s = subscripted letter
m,n,p = numeric values
f = formula
F = function
range = an iterative sequence or series of value.

Type Symbology

D = direct command only
I = indirect command only
O = operational command
F = file command
S = special command

B.1 LIST OF ALL COMMANDS

Command Format	Type	Description
CANCEL	D,O	Cancels a currently stopped process when the user does not desire to resume execution.
(CANCEL)	D,O	Cancels a currently stopped process which was initiated by a parenthetical DO.

Command Format	Type	Description
DELETE { 1 s s(m,n) FORM m STEP m.n PART m FORMULA f ALL STEPS ALL PARTS ALL FORMULAS ALL FORMS ALL VALUES ALL	0	Erases the specified item from core storage and frees the space occupied by it for some other use.
DEMAND { 1 s(m,n) 1 AS "any text" s(m,n) AS "any text"	I,0	Causes AID to type out a message requesting the user to supply a value for the specified item. Only one variable can be specified in each DEMAND command.
DISCARD ITEM m (code)	F	Deletes ITEM m from the disk file currently in use. code is optional.
DO { STEP m.n STEP m.n, p TIMES STEP m.n FOR 1=range PART m PART m, p TIMES PART m FOR 1=range (DO ... same as above ...)	0	Executes an indirect step or part. If the DO command is a direct step, control returns to the user at the completion of the DO; if an indirect step, control returns to the step following the DO. Initiates a new execution without cancelling the currently stopped process.
DONE	I,0	Skips execution of the remaining steps of a part during the current iteration.

Command Format	Type	Description
FILE { 1 s s(m,n) FORM m STEP m.n PART m FORMULA f ALL STEPS ALL PARTS ALL FORMULAS ALL FORMS ALL VALUES ALL } AS ITEM n (code)	F	Stores the specified item in the disk core file currently open. Core storage is not affected in any way. (code) is optional.
FORM m: ....←←←←← text	O	Defines a format to be used in editing typeouts for purposes of readability. ←←←←←.←←←← fixed point notation up to nine digit positions plus the decimal point) .... scientific notation (minimum of seven positions maximum of fourteen) text any text to be included in the line; <ins>not</ins> enclosed in quotation marks unless they are part of the text.
GO	D,O	Continues execution of a currently stopped process; opposite of the CANCEL command.
IF Clause verb ... IF proposition	M	Can be appended to any command (except the abbreviated SET command) to make the command conditional; the command is executed only if the proposition is true.

Command Format	Type	Description
LET { l=m l=formula F(l)=m F(l)=proposition }	O	Defines arithmetic formulas, Boolean expressions (propositions), and user functions and associates them with identifiers. The formula, expression, or function with which an identifier is associated is re-evaluated each time the identifier appears during an execution.
LET s BE SPARSE	S	Sets undefined array elements to zero.
LINE	O	Advances the Teletype paper form one line.
PAGE	O	Advances the Teletype paper form to the top of the next page.
QUIT	O	Skips execution of the remaining steps of a part and satisfies the DO command for that part by cancelling any further iterations. Usually given conditionally.
RECALL ITEM m (code)	F	Reads an item, previously stored by a FILE command, from the currently open disk file into core. (code) is optional and is for documentation only.
RESET TIMER	S	Resets TIMER to zero.
SET { l=m l=proposition s(m,n)=m s(m,n)=proposition }	O	Defines an identifier as equivalent to a fixed value, which is calculated once and then used whenever the identifier appears. A short form of the SET command, where the word SET is omitted, can be used if the command is direct.

Command Format	Type	Description
STOP	I,O	Temporarily halts the current process at the point where the STOP command appears and returns control to the user. The stopped process can be resumed by typing GO.
TO { PART m STEP m.n	I,O	Discontinues the sequential execution of the part currently being executed and transfers control to another step or part; when the new part is finished, the direct command which initiated the execution is satisfied.
TYPE { m s s(m,n) proposition "any text" ← FORM m STEP m.n PART m FORMULA f F(x) F(proposition) ALL STEPS ALL PARTS ALL FORMULAS ALL FORMS ALL VALUES ALL TIME TIMER SIZE ITEM-LIST	O S S S F	Types out the specified information on the user's console. Several individual TYPE commands may be combined into one (except for TYPE "any text" or TYPE ITEM-LIST). The command TYPE ... IN FORM n causes the listed items to be typed out in the format specified by FORM n. n can be a numeric value (for example, FORM 3) or it can be a numeric formula (for example, FORM (2*x-y)).
USE FILE filename	F	Makes a disk file available for use. The disk file thus addressed remains open for use (by DISCARD, FILE, RECALL, and TYPE ITEM-LIST commands) until another USE command is given or the AID program is terminated.

B.2 FILE COMMANDS

Command Format	Section Reference	Description
DISCARD ITEM m (code)	5.4	Deletes item m from the disk file currently in use. (code) is optional.
FILE { 1 s s(m,n) FORM m STEP m.n PART m FORMULA f ALL STEPS ALL PARTS ALL FORMULAS ALL FORMS ALL VALUES ALL } AS ITEM n (code)	5.7	Stores the specified item in the disk file currently open. Core storage is not affected in any way. (code) is optional.
RECALL ITEM m (code)	5.15	Reads item m, previously stored by a FILE command, from the disk file into core. (code) is for documentation only.
TYPE ITEM-LIST	5.20	Obtains a typeout of the directory of the currently open disk file.
USE FILE filename	5.21	Makes a disk file available for use. The file thus addressed remains open for use (by DISCARD, FILE, RECALL, and TYPE ITEM-LIST commands) until another USE command is given or the AID program is terminated.

APPENDIX C

AID CHARACTER SET

Standard Mathematical Symbol	AID Symbol	Typing Method Models 33 and 35
	A through Z	Strike appropriate key; no SHIFT.
	0,1 through 9	Strike appropriate key; no SHIFT.
Operators: | | (absolute)	! !	Strike the !, 1 key with SHIFT.
[ ] (brackets)	[ ]	[Strike K with SHIFT.] Strike M with SHIFT.
( ) (parentheses)	( )	(Strike the (,8 key with SHIFT.) Strike the),9 key with SHIFT.
x^e (exponent)	x↑e	Strike the ↑, N key with SHIFT.
/ (divide)	/	Strike the ?,/ key; no SHIFT.
* (multiplication)	*	Strike the *,: key with SHIFT.
+ (addition)	+	Strike the +,; key with SHIFT.
- (subtraction)	-	Strike the =,- key; no SHIFT.

Standard Mathematical Symbol	AID Symbol	Typing Method Models 33 and 35
Boolean Expressions: = (equal)	=	Strike the =,- key with SHIFT.
≠ (not equal)	#	Strike the #,3 key with SHIFT.
≤ (equal to or less than)	<= (2 characters)	Strike the <,. key with SHIFT; then strike the =,- key with SHIFT.
≥ (equal to or greater than)	>= (2 characters)	Strike the >,. key with SHIFT; then strike the =,- key with SHIFT.
	RUBOUT (types back as deleted characters between \\ ).	Strike RUBOUT key to erase each preceding character in error; then type correctly.
null item	←	Strike 0 key with SHIFT.
	$ (current line number)	Strike the $,4 key with SHIFT.
	↑U (cancel entire line)	Strike the U key with CTRL.

APPENDIX D

AID DIAGNOSTIC MESSAGES

Message	Meaning
x = ???	A value has not been supplied by the user for variable x.
DONE.	Signals completion of a file command (DISCARD FILE, RECALL).
DONE. I'M READY TO GO AT / FROM / IN STEP m.n	... AT STEP m.n ... Task was suspended by an interruption or error during the interpretation of an indirect step. ... FROM STEP m.n ... Task was suspended by a stopping command. ... IN STEP m.n ... Task was suspended during an indirectly initiated DO command. AID resumes execution whenever the user types GO.

Message	Meaning
DONE. I'M READY TO GO AT / FROM / IN STEP m.n, ALTHO I CAN'T FIND IT.	Same as above, except that the step at which AID is prepared to resume can no longer be found in immediate storage. Possibly, a direct command (or a routine initiated by a parenthetical DO) has deleted the step in the interim. Upon receipt of a GO command from the user, AID will attempt to resume at the step following the missing step.
DON'T GIVE THIS COMMAND DIRECTLY / INDIRECTLY	This command can be given only indirectly (TO DONE, STOP, DEMAND) or only directly (CANCEL, GO).
EH?	The previously entered line is incorrect. Indirect commands: The step number was incorrectly typed. Direct LET commands: LET x portion is incorrect. Other direct commands: A space was omitted. The terminating period was omitted. The command is not legitimate. An expression is incorrectly written. To continue, retype the command correctly.

Message	Meaning
ERROR AT STEP m.n: EH?	The step number is correct, but the command is incorrect. a. Request a typeout of the step in error. b. Check for the errors listed under "Eh?". c. Retype the command correctly. d. Type GO to continue.
ERROR AT STEP m.n:	The step in error refers to a non-existent step or part.
I CAN'T FIND THE REQUIRED STEP / FORM / PART / FORMULA	Correct the error and type GO to continue.
ERROR AT STEP m.n: (IN FORMULA x): z = ???	The variable z has not been assigned a value by the user. Check for any other errors, define variable z correctly, and type GO to continue.
ERROR IN FORMULA x: EH?	(Following a direct command in which x was used) The form of the expression for x is in error. a. Request a typeout of formula x. b. Check for the errors listed under "Eh?". c. Formula x may be correctly written, but the definition of one or more identifiers is not consistent with their use in formula x.
FILE NUMBER MUST BE POSITIVE INTEGER <=275Ø.	The filename of a USE command must not be greater than the value 2750.

Message	Meaning
FORM NUMBER MUST BE INTEGER AND 1<=FORM<1Ø↑9.	Form numbers must be integers in the range 1 through 10^9-1.
I CAN'T EXPRESS THE VALUE IN YOUR FORM.	A value cannot be expressed in the format specified by the FORM (e.g. the value is too large to specify in fixed point notation). To correct, follow the steps given under 'I HAVE TOO MANY VALUES FOR THE FORM.'
I CAN'T FIND THE REQUIRED FORM / ITEM / PART / STEP	Either the element has never been defined or has been deleted.
I CAN'T MAKE OUT YOUR FIELDS IN THE FORM	The fields in the form specified were typed in such a way that AID cannot distinguish their beginning or ending. Possibly, there are either no fields in the form or two or more are run together with no intervening space.
I HAVE AN ARGUMENT<=Ø FOR LOG.	The argument for the LOG function must be greater than zero.
I HAVE A NEGATIVE ARGUMENT FOR SQRT.	Square root arguments must be positive.
I HAVE A NEGATIVE BASE TO A FRACTIONAL POWER.	An attempt was made to raise a negative value to a fractional power. For example, TYPE (-Y)↑(1/2).
I HAVE AN OVERFLOW.	Some number has exceeded 9.99999999.10↑99 in magnitude.
I HAVE A ZERO DIVISOR.	An attempt was made to divide by zero.
I HAVE NOTHING TO DO.	The user has typed GO, but there is no currently stopped process which can be continued.

Message	Meaning
I HAVE TOO FEW VALUES	An insufficient number of arguments has been supplied for a function.
I HAVE TOO MANY VALUES FOR THE FORM.	There are not enough fields in the form to receive all the values to be typed. a. Type the form and the values. b. Check for errors. c. Change either the TYPE command or the FORM to make them compatible and then type GO to continue.
I HAVE ZERO TO A NEGATIVE POWER.	An attempt was made to raise zero to a negative power.
ILLEGAL SET OF VALUES FOR ITERATION	An error has been detected in a range clause of a function or a DO command, such that the ending value can never be reached (e.g., the increment is zero).
I'M AT STEP m.n	When the user responds to a DEMAND produced request (x=*) with a return only, AID types back this message.
INDEX VALUE MUST BE INTEGER AND !INDEX! <25Ø	All index values (subscripts) must be integral and must have an absolute value of <250.
I NEED INDIVIDUAL VALUES FOR A FORM.	A command was given to type a subscripted variable in a form (e.g., TYPE B IN FORM 1, where B is a subscripted variable). Individual values only can be specified for TYPE ... IN FORM n commands.
I RAN OUT OF SPACE.	User's core memory is filled due to one of the following errors. a. Endless loops because of DO commands or because DO was typed instead of TO.

Message	Meaning
	b. Unlimited recursive definition. c. Variable x defined in terms of y defined in terms of x via LET command. d. Program is too large for available memory; use TYPE SIZE command to determine how much core has been used. File commands can be used to store parts of the routine and execute them one at a time.
ITEM NUMBER MUST BE <=25.	The item number in file commands (DISCARD, FILE, RECALL) must be less than or equal to 25.
NUMBER-OF-TIMES MUST BE INTEGER AND >=Ø	The value specified in the TIMES clause of a DO command must be a positive integer.
PART NUMBER MUST BE INTEGER AND 1<=PART<1Ø↑9.	Part numbers must be integers and in the range 1 through 10^9-1.
PLEASE DELETE THE ITEM OR USE A NEW ITEM NUMBER.	The user has attempted to FILE information into an item which already exists on the currently open disk file. The user must either DISCARD the item prior to filing the new information or use a different item number in the FILE command.
PLEASE KEEP !X! <1ØØ FOR SIN(X) AND COS(X).	Arguments for the SINE and COSINE functions must be less than 100.
PLEASE LIMIT ID'S TO 5 LETTERS AND/OR DIGITS.	Filename in a USE file command or code in a DISCARD, FILE, or RECALL command exceeds five characters in length or contains special characters.
PLEASE LIMIT LINES TO 78 UNITS (CHECK MARGIN STOPS) SAY AGAIN:	User typeins are limited to single-line, 78-character strings.

Message	Meaning
PLEASE LIMIT NUMBERS TO 9 SIGNIFICANT DIGITS.	Numeric values are limited to nine significant digits.
PLEASE LIMIT NUMBER OF INDICES TO 1Ø.	The number of subscripts following an identifier cannot exceed 10.
PLEASE LIMIT NUMBER OF PARAMETERS TO TEN.	The number of arguments for a function is limited to 10.
PLEASE LIMIT STEP LABELS TO 9 SIGNIFICANT DIGITS.	Step numbers can be up to nine digits in length.
REVOKED. I RAN OUT OF SPACE.	See 'I RAN OUT OF SPACE'.
ROGER.	Signals successful completion of a USE file command.
SOMETHING'S WRONG. I CAN'T ACCESS THE FILES.	A system I/O error (or other type of AID error) has occurred. Begin again.
SOMETHING'S WRONG. TRY AGAIN.	AID has found something unusual in its internal records or has received contradictory signals from its I/O routine. Begin again.
SORRY. SAY AGAIN:	A transmission error occurred on the previous typein. This message is preceded by the erroneous line with # symbols typed where the failure occurred. Retype the line.
STEP NUMBER MUST SATISFY 1<=STEP<1Ø↑9.	Step numbers must be in the range 1 through 10^9-1.
STOPPED BY STEP m.n	Process has been temporarily halted by a STOP command at STEP m.n.
YOU HAVEN'T TOLD ME WHAT FILE TO USE.	The user has issued a DISCARD, FILE, RECALL, or TYPE ITEM-LIST command before he has given a USE file command.