9824-6001-RU-000

STL ON-LINE COMPUTER

Volume II — Users' Manual

C. C. Farrington and D. Pope

DECEMBER 28, 1964

PHYSICAL RESEARCH DIVISION

TRW SPACE TECHNOLOGY LABORATORIES

THOMPSON RAMO WOOLDRIDGE INC.

9824-6001-RU-000

The STL On-Line Computer

Volume 2 - User's Manual

C. C. Farrington and D. Pope

December 28, 1964

PHYSICAL RESEARCH DIVISION
TRW/Space Technology Laboratories
One Space Park
Redondo Beach, California

This is the second of a series of three volumes designed to acquaint users with the On-Line Center. The first volume provides the user with a description of the structure of the on-line computer and its general features. Volume 3, On-Line Techniques, gives some specific suggestions for the use of the computer in problem solving. The present volume is intended to serve as a reference manual for those already familiar with the contents of Volume 1 and with the terminology used there. The following pages contain detailed descriptions of the computer's basic set of operations and the procedures for specifying them by means of the keyboards.

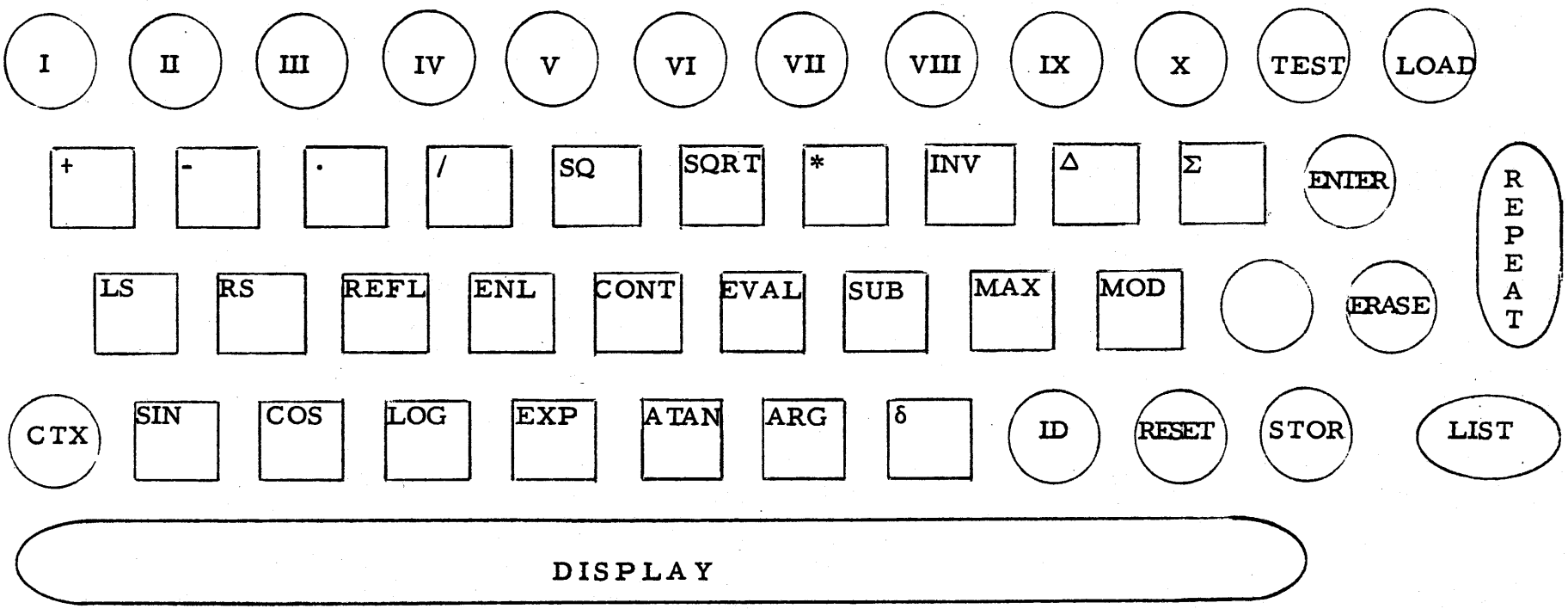


Figure 1. OPERATOR KEYBOARD

"Fixed" keys, whose subroutines are common to all levels, are shown as circles or ellipses; "variable" keys, whose subroutines change from level to level are shown as square. Console programs can be assigned only to variable keys of user levels (XI thru XIX).

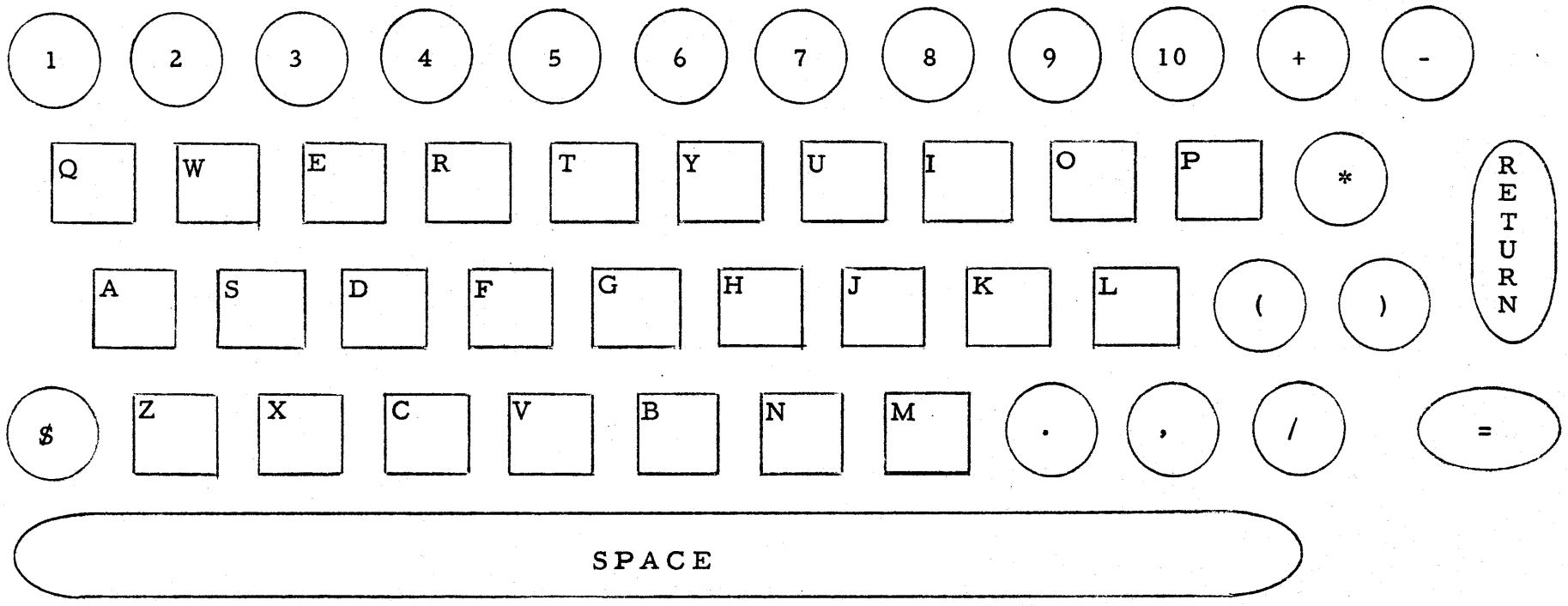


Figure 2. OPERAND KEYBOARD

A. The Operations on Levels II and III

These operations have many features in common which are described here. The second section of this manual describes all of the operations on all levels individually.

1. Types of Operations

The operations on Levels II and III are of two types, predicate-free and single predicate. The predicate-free operations are those which are brought about by pressing a single key. When the key is pressed the operation is performed immediately. The single predicate operations are those which require some additional information before they can be performed. A typical example is the Load operation, which requires information specifying what is to be loaded.

2. Continuing Character of Single-Predicate Operations

All single-predicate operations on Levels II or III are continuing operations in the sense that the operation will be performed on succeeding predicates if two or more predicates are given.

3. Types of Predicates and Their Specification

The single-predicate operations on Level II are Load, Add, Subtract, Divide, Multiply, Evaluate, Substitute, Store, and Display. The Store operation stores the contents of the Y register in the mass memory, and its predicate must be an address. The predicate for all other single predicate operation on Level II may be either an address or a decimal

representation of a numerical constant. If the predicate is an address the vector stored at that address is the "predicate operand" for the operation. The address may be given immediately after the operation key is pressed if the desired operand is in the current bank, or the address may be preceded by a bank change if the desired operand is in another bank. A bank change can also be made before an operation key is pressed. In either case the bank change is made by pressing $ and then the number of the new bank.

If the predicate is the decimal representation of a real number a, then a vector all of whose components are equal to a is generated, and that vector becomes the predicate operand. The vector generated has the same number of components as the vectors in the current array.

An operation is specified by pressing the appropriate key on the left, or operator, keyboard. A predicate is specified by pressing the appropriate key or keys on the right, or operand, keyboard. Addresses are specified simply by pressing the corresponding letter key. Decimal representations are specified by pressing one or more numerical keys, using the form

$$\pm\, a\, b \cdots e.f\, g \cdots h \pm r\, s\ ,$$

to represent the decimal number

$$(\pm\, a\, b \cdots e.f\, g \cdots h) \cdot 10^{\pm rs}$$

This format has the following restrictions and freedoms:

1) An initial + is optional, i.e. in the absence of an initial minus sign the number is assumed positive.

2) Any number of digits may be entered in the mantissa a b ··· e.f g ··· h. If more than eight significant digits are entered the mantissa is rounded off at the eighth.

3) The decimal point may be put anywhere or be omitted. If none is supplied, it is assumed to be after the last digit entered, i.e. the mantissa is assumed to be an integer.

4) The specification of a decimal scale is optional. If one is entered, it may have either one or two digits. (Digits beyond the second are ignored.)

The computer recognizes that the specification of a decimal representation has been completed, and thereupon executes the antecedent single predicate operation, as soon as a letter key, the Carriage Return key, or an operation key is pressed.

The single predicate operations on Level III are Load, Add, Subtract, Multiply, Divide, Substitute, Store, and Display. The predicate for the Load operation may be either an address or a decimal representation. All others accept only addresses. If the decimal representation is given as the predicate, the Load operation on Level III responds in the same way as does the Load operation on Level II. Hece $\alpha + i\beta$ may be input as a constant vector by the instruction

III LOAD $\underline{\alpha}$ REFLECT LOAD $\underline{\beta}$

where $\underline{\alpha}$ and $\underline{\beta}$ denote the appropriate decimal representations.

4. Standardization

An n component real vector is stored in "floating vector" form. One memory cell is used to store a binary scale. The next n cells are used to store the "mantissas", which are numbers between zero and one in magnitude. The true value of a component is

$$X_j = x_j \cdot 2^{S[X]}$$

where x_j is the mantissa and $S[X]$ is the binary scale. A non-zero real vector is said to be in "standard form" when at least one mantissa satisfies the condition

$$\frac{1}{2} \leq x_j < 1 \quad .$$

A complex vector is stored as a pair of real vectors. The true value of a component of a complex vector is

$$X_j + iY_j = x_j \cdot 2^{S[X]} + iY_j \cdot 2^{S[Y]}$$

A non-zero complex vector is said to be in standard form if at least one mantissa, from either the real or imaginary part, is between 1/2 and 1 in magnitude and the binary scales of its real and imaginary parts are equal. Hence the true value of a component of a complex vector in standard form is

$$Y_j + iY_j = (x_j + iY_j) \cdot 2^S$$

where $S = S[X] = S[Y]$.

Several of the operations on Levels II and III, namely Load (from memory), Store, Substitute, Display, Reflect, Right Shift, and Left Shift, merely transfer data from one place to another without altering it. The other operations, such as Add and Sine, involve arithmetic calculations and will, in general, alter data. All of the operations in the latter category, with the exceptions of Expand and Contract on both levels Load (a constant) on Level III, leave the calculated result in standard form. Note, however, that a complex vector formed by operations on Level II or by Substitute on Level III may not in general be in standard form, since $S[X]$ and $S[Y]$ may not be equal.

5. Zero Convention

All operations which from their mathematical definitions would give an improper or overflow result cause the answer to be set to zero. Thus dividing by zero, taking the square root or logarithm of a negative number on Level II, evaluating a function outside its interval of definition, and seeking the argument of zero on Level III all cause the corresponding component to be set to zero. Likewise, exponentiating or taking sine or cosine of a vector having any component larger than 2^{25} causes the entire vector to be set to zero since in the first case an overflow would otherwise result and in the latter two cases highly inaccurate values would be obtained.

6. Level II (III) Operations on Complex (Real) Arrays

Vectors stored in real arrays are treated by the operations on Level III as if they were complex vectors whose real and imaginary parts were equal. Vectors stored in complex arrays are treated by the operations on Level II as if they consisted only of their imaginary parts; their real parts are ignored.

7. Non-existent Addresses, Banks, or Arrays

References to non-existent addresses, banks, or arrys are ignored. Thus if there is no location with the address Z in the current bank, the predicate Z in the instruction LOAD Z is ignored. Likewise, if there are no locations at all assigned to bank 9, of the current array, the bank change $ 9 is ignored and the current bank remains current. Similarly, if an array Z has not yet been defined by a Context operation on Levels IV or V, the instruction IV LOAD Z or V LOAD Z will be ignored, and the current array will remain current. If no array at all has been defined, so that there is no current array or current bank, all operations on Levels II and III are without effect.

8. Storage Space

Each console has 9500 words of mass memory storage space which the user may allocate has he chooses. For most users, it will be quite sufficient to assume that 1000 words will be used for storage of console programs and symbols, leaving 8500 words for data storage. A real array of m vectors each with n components requires m(n+1) words. A

complex array with the dimensions requires 2m(n+1) words. It is the user's responsibility in defining arrays to be sure that he does not exceed the storage limits.

A simple procedure for the beginner is just to define one complex array, of dimensions m = 26, n = 127. Then each vector has maximum length, the array fills one bank, and you may work freely on Levels II or III.

B. Descriptions of Individual Operations

This section contains descriptions of the operations which are brought about by pressing keys on the left keyboard. Wherever practicable the description of an operation θ is worded so as to be a direct answer to the question, "What happens when the θ key is pressed?" In the case of single predicate operations the description is necessarily an explanation of what happens after the predicate has been given by the right keyboard. In the case of continuing operations it is an explanation of what happens after the first predicate has been given, since the result of giving further predicates can easily be inferred from this.

The capital letters X, Y, U, and V will be used to denote vectors stored in the X, Y, U, and V registers, respectively. If the contents of the X or Y register are altered by an operation, X and Y will be used to denote the contents before the operation is executed, and X' and Y' will be used to denote the contents after it is executed.

Several operations use either the U or V register or both at some intermediate stage in the calculation. When it is desirable or convenient to make reference to the contents of one of these registers at such an intermediate stage, the notation U or V will be used (without a prime). If an explanation of such intermediate stages of the calculation does not seem worthwhile, the description will simply contain the assertion that the U or V register is altered. If no mention of a register is made, this means that the contents of the register are not altered.

The letter n is used consistently to denote the dimension of the vectors in the current array. Where the meaning seemed clear, in writing equations for the components X_j' or Y_j' the statement "j=1, . . . , n" has been omitted.

Add +

Level II: The predicate vector is loaded into the V register. Then V is added to Y and the sum is left in the Y register.

V = the specified vector

$$Y'_j = Y_j + V_j$$

Level III: A vector, specified by its address, is placed in the U and V registers. If the current array is complex, then the real part of the specified vector is placed in the U register and the imaginary part in the V register. If the current array is real, then the specified (real) vector is placed in both the U and the V registers. After the U and V registers are loaded U + iV is added to X + iY. The real and imaginary parts of the sum are placed in the X and Y registers, respectively.

$U + iV$ = the specified vector

$$X'_j = X_j + U_j$$
$$Y'_j = Y_j + V_j$$

Arctangent ARC TAN

Level II: Y is replaced by its Arctangent.

$$Y'_j = \text{Arctan } Y_j$$

The Arctangent function is defined such that

$$-\frac{\pi}{2} \leq Y'_j \leq \frac{\pi}{2} .$$

The contents of the V register are altered.

Level III: At present this operation has no effect.

Argument ARG

Level II: Y is replaced by Y' in accordance with the equation

$$Y'_j = \begin{cases} 0 & \text{if } Y_j \geq 0 \\ \pi & \text{if } Y_j < 0 \end{cases}$$

Level III: X + iY is replaced by its (real) argument θ:

$$X'_j = \theta_j$$

$$Y'_j = 0$$

Each component θ_j is the angle, in radians, in the polar decomposition, $X_j + iY_j = r_j e^{i\theta_j}$. These angles are made unique by the continuity requirement that successive values of θ differ by less than π.

Binary Scale MOD

Level IX: This displays the binary scale of the V register on the scope at the current carriage position.

The Y register is altered.

Decimal Value EVAL

Level IX: This displays the decimal value of the first value of the Y register on the scope at the current carriage position.

The Y register is altered.

Conjugate *

Level II: Y is replaced by its negative.

$$Y_j' = -Y_j \quad .$$

Level III: X + iY is replaced by its complex conjugate. This operation is actually the same as on Level II.

$$X_j' = X_j$$

$$Y_j' = -Y_j \quad .$$

Context CTX

Level IV: A real array is defined and space in the mass memory is allocated for it. The array defined becomes the "current array" and bank 1 becomes the "current bank."

A real array which will be designated by the letter A and which will have m vectors each with n components is defined by the sequence

IV CTX A jhl pqr

where jkl and pqr denote three digit specifications of m and n made by pressing numerical keys on the right keyboard (e.g., 052 120 specifies that m = 52 and n = 120). Any of the 26 letters of the alphabet may be used to designate the array, provided that that letter has not previously been used to designate another real or complex array. (If the same letter is used twice, all record of the first usage is lost.)

The numbers m and n determine the amount of space allocated for the array and determine the structure of addresses within it. If m is less than or equal to 26, then all the vectors to be stored in the array will be stored in

bank 1 and will be addressed by the first m letters of the alphabet. If m is greater than 26 but less than or equal to 52, there will be 26 vector storage locations available in bank 1, with addresses A through Z, and m-26 locations available in bank 2, addressed by the first m-26 letters of the alphabet. If m is greater than 52, higher numbered banks will be utilized as needed.

Level V: This operation is the same as the Context operation on level IV except that it defines a complex array, consisting of pairs of real vectors, instead of a real array.

Cosine COS

Level II: Y is replaced by cos Y, Y being in radian measure. If the scale of Y is greater than 26, Y is replaced by the zero vector.

$$Y'_j = \cos Y_j$$

The contents of the V register are altered.

Level III: X + iY is replaced by cos (X + iY)

$$X'_j = \cos X_j \cosh Y_j$$

$$Y'_j = \sin X_j \sinh Y_j$$

If the scale of X or Y is greater than 26, both X and Y are replaced by zero. X_j and Y_j are in radian measure.

The contents of the U and V registers are altered.

Delta Function δ

Level II: Y is replaced by $\delta(Y)$ where δ is the Kronecker-Dirac delta function.

$$Y'_j = \left\{ \begin{array}{l} 1 \text{ if } Y_j = 0 \text{ or } Y_j Y_{j+1} < 0 \\ \\ 0 \text{ otherwise} \end{array} \right\} \quad j=1,\ldots,n-1$$

$$Y'_n = \left\{ \begin{array}{l} 1 \text{ if } Y_n = 0 \\ \\ 0 \text{ if } Y_n \neq 0 \end{array} \right.$$

If n=1 only the latter equation applies.

Level III: X + iY is replaced by the real function $\delta(X)\,\delta(Y)$:

$$X'_j = \left\{ \begin{array}{l} 1 \text{ if } X_j = 0 \text{ or } X_j X_{j+1} < 0 \text{ and if } Y_j = 0 \text{ or } Y_j Y_{j+1} < 0 \\ \\ 0 \text{ otherwise} \end{array} \right.$$

$$Y'_j = 0$$

Display DISPLAY

Level I: This operation enables the user to "draw" a figure on the CRT, which may then be stored for subsequent display. Thus, special symbols can be created by the user and then used in the typing mode (Level IX Display) along with the letters and numbers provided, or any kind of diagram or figure can be drawn, stored, and redisplayed later.

The figure is formed on the CRT by adjoining small line segments (which we may call vectors), each vector starting at the end of the last vector, and going a short distance in one of 16 directions, each direction being specified by a key on the right keyboard. In this way, any symbol which can be drawn in one continuous curve, may be built up on the CRT.

In creating a symbol or figure, three steps are necessary: positioning the origin, positioning the starting point, and tracing the curve. These are explained below.

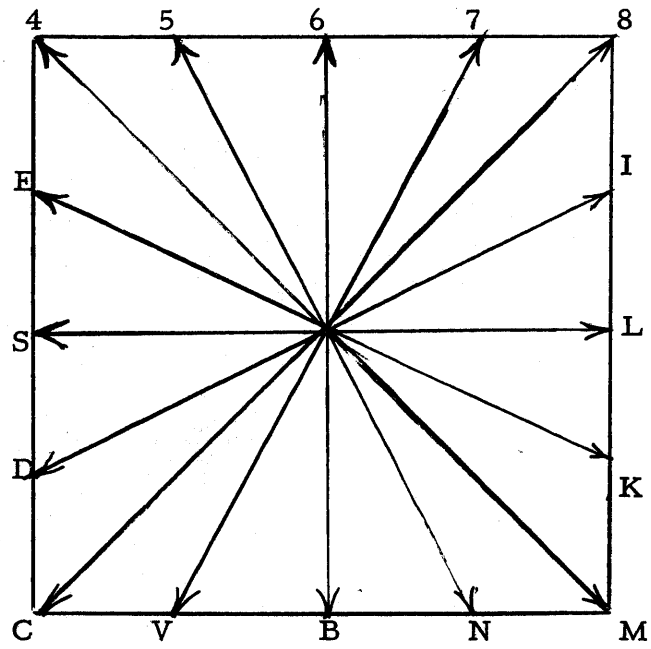
1. Positioning the Origin. In defining a symbol which is to be typed on the CRT in a text using Level IX display, the symbol may be thought of a being placed in a "frame". The face of the CRT is divided into 21 rows, (or lines of print) each row consisting of 25 frames. Each frame is a rectangular space which normally would contain a letter or some other "standard sized" symbol. The origin of a given frame is

always in the lower left hand corner of the frame. Pressing the (Level I) DISPLAY key selects the leftmost frame in the top row, and this is the standard frame which must be used for a symbol which is to be typed in a text later using the Level IX display.

However, another frame may be selected to start the symbol, if desired, by using the Space Bar, and the Carriage Return key. Pressing the Space Bar moves the origin to the next frame to the right, and pressing the Carriage Return key moves it to the leftmost frame of the next row below the current position.

2. Positioning the Starting Point. The starting point for the first vector in the symbol is accomplished by pressing two decimal digit keys, p, and q, with $0 \leq p \leq 9$, $0 \leq q \leq 9$. After these two keys are pressed, the starting point is illuminated on the CRT. The digits p and q determine the x and y coordinates of the starting point, relative to the origin of the frame selected. The frame is thought of as being 10 units wide, and 12 units high. The standard sized capital Roman letters are 8 units high, and are centered at (5, 5) relative to the origin. Thus the bottom of the letter rests on the line 1 unit above the bottom of the frame.

3. Tracing the Curve. This part of the operation consists in giving a sequence of vectors to define the curve. The first vector emanates from the starting point, and each succeeding vector starts at the end of the last vector. There are 16 possible directions provided for a vector, produced by depressing one of 16 keys on the right-hand keyboard, according to the pattern below.



Each key on the keyboard is considered to be 1 unit away from its neighboring keys. To determine the length and

and direction of the vector formed by depressing a key, count from the center "0" up(+) or down (-) for the y component, left(-) or right(+) for the x component. Thus the key "7" will result in a vector with x component +1, and y component +2 units. Similarly, the "c" key results in a vector with x component -2 and y component -2. Hence the keys may be used as a "compass" to direct the line segments in building up the curve. To take a specific example, the sequence of keys

I DISPLAY 3 1 6666 LLB

will produce the Greek capital letter, Γ, centered in the frame. The vectors start at the bottom of the symbol, the vertical line 8 units high being produced by the key pushes "6666". The top is then formed, 4 units long, by the keys "LL", and the serif is finished off by a downward stroke of 2 units given by the key "B". The starting point for the vertical line was taken to be (3, 1), which centers the symbol at (5, 5) to conform to the convention for the Roman letters.

If an error is made in tracing the curve, it can be corrected by depressing the Back Space key. This removes the last

vector from the list, and restarts at the end of the next-to-last vector. If an error is made in positioning the origin or the starting point, the Display key must be depressed, which wipes out the entire list, and starts the entire operation over.

After one curve is traced, the origin may be moved to another frame by pressing the carriage return key and space bar the appropriate number of times. Then steps 2 and 3 may be repeated to form another figure beginning in the new frame.

Level IX: The right keyboard is turned into a typewriter; the letters, numbers and other symbols subsequently typed appear on the CRT. The nine banks act as nine cases with bank 1 providing capital Roman letters and the decimal digits 0 through 9. Banks 5 through 9 provide symbols created by the Display operation and stored by the Store operation on Level I.

As with other level changes, pressing the IX key establishes bank 1 as the current bank. Bank changes are made in the usual way by pressing the bank change key $ followed by one of the digit keys, 1 through 9. This does not affect the CRT.

The Space Bar and Carriage Return operate in the same manner as on a conventional typewriter. The face of the CRT is 25 "spaces" wide and 21 lines high. When the right hand margin is reached, by typing a sequence of symbols and "spaces", a carriage return is automatically executed, so that the next symbol typed appears at the left hand margin on the next row. When Level IX Display is first initiated, typing starts in the upper left-hand corner of the CRT. When Level IX Display is initiated subsequently, typing is resumed at the place where it left off. Typing can be moved up a line or down a line by means of the Roll Up and Roll Down operations (+ and - on Level IX). Typing can be restarted in the upper left hand corner by means of the Restart operation(RS on Level IX). Note: Roll-up, Roll-down, Restart must be followed by Display.

Divide /

Level II: The predicate vector is loaded into the V register. Then Y is divided by V and the quotient is left in the Y register.

$$V = \text{the specified vector}$$

$$Y'_j = Y_j \; / \; V_j$$

Level III: A vector, specified by its address, is placed in the U and V registers. If the current array is complex, then the real part of the specified vector is placed in the U register and the imaginary part in the V register. If the current array is real, then the specified (real) vector is placed in both the U and the V registers. After the U and V registers are loaded X + iY is divided by U + iV. The real and imaginary parts of the quotient are placed in the X and Y registers, respectively.

$$U + iV = \text{the specified vector}$$

$$X'_j = (X_j U_j + Y_j V_j)/(U_j^2 + V_j^2)$$

$$Y' = (Y_j U_j - X_j V_j)/(U_j^2 + V_j^2)$$

ENTER

All levels: If ENTER occurs in a console program list, the console program will halt at that point and the computer returns to manual mode. If, after carrying out any desired operations in the manual mode, you push ENTER, the computer will return to the interrupted console program and proceed to execute the remainder of it. This provides the capability to have console programs interrupt, awaiting data from the user.

ERASE

All Levels: The image on the face of the CRT is erased.

Evaluate EVAL

Level II: The real function y(x) stored in the X and Y registers is evaluated at the points $U_1, \ldots, U_n$ of the predicate vector. If the latter is a constant vector with each component equal to a, the result left in the Y register will be a constant vector with each component equal to the function value $y(a)$.

The evaluation is done using linear interpolation, in the following way. First the vector specified by the predicate is loaded into the U register. Then each point U_j of U is examined in turn, and a pair of successive points X_k, X_{k+1} of X is sought such that U_j lies between X_k and X_{k+1}. If there is no such pair, the resulting value V_j is set equal to zero. If there is such a pair, the function values Y_k and Y_{k+1} are interpolated linearly to determine the value V_j of y at U_j:

$$V_j = Y_k + \frac{U_j - X_k}{X_{k+1} - X_k} (Y_{k+1} - Y_k)$$

Note that k is a function of j.

The searching is done in the following way. Beginning at $j = 1$, and until a successful search has been made, the search is started with the pair X_1, X_2. After that, the search is

started with the last pair found and proceeds cyclically, the pair X_1, X_2 following X_{n-1}, X_n.

After the last component V_n has been calculated, U is transferred to the X register and V is transferred to the Y register.

$Y' = V$ (given above)

$X' = U$ the predicate vector

Level III: At present this operation has no effect.

Exponentiate EXP

Level II: Y is replaced by e^{Y}:

$$Y_j' = \exp Y_j$$

Warning: If any component of Y is greater than 2^{25}, then every component of Y is replaced by zero.

Level III: X + iY is replaced by $e^{(X + iY)}$

$$X_j' = \exp X_j \cos Y_j$$

$$Y_j' = \exp X_j \sin Y_j$$

Warning: If any component of either X or Y is greater than 2^{25} then every component of both X and Y is replaced by zero.

FORWARD DIFFERENCE Δ

Level II: Each component of Y is replaced by the corresponding first forward difference, the difference for the last component being extrapolated linearly from the previous two differences. An exception is made if n = 1 or 2.

$$\left.\begin{aligned} Y'_j &= Y_{j+1} - Y_j, \qquad j=1,\ldots,n-1 \\ Y'_n &= 2Y'_{n-1} - Y'_{n-2} = 2Y_n - 3Y_{n-1} + Y_{n-2} \end{aligned}\right\} \text{ if } n > 2$$

$$Y'_1 = 0 \qquad \text{if } n = 1$$

$$Y'_1 = Y'_2 = Y_2 - Y_1 \qquad \text{if } n = 2$$

Level III: The first forward difference is taken of both the X and Y vectors as on Level II.

$$\left.\begin{aligned} X'_j &= X_{j+1} - X_j \\ Y'_j &= Y_{j+1} - Y_j \end{aligned}\right\} \quad j=1,\ldots,n-1 \quad \text{if } n > 2$$

$$\left.\begin{aligned} X'_n &= 2X_n - 3X_{n-1} + X_{n-2} \\ Y'_n &= 2Y_n - 3Y_{n-1} + Y_{n-2} \end{aligned}\right\} \quad \text{if } n > 2$$

$$X'_1 = Y'_1 = 0 \qquad \text{if } n = 1$$

$$\left.\begin{aligned} X'_1 &= X'_2 = X_2 - X_1 \\ Y'_1 &= Y'_2 = Y_2 - Y_1 \end{aligned}\right\} \quad \text{if } n = 2$$

IDENTITY ID

Level II: The identity function, $y = x$, at n equally spaced points on the interval (-1, 1) is generated and placed in the X and Y registers.

$$X'_j = Y'_j = (2j-n-1)/(n-1) \quad \text{if } n > 1$$

$$X'_1 = Y'_1 = 0 \qquad \text{if } n = 1$$

Level III: Same as Level II.

Inverse INV

Level II: Each non-zero component of Y is replaced by its reciprocal. Each zero component is left unchanged.

$$Y'_j = \begin{cases} Y_j^{-1} & \text{if } Y_j \neq 0 \\ 0 & \text{if } Y_j = 0 \end{cases}$$

Level III: Each non-zero component of the complex vector X + iY is replaced by its complex reciprocal. Each zero component is left unchanged.

$$\left.\begin{aligned} X'_j &= X_j/(X_j^2 + Y_j^2) \\ Y'_j &= -Y_j/(Y_j^2 = Y_j^2) \end{aligned}\right\} \quad \text{if } X_j^2 + Y_j^2 \neq 0$$

$$X'_j = Y'_j = 0 \qquad \text{if } X_j^2 + Y_j^2 = 0$$

Left Shift LS

Level II: A circular left shift is performed on the components of Y; i.e. the first component of Y is moved to the last position and all other components are shifted one position to the left.

$$Y'_j = Y_{j+1}\ , \qquad j=1,\ldots,n-1$$

$$Y'_n = Y_1$$

Level III: A circular left shift is performed on the components of both X and Y.

$$X'_j = X_{j+1}, \qquad j=1,\ldots,n-1$$

$$X'_n = X_1$$

$$Y'_j = Y_{k+1}, \qquad j=1,\ldots,n-1$$

$$Y_n = Y_1$$

LIST

All Levels: The computer is removed from the manual mode and is placed in the console programming mode. A list is made in the Y register of the keys, on subsequent keyboards, subsequently pressed. The corresponding operations are not executed, but their names are displayed on the scope. The list is terminated by pressing the List key again, and the computer is returned to manual control. This list may then be assigned to a variable key on the left keyboard on one of the user levels XI through XIX by means of the Store operation on that level. Thereafter whenever the key associated with this list is pressed on that level, the sequence of operations indicated will be executed.

The list should contain all level and bank changes necessary to specify the operations desired. In particular, the list should begin with the level number of the first operation to be performed unless that level is the same as the level on which the list is to be stored. The list should generally end with the level number of the level on which the list is to be stored. This is necessary if the operation is to be repeatable by means of the Repeat operation and is

usually convenient in any case. Thus, the sequence of key pushes at the end of a console program which is to be stored in + on level XII, say, would be

XII LIST XII STORE +

where the first XII is in the list, and hence causes no level change to occur now. The second XII comes after termination of the list, and therefore makes XII the current level so that STORE + stores the list in location + on level XII. Alternatively, you could call in level XII before starting to make the list, in which case the following of the list would be unnecessary.

Maximum MAX

Level II: The maximum component of Y is found and every component of Y is replaced by it.

$$Y'_j = \max (Y_1, \ldots, Y_n), \qquad j=1, \ldots, n$$

Level III: The maximum component of Y is found and every component of Y is replaced by it. Every component of X is replaced by the component of X having the same index as the Y component found. If two or more components of Y are equal to the maximum component, the one having the smaller or smallest index is chosen.

$$\left.\begin{aligned} X'_j &= X_k \\ Y'_j &= Y_k \end{aligned}\right\} \quad j=1, \ldots, n,$$

where k is the smallest integer such that $Y_k = \max (Y_1, \ldots, Y_n)$.

Modulus MOD

Level II: Y is replaced by its absolute value

$$Y'_j = |Y_j|$$

Level III: X + iY is replaced by its (real) modulus.

$$X'_j = \sqrt{X_j^2 + Y_j^2}$$

$$Y'_j = 0$$

The contents of the V register are altered.

MULTIPLY

Level II: The predicate vector is loaded into the V register. Then Y is multiplied by V and the product is left in the Y register.

V = the specified vector

$Y'_j = Y_j \cdot V_j$

Level III: A vector, specified by its address, is placed in the U and V registers. If the current array is complex, then the real part of the specified vector is placed in the U register and the imaginary part in the V register. If the current array is real, then the specified (real) vector is placed in both the U and the V registers. After the U and V registers are loaded X + iY is multiplied by U + iV. The real and imaginary parts of the product are placed in the X and Y registers, respectively.

$U + iV$ = the specified vector

$X'_j = X_j U_j - Y_j V_j$

$Y'_j = X_j V_j + Y_j U_j$

Natural Logarithm LOG

Level II: Each positive component of Y is replaced by its natural logarithm. Each non-positive component of Y is replaced by zero.

$$Y'_j \begin{cases} \log_e Y_j & \text{if } Y_j > 0 \\ 0 & \text{if } Y_j \leqslant 0 \end{cases}$$

The contents of the V register are altered.

Level III: Each non-zero component of X + iY is replaced by its logarithm. Each zero component ($X_j = Y_j = 0$) is left unaltered.

$$\left.\begin{array}{l} X'_j = \frac{1}{2} \log (X_j^2 + Y_j^2) \\ \\ Y'_j = \arg (X_j + iY_j) \end{array}\right\} \text{ unless } X_j = Y_j = 0$$

For a detailed description of the function arg(z) see the description of the Argument operation

RESET

All Levels: The execution of any console program currently being run is halted at the next basic subroutine, and the computer is returned to manual control. The current level, bank and array are left unchanged.

Load LOAD

Level II: The predicate operand is put into the Y register. If the current array is real, and an address is specified, then the vector at that address is transferred to the Y register; if the current array is complex, then the imaginary part of the vector is put into the Y register. If the predicate is specified by giving, on the numerical keys, the decimal representation of a real number, a, then a vector is generated in the Y register each of whose components is equal to a. The constant vector thus generated has the same number of components as the vectors within the current array.

Level III: The predicate operand is put into the X and Y registers. If the current array is real and an address is specified, then the complex vector at that address replaces X + iY. If the current array is real, both X and Y are replaced by the specified vector. If the predicate is specified by giving the decimal representation of a real number LOAD operates in the same fashion as LOAD on Level II. Thus, to generate the constant complex vector a + ib, we

III LOAD a REFL LOAD b

where a and b are the appropriate decimal representations.

Level IV: The designated real array becomes the current array and bank 1 becomes the current bank. The format is

IV LOAD A,

where A stands for the letter designating the new real array.

Level V: The designated complex array becomes the current array and bank 1 becomes the current bank. The format is

V LOAD B,

where B stands for the letter designating the new complex array.

Reflect REFL

Level II: The Y vector is reflected about its midpoint, so that the first component becomes the last, the second component becomes the next to last, etc.

$$Y'_j = Y_{n+1-j}$$

Level III: The X and Y vectors are interchanged. This corresponds to a reflection about the 45° line in the complex plane.

$$X'_j = Y_j$$

$$Y'_j = X_j$$

REPEAT

All Levels: The instruction REPEAT θ $d_1d_2d_3$ causes the operation θ on the current level to be executed k times, where k is the positive integer represented by the digits $d_1d_2d_3$. In effect the Repeat operation presses the θ key k times.

The Repeat operation may be part of a console program, and the operation θ may be a console program. However, if θ is a console program it must end on the same level it is stored on. Otherwise, the second time the θ key occurs within the console program list it will be interpreted by the computer to be that key of a different level.

Right Shift RS

Level II: A circular right shift is performed on the components of Y; i.e. the last component of Y is moved to the first position and all other components are shifted one position to the right.

$$Y'_1 = Y_n$$

$$Y'_j = Y_{j-1}, \quad j=2,\ldots,n$$

Level III: A circular right shift is performed on the components of both X and Y.

$$X'_1 = X_n$$

$$X'_j = X_{j-1}, \quad j=2,\ldots,n$$

$$Y'_1 = Y_n$$

$$Y'_j = Y_{j-1}, \quad j=2,\ldots,n$$

Roll Up +

Level IX: The typewriter carriage is rolled up one line. The symbols subsequently typed will appear on the CRT one line above where they would have otherwise. Roll up must be followed by Display.

Roll Down -

Level IX: The typewriter carriage is rolled down one line. The symbols subsequently typed will appear on the CRT one line above where they would have otherwise. Roll Down must be followed by Display.

Restart RS

Level IX: Typing is restarted at the upper left-hand corner of the CRT. Restart must be followed by Display.

Sine SIN

Level II: Y is replaced by sin Y, Y being in radian measure. If the scale of Y is greater than 26, Y is replaced by the zero vector.

$$Y'_j = \sin Y_j$$

The contents of the V register are altered.

Level III: X + iY is replaced by sin (X + iY):

$$X'_j = \sin X_j \cosh Y_j$$

$$Y'_j = \cos X_j \sinh Y_j$$

X_j and Y_j are in radian measure. If the scale of X or Y is greater than 26, both X and Y are replaced by the zero vector.

The contents of the U and V registers are altered.

Square SQ

Level II: Y is replaced by its square.

$$Y'_j = Y_j^2$$

Level III: X + iY is replaced by its square,

$$X'_j = X_j^2 - Y_j^2$$

$$Y'_j = 2X_j\ Y_j$$

The contents of the U and V register are altered.

Square Root SQRT

Level II: Each non-negative component of Y is replaced by its non-negative square root. Each negative component of Y is replaced by zero.

$$Y'_j = \begin{cases} \sqrt{Y_j} & \text{if } Y_j \geqslant 0 \\ 0 & \text{if } Y_j < 0 \end{cases}$$

The contents of the V register are altered.

Level III: X + iY is replaced by its square root,

$$X'_j + iY'_j = \sqrt{X_j + iY_j}$$

where the square root function $\sqrt{z}$ is made unique by assuming the z plane cut along the negative real axis for j=1 and thereafter staying on a single Rieman sheet. [An equivalent description is the following:

1) $X'_1 \geqslant 0$ and (sign of Y'_1) = sign of Y_1

2) Of the two possible choices $X'_j + iY'_j$ for $j > 1$, differing by a factor $e^{i\pi}$, we choose the one closest to $X'_{j-1} + i\,Y'_{j-1}$.]

The contents of the U and V registers are altered.

Store STOR

Level I: A symbol or figure created in the Y register by means of the level I Display operation is stored in one of the banks 5 through 9 of the "typewriter." The key which is to "type" the symbol or figure is indicated by pressing that key after the Store key is pressed. It may be any of the alphanumeric keys, A through Z or O through 9. The bank number must be indicated, by pressing $ and then the appropriate digit key, before the Store key is pressed.

The Store operation adds the symbol or figure in the Y register to the repertory of symbols and figures which can be typed by means of the level IX Display operation.

Level II: The vector in the Y register is stored in the next specified address, if the current array is real. It is stored in the imaginary vector of the next specified address if the current array is complex.

Level III: The vector in the X + iY registers is stored in the next specified address if the current array is complex. If the current array is real, only the Y register contents are stored.

Levels XI through XIX: A list or console program list previously formed in the Y register by LIST is assigned to the variable key subsequently pressed on the operator keyboard and to the current level. The "variable" keys (i.e., the keys available for this purpose) are those which correspond in physical location to the letter keys on the operand keyboard, i.e., those shown as square in Fig.1. The list or console program in the Y register is formed by means of the List operation.

Substitute SUB

Levels II and III: On both levels II and III the Substitute operation is identical with the Load operation on level II except that the predicate vector is put into the X register instead of the Y register, and the Y register is left unchanged. This operation provides the capability for composing a complex vector from two real ones or, more generally, for making cross plots of any two variables. As with LOAD, the predicate operand may be a stored vector, specified by an address, or a constant vector specified by the numerical keys.

Subtract —

Level II: The predicate vector is loaded into the V register. Then V is subtracted from Y and the difference is left in the Y register.

$$V = \text{the specified vector}$$

$$Y'_j = Y_j - V_j$$

Level III: A vector, specified by its address, is placed in the U and V registers. If the current array is complex, then the real part of the specified vector is placed in the U register and the imaginary part in the V register. If the current array is real, then the specified (real) vector is placed in both the U and the V registers. After the U and V registers are loaded U + iV is subtracted from X + iY. The real and imaginary parts of the difference are placed in the X and Y registers, respectively.

$$U + iV = \text{the specified vector}$$

$$X'_j = X_j - U_j$$

$$Y'_j = Y_j - V_j$$

SUM Σ

Level II: Y is replaced by the running sum of Y:

$$Y'_j = \sum_{k=1}^{j} Y_k ,$$

Note that the first component remains unchanged and the last component is replaced by the sum of all the components.

Level III: Both X and Y are summed as on Level II.

$$X'_j = \sum_{k=1}^{j} X_k$$

$$Y'_j = \sum_{k=1}^{j} Y_k$$

TEST

All Levels: This operation is to be used only within a console program and provides the capability for making console programs branch, loop conditionally, etc. A test is made of the sign of the first component of the Y vector. The next operation θ in the console program is executed only if Y_1 is greater than or equal to zero. Otherwise θ is skipped.