9824-6001-RU-000

STL ON-LINE COMPUTER

Volume I — General Description

B. D. Fried

DECEMBER 28, 1964

PHYSICAL RESEARCH DIVISION

TRW SPACE TECHNOLOGY LABORATORIES

THOMPSON RAMO WOOLDRIDGE INC.

The STL On-Line Computer

Volume 1 - General Description

B. D. Fried

December 28, 1964

Second Edition - 12/31/64

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

TABLE OF CONTENTS

LIST OF TABLES

This is the first of a series of three volumes designed to acquaint users with the On-Line Center, a new facility which provides a convenient means of direct communication between a digital computer and the ultimate user, i.e., the originator of a problem. The present volume provides a general description of the facility. Volume 2, the Users' Manual, supplements this with a detailed description of the individual capabilities of the system. Volume 3, On-Line Techniques, gives some specific suggestions for use of the OLC in problem solving, based upon experience to date in a number of problem areas.

The programming structure of the OLC was designed by Glen Culler and implemented by Neil Waldhart, Gale Schluter, Len Messersmith, and Frank Jurkovich. The equipment used was built by the Bunker Ramo Corp.

A. When to Use the On-Line Center

The OLC provides a comfortable and convenient means of rapid communication with the computer. It is most useful, therefore, for problems requiring a significant interaction between user and computer. In many problems you need to experiment with different formulations, different methods of solution, or different methods of numerical analysis, and the OLC makes this kind of experimentation feasible. You may also find it useful for small problems of a more conventional nature, where the OLC will be faster than a hand computer, simply because it is electronic, but may also yield results faster than the 7094 just because it is so easy to program for small tasks.

Since it is a relatively small computer, compared to the 7094, the OLC will generally offer no advantage for any problem for which a satisfactory, checked-out 7094 program already exists. Moreover, for very large, highly structured problems, involving large amounts of machine time, the OLC may be useful in devising methods of solution, in exploring particular features, or perhaps in solving special cases. However, once a problem of this kind is well understood, so that a satisfactory 7094 program can be developed, the OLC will probably be of little help. A method of transferring programs, created and checked out on the OLC, to the 7094 for production running is planned, but has not yet been implemented.

B. Physical Description

The OLC has four identical user's consoles or stations. Each consists of two electric typewriter keyboards and a cathode ray tube oscilloscope. The four consoles operate simultaneously, each having control over approximately one quarter of the magnetic core and drum which comprise the computer memory, or storage. The arithmetic and control circuits of the computer are shared, in democratic fashion, by the four consoles in such a way that each user always has small requests serviced immediately, no matter how long or complicated a program is being executed for any other user. The other input/output capabilities include a Calcomp plotter, an output writer (i.e, the typing part of the typewriter) and two magnetic tape units. These, also, are shared by the four consoles, but, by their very nature, do not require, and hence do not always have, the same character of immediate response.

C. User Control and Computer Feedback

Control of the computer is accomplished with the keyboards. Of the two associated with a console, one is designated as the "operator keyboard", the other as the "operand keyboard". Immediate computer feedback is provided through the oscilloscope.

1. The operator keyboard simply provides a convenient way of initiating computer subroutines. There is a one-to-one correspondence between operator keys and computer subroutines: pressing a given key causes the computer to execute a particular program. Notice particularly that it is not necessary to use more than one of the operator keys; for example, to take the square root of a number or a function you push the single key labeled SQRT; you do not type the 10 keys which spell out the name of that operation, or even the four keys, S Q R T, which spell its abbreviation. For convenience, the keys of the operator keyboard are labeled with the names of, or symbols for, the operations which they initiate, as shown in Fig. 1. As explained in more detail below, the basic subroutines of the OLC provide the fundamental operations of mathematics, data processing and logic.

2. The operand keyboard provides a convenient way to store data in the computer's mass memory and to retrieve it. The operand keys consist of numerical, alphabetic and a few special keys as shown in Figure 2. The numerical keys are used for entering numerical data and for a few other purposes. The alphabetic keys are used for addressing data, i.e., the letters A through Z in effect constitute the "addresses" of data, so

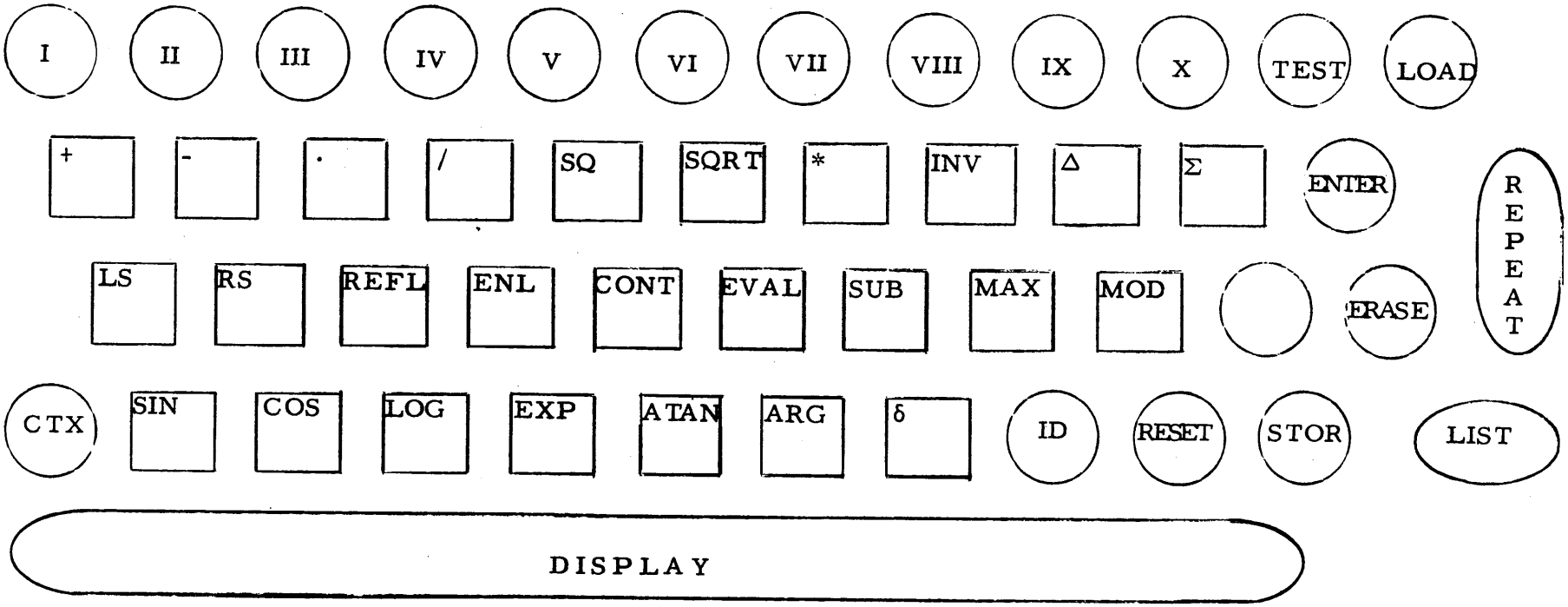


Figure 1. OPERATOR KEYBOARD

"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).

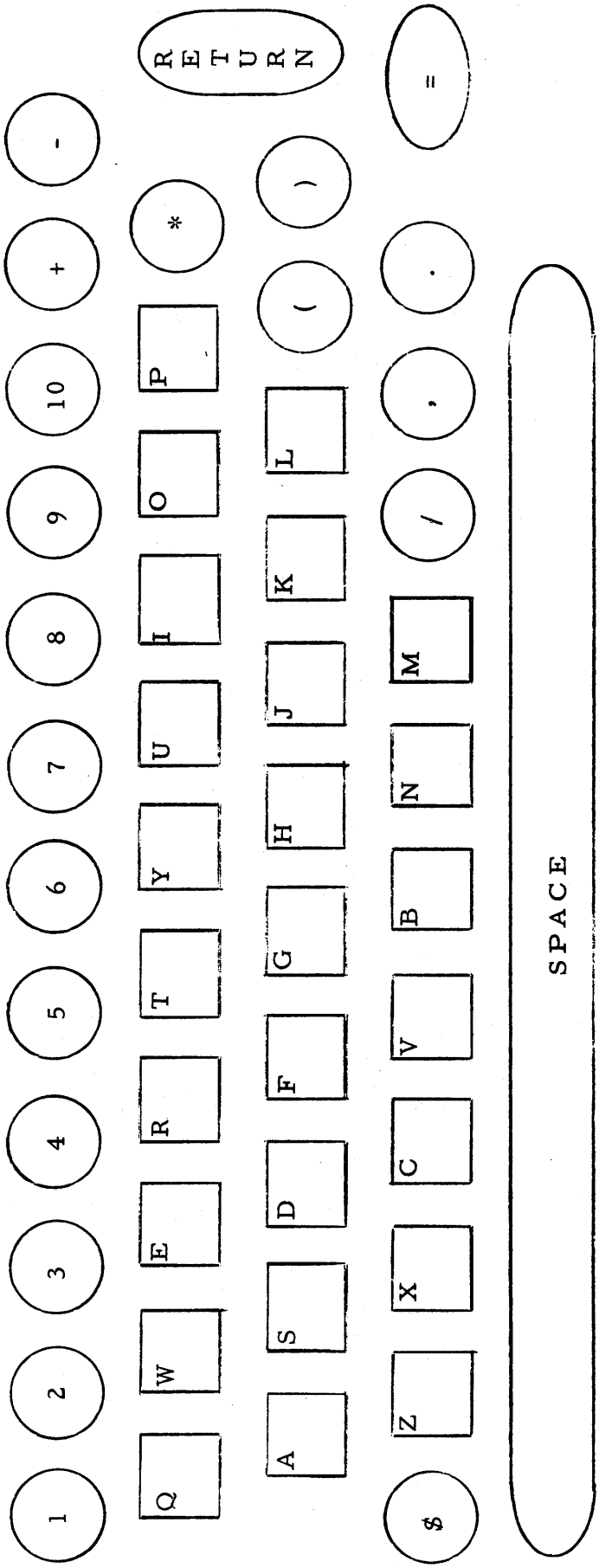


Figure 2. OPERAND KEYBOARD

far as the user is concerned. Thus, we can, and shall, talk about "location A" or "location H", but need never be concerned as to where in the mass memory (e.g., magnetic drum or disc) the data in question is actually stored. Henceforth, the term address for data will simply mean one of the 26 alphabetic keys of the operand keyboard. Specifying an address will mean pushing one of those keys.

As with all computers, data to be operated on is brought from the mass memory into the fast (magnetic core) memory and, after being transformed in the desired way, is returned to the mass memory. These basic data transfer operations are accomplished in the OLC system by two keys, labeled LOAD and STORE, on the operator keyboard. If you wish to work on (or examine) the data in location P, for example, you push the LOAD key on the operator keyboard, followed by the P key on the operand keyboard. You may then operate on this data, using any other keys of the operator keyboard. Finally, to store the result in some other location, say B, you push STORE on the operator keyboard, followed by B on the operand keyboard; or in an evident notation, you push STORE B. The previous contents of B are replaced with the new results. The data originally in P remains unchanged.

3. The oscilloscope allows the computer to feedback information to the user in graphical form (as curves, plots, diagrams, etc.), or in alphanumeric form (numerical vlaues, symbols, English language messages, etc.). These are described in more detail below. Like all activities of the computer, however, this feedback occurs only under user control; i.e., among the operations which you can initiate by pushing keys on the operator and operand keyboards are those which initiate these various kinds of display.

D. Organization of Data; Vectors and Arrays

The nature of the "data" referred to above is left to the choice of the user. In particular, you may wish them to be just single numbers, in which case the OLC will, essentially, function as a fancy hand computer and plotter, albeit with a much larger repertoire of operations than the usual desk variety. It is generally advantageous, however, to choose as operands not single numbers but rather a set, or list, of numbers, for in almost all mathematical and scientific applications, and in much of data processing as well, we wish a given operation to be carried out not on a single number but on a whole set. In mathematical terms, such a list of numbers is called a vector, and that is the term we shall generally adopt here. It is equally correct, and, in some applications very useful, to think of the numbers as constituting the values of the dependent variable of a real function, so we shall regard the two terms as synonomous. A real vector with n components is a list of n real numbers. A complex vector with n components is a list of n complex numbers, or equivalently, a two-column list, with n entries in each column.

You may choose the number of components of the vector (or number of entries in the list) to be anything from n = 1 to n = 127. With n = 127, you get a computer representation of a function which is satisfactory for a large fraction of problems. However, in some cases, both in scientific and data processing applications, one would like to deal with larger blocks of data. For this reason, the OLC provides for the use of arrays. An array is just a collection of m vectors where m, like n, is chosen by you,

subject only to the limitations imposed by the size of the portion of mass memory allocated to one console. The properties and uses of arrays are discussed in more detail in sections G and H.

Organization of the data into such blocks has two benefits. From a practical point of view, it allows you to initiate a good deal of computing with a very small effort, in terms of the number of keys pushed. Thus, if vectors have previously been stored in locations A and B, then the 6 key pushes: LOAD A + B STORE C cause the following things to happen. The vectors stored in locations A and B are brought into the fast memory; n additions are performed; and the resulting vector is stored in location C. A far more important feature for mathematical, scientific, and engineering applications is that this organization into vectors or functions reproduces the very structure of mathematics itself. As a result, the operations you carry out at the OLC have a very close correspondence with those used in the mathematical analysis required to set up or solve a problem, as will become apparent a little later.

E. Console Programming

As with any computer, it is necessary to have some means of constructing new programs or subroutines (beyond those initially furnished to the user) or modifying those previously created. It is here that the OLC differs most profoundly from conventional methods of computer organization, for it is just in the programming procedure that the user loses the ability to interact, in any direct fashion, with the computer. The labor involved in writing programs, checking them out, and making them correctly operational is sufficiently great that experimentation, involving unanticipated programming changes, is generally very difficult. The OLC provides a programming procedure which can be carried out directly, by the user at the console, and checked immediately. There is a minimum of bookkeeping work and no prior experience with computer programming is required.

A program consists simply of a list* of operations to be performed. The kind of operation sketched in the last two sections, which we may denote as the "manual mode", involves pushing a sequence of keys, from both the operator and operand keyboards, which cause the computer to carry out the desired operations on the indicated data. Clearly, then, we need only a convenient procedure for constructing, and subsequently storing, a list of such key pushes, together with some means of causing the computer to execute this list of key pushes at any later time.

*This list of operations should not be confused with the data lists, or vectors, discussed in section D. Henceforth, we will use the term "list" to denote a list of operations.

This capability is provided by the LIST operation on the operator keyboard, which works as follows. You push LIST and then push any sequence of keys on either keyboard, just as though you were operating the computer in the manual mode. However, instead of responding to these key pushes in the usual way, i.e. by initiating the corresponding subroutines, the computer simply records, and also displays on the oscilloscope, a list of the keys you push. When you push LIST a second time, it terminates the list and returns to the manual mode. If you now push STORE, followed by some key of the operator keyboard, the list you have constructed will be stored there.* If, at any later time, you push that key of the operator keyboard, the computer will execute the list of operations, just as though you were pushing the component keys manually.

The program thus created, which we shall call a "console program", now becomes a part of the OLC so far as your use is concerned, and can be initiated with a key push in exactly the same way as any of the basic subroutines originally supplied. Note particularly that this console program can itself constitute one of the components of a subsequent console program; i.e. after pushing LIST you may push any key, including those which initiate other console programs. Given this pyramiding feature, which is of course a vital ingredient of any computer, it is possible to construct programs of virtually unlimited complexity.

---

* Lists may be stored in any of the 26 keys shown as square in Figure 1, i.e., those which correspond to the alphabetic keys of the operand keyboard, Figure 2.

In addition to executing any previously constructed console program, P, you may also cause the computer to display, on the scope, the list of key pushes which comprise P, by pushing DISPLAY followed by the P key (of the operator keyboard).

F. Operator Levels and Systems

The 48 keys physically present on the operator keyboard are not sufficient to accommodate the necessary repertoire of basic subroutines, let alone the console programs constructed by a user. The totality of operators in the OLC are therefore grouped into "levels", those of a given mathematical or logical character being kept on one level. All of the basic subroutines are located on Levels I through IX. The level desired at any time is selected by pushing one of the keys, labeled I through IX, in the top row of the operator keyboard. Once a level is chosen, all subsequent key pushes are interpreted by the computer as belonging to that level (which we will term the "current level"), until a different level changing key is pushed by the user. These nine levels comprise what is called the Basic System.

An additional nine levels, all of whose keys are initially null operators, are provided to accommodate the console programs created by a user. Access to these is provided by first pushing the X key, followed by one of the keys I through IX; one may think of these as comprising Levels XI through XIX, or, alternatively, as being Levels I through IX of the User System. After creating a list, as described in the previous section, and before storing it by pushing some operator key, you must go to one of the user levels, since console programs may not be stored in the Basic System.

A change in level alters the meaning of all keys on the operator keyboard, save for a few, like LIST, which are common to all levels. The level changes are thus somewhat analogous to the shift operation on a normal typewriter, save that we have 18 different levels, rather than just two. Note that the one-to-one correspondence between operations of the system and keys of the operand keyboard, mentioned in section C, is valid only within a given level. A single physical key may initiate any one of 18 different subroutines, depending on the identity of the current level.

As you work with the OLC, you will build up, in your User System, a computing capability tailored to your own problems needs and interests, and expressed in a problem-oriented language created by you in the process of console programming. The User System thus created is a private one; whenever you leave the OLC, the contents of your part of the computer--the data lists you have stored and the operation lists or console programs you have created--are stored on magnetic tape. When you return, this data is loaded back into your console's portion of the computer, restoring it to the same state as when you left. In the interim, other people will use this console, or other ones, to create their own User Systems, but these need have no intersection with yours, save for the basic subroutines common to all users.

G. Data Levels and Arrays

In using the OLC you have the freedom to partition your share of the mass memory in any convenient fashion. If you wish to have storage space for m vectors, each with n components, you simply define an array with the dimensions m by n. Just as with the individual vectors, we use the letters A through Z to label the arrays. The CONTEXT key on the operator keyboard is used in defining arrays, the format being:

CONTEXT A jkl pqr

where A is the name chosen for the array in question, and jkl and pqr are three-digit specifications (typed in on the numerical keys of the operand keyboard) for m and n, respectively. The computer will thereupon allocate the necessary space in mass memory to accommodate m vectors, each having n components.

If m is less than 26, the addresses for the vectors constituting this array will be the first m letters of the alphabet, i.e., you can store vectors in, or lead vectors from, the first m keys, A, B, ... of the operand keyboard. If m is greater than 26, then it is necessary to introduce the concept of "data levels" ; these are completely analogous to the operator levels of the operator keyboard, although, to save confusion, we shall refer to levels on the operand keyboard as data "banks". Thus, if m is 52, for example, then its constituent vectors will have addresses A through Z on banks 1 and 2. Bank changes are

initiated by pushing the $ key on the operand keyboard, followed by the appropriate numerical key, 1 through 9. Just as with the operator levels, once a bank has been indicated it remains the "current" bank until some other bank is selected. After you have pushed $1, the computer will interpret all addresses as vectors on bank 1. To address a vector in location H on bank 2, you push $2 before pushing H. Thereafter, all addresses will be interpreted as being on bank 2, unless and until such time as you again change banks.

*In a similar fashion, you can work with as many as 26 different arrays. A particular array is designated by using LOAD on an array level (Level IV for real arrays, Level V for complex arrays). For example, the sequence

IV LOAD G II LOAD A + B IV LOAD K II $2 STORE C

results in the addition of the vectors in locations A and B on bank 1 of array G, the results being stored in location C on bank 2 of array K. As with levels and data banks, an array remains "current" until changed by the user.

New users will usually find it sufficient to work with a single array. In order to accommodate both real and complex vectors, of any size up to the maximum of 127, you could simply define, using CONTEXT on Level V, an array of dimension 26 by 127. This would provide room for one bank of complex vectors, which will suffice for simple problems. As you acquire more familiarity with the system, you can take advantage of the

*This paragraph should be omitted in a first reading.

array capabilities to make more efficient use of the storage space, e.g. by defining one real array and one complex one, since it is of course wasteful to use complex arrays for storing real vectors.

Note especially that you have not only the freedom to define one array (or more) in which your vectors can be stored; you have an obligation to do so, for until you have defined at least one array the computer will ignore any instructions you may give it to store a vector. Thus, when you first use the OLC, begin by defining at least one real or complex array.

H. The Basic System

We give a brief description of the capabilities available within the Basic System, although we defer details to Volume 2. We organize the discussion by levels:

Level I. Logical Operations

This level provides the elementary logical and data processing operations, for example, the capability to address any portion of the vector registers (defined in the next paragraph) including the scales. Also, a basic display capability is available, which allows you to create, in on-line fashion, on the oscilloscope, any desired symbol or other graphical display, and to store this in a "symbol table". In particular, you can construct in this way new "cases" or "fonts" of type and use these in the typing operations of Level IX, described below.

Level II. Real Vector Operations

These operators provide the basic facility needed to carry out the processes of real variable analysis on the computer. They have a sufficiently basic role within the system that we have elected to use their names or symbols in physically labelling the keys of the operand keyboard. Their characterization is facilitated by introducing the concept of registers. A register is simply a group of 128 memory locations in the fast (core) memory of the computer, used for temporary storage of a vector. Each console in the OLC has four registers, which we call the X,

Y, U, and V registers. (X and V are the principal registers; U and V are auxiliary registers.) Whenever a vector is brought from the mass memory into the fast memory to be worked on in some way, it goes into one of these registers. Likewise, the results of any calculations are left in one of these registers, and the STORE operator transfers a vector from a register to the designated storage location in the mass memory.

Some of the operators of Level II are predicate free. They work on the contents of the Y register, and leave the results there. For example, when SQRT is pushed, the computer takes the square root of each number in the Y register and leaves the result in the Y register. In symbolic form, we describe the action of SQRT as:

$$Y_j' = \sqrt{Y_j}$$

where Y_j denotes a typical element of the Y register before the operation and Y_j' denotes the same element afterward, with $j=1, 2, \ldots, n$. In similar fashion, we can characterize the other predicate free operations as shown in Table 1a. Like an intransitive verb, a predicate free operator requires no additional information since the data to be used is assumed to already be in the Y register. Therefore, the response is immediate and the operation is initiated as soon as the operator key has been pushed.

The other operations of Level II we call single predicate operators. Most are of a binary character, requiring at least two operands. For example, addition of functions requires two addends. In such cases, one

operand is assumed to be the contents of the Y register, and one more, the predicate operand, is specified by the user. If the operator is followed by specification of an address, then the predicate operand is just the vector stored at that location. If the operator is instead followed by numerical keys, then the predicate operand is a constant vector, with each component equal to the designated number.

When you push +, for example, nothing happens. However, as soon as you specify an address (i.e., some alphabetic key of the operand keyboard), say A, then the vector in location A is added to the vector in the Y register and the resulting sum is left in the Y register. (To be more precise, the vector in A is first transferred into the U register, and then the contents of the U register are added to those of the Y register. The auxiliary registers are typically used in this fashion by single predicate operators, but this is generally irrelevant to the user, who need only consider the X and Y registers and can, except in very special circumstances, ignore the existence of the U and V registers.)

A brief characterization of the other single predicate operators of Level II is given in Table 1b. Most are familiar from mathematics, but a few deserve special comment:

EVALUATE uses linear interpolation to give the values of a function at any desired values of the independent variable lying within the domain over which the function is initially known.

DISPLAY puts on the scope the curve obtained by plotting the predicate operand vector vs the vector in the X register.

SUBSTITUTE allows for cross plotting of any two vectors.

ID creates the "identity" function or vector, i.e, n equally spaced points on the interval from -1 to = +1.

δ is the computer version of a "Dirac delta function" defined to be 1 at a zero, or zero crossing, of the Y register, and 0 everywhere else.

ENLARGE and CONTRACT change the appearance of a vector when it is displayed.

All of the single predicate operators have a continuing character, in that they will act on any number of operands for which addresses are specified. For example, to form the sum of the four vectors in A, B, C, and D, it suffices to do

LOAD A + B C D

Repetition of + is not required. (However, if several numerical operands are specified in succession, the carriage return key must be used between them.) Like transitive verbs, single predicate operators require the specification of additional information of some kind. Therefore, the action is not initiated by the operator key itself but by the key or keys following it which specify the predicate operand.

Level III. Complex Vector Operations

These operators are the analogue, for complex vectors, of those on Level II. The X and Y registers, taken together, can be considered as a single complex register, whose contents we may denote as X + iY. The operators on Level III transform X + iY just as those of Level II transform Y. They are summarized in Table 2.

Level IV. Real Array Operations

This level provides the capability to do real variable analysis conveniently for functions represented by more than 127 points. The vectors of an array are here regarded as a kind of supervector: the first n components are just the n components of the first vector of the array; the next n components are the n components of the second vector of the array; etc.

As discussed further in section J, one binary scale is associated with each vector of an array. However, even if your vectors have 127 points or less, it is sometimes desirable to associate more than one scale with a single vector. If the vectors in a problem have 120 components, for example, and you wish to have a scale every 10th point, rather than one for the whole vector, then you simply work with arrays of dimensions 12 by 10, rather than with single vectors of 120 components.

The array operations are analogous to the Level II operations described in Table 1. In addition, this level has the CONTEXT operation, used in defining arrays, as explained in section G.

Level V. Complex Array Operations

This has the same relation to Level III as Level IV has to Level II.

Levels VI (Real Matrix Operations) and VII(Complex Matrix Operations), are not implemented at present, and Level VIII is deliberately left blank to accommodate elements of the basic system not yet planned for.

Level IX. Typing

This level allows you to compose, on the oscilloscope, alphanumeric messages of any kind, including any symbols defined via Level I. It also has the very important operators which show the binary scale of the Y register (see section J) or the decimal value of the first component of the Y register. DISPLAY on Level IX converts the operand keyboard into an ordinary typewriter, save that the messages appear on the oscilloscope. The $ key, followed by a numerical key, is used to change to different banks of the symbol table (e.g., those where symbols constructed on Level I have been stored). Carriage positioning operations are also available, as indicated in Table 3.

Using the capabilities of Levels I and IX, you can associate with any key on a user level (XI through XIX) any name or symbol you wish. As a result, when you display a console program (as described at the end of section F) those components of it which are themselves console programs, i.e. those which are key pushes on a user level, will be represented by the name you have assigned. (If you assigned none, then they will be represented by the standard symbols of Figs. 1 and 2.)

I. System Operations Common to All Levels

REPEAT allows you to repeat any operation, in the Basic System or User's System, any number of times. You simply call in the correct level, then push REPEAT, followed by the operator and (on the numerical keys) the number of times it is to be repeated.

RESET is used to stop a console program. When the computer comes to the next operation of the program it is running, it will be interrupted and will return to the manual mode.

TEST allows you to make a console program branch according to the sign of the first component of the Y register. It is only for use in a console program and operates as follows. If Y_1 is positive, or zero, then the next instruction following TEST in the console program list is carried out, in normal fashion. If Y_1 is negative, then the instruction immediately following TEST is ignored. Although TEST can be used anywhere in a console program, it is most conveniently used at the end, in order to make a conditional loop of some kind. For example, you might make TEST a part of a program, P, and have P conclude with the operators TEST P. If the result of P is such as to leave Y_1 non-negative, then P will be repeated. Otherwise, it will come to an end. Of course, if P is such that Y_1 never becomes negative, P will repeat itself indefinitely, but can be stopped with RESET.

ENTER is used only in a console program. When a console program reaches ENTER it will interrupt itself and return to manual mode, allowing

you to enter data or take other action (not including the running of some other console program). Upon completion of the manual operations, you may push ENTER and the console program will resume where it left off.

ERASE erases the oscilloscope. Since the OLC uses a storage scope, no partial erase of information is possible.

J. Scaling Considerations

The OLC uses a floating vector organization, whereby a vector with n components is actually represented in the machine as n numbers (the "mantissas"), all of magnitude less than 1, together with a single binary scale. Thus, if a_j are the mantissas representing the vector A and $S[A]$ is the binary scale, then the actual components of A are the numbers

$$A_j = a_j \cdot 2^{S[A]}$$

All of the operations of Level II and III deal with the scales in an appropriate manner. However, when a curve is displayed on the scope, it is really the mantissas which are used to form the display points and one can get information about the absolute magnitudes of the vector only by displaying also the binary scale (using the BiS operator of Level IX) as well as the curve displayed with the DISPLAY operator of Levels II or III. If a curve has scale S, then the limits of the display area on the scope are $\pm 2^S$. Note that if two curves are displayed simultaneously, correct conclusions concerning intersections, etc. can be deduced only if both have the same scale. If this is not already the case, it can be accomplished using ENLARGE or CONTRACT on Levels II or III.

All operations of Level II leave the result in the Y vector in "standard" form, meaning that at least one of the mantissas lies between 1/2 and 1, unless the vector is identically zero. Operations of Level III result in a "standard" form for the X or the Y register, or both; in addition, the scales of the two registers are made equal.

Table 1a

Predicate-free Operators of Level II

Operator	Action Taken*
SQUARE	$Y'_j = (Y_j)^2$
SQRT	$Y'_j = \sqrt{Y_j}$
*	$Y'_j = -Y_j$
INVERSE	$Y'_j = (Y_j)^{-1}$
Δ	$Y'_j = Y_{j+1} - Y_j$, $j=1, 2, \ldots, n-1$; $Y'_n = 2Y'_{n-1} - Y'_{n-2}$
Σ	$Y'_j = \sum_{k=1}^{j} Y_k$
LSHIFT	$Y'_j = Y_{j+1}$ $j=1, 2\ldots, n-1$; $Y'_n = Y_1$
RSHIFT	$Y'_j = Y_{j-1}$ $j=2, 3\ldots, n$; $Y'_1 = Y_n$
REFLECT	$Y'_j = Y_{n-j+1}$
ENLARGE	$Y'_j = \begin{cases} 2Y_j & \text{if } Y_j < 1/2 \\ 0 & \text{otherwise} \end{cases}$ $S[Y'] = S[Y] - 1$
CONTRACT	$Y'_j = \frac{1}{2} Y_j$ $S[Y'] = S[Y] + 1$
MAX	$Y'_j = \text{Max}(Y_1 \ldots . Y_n)$
MOD	$Y'_j = \lvert Y_j \rvert$

Table 1a (cont'd)

Operator	Action Taken*
SIN	$Y'_j = \sin Y_j$
COS	$Y'_j = \cos Y_j$
LOG	$Y'_j = \log_e Y_j$
EXP	$Y'_j = e^{Y_j}$
ARCTAN	$Y'_j = \text{ArcTan } Y_j, \ -\pi/2 < Y'_j < \pi/2$
ARG	$Y'_j = \begin{cases} 0 \text{ if } Y_j \geqslant 0 \\ \pi \text{ if } Y_j < 0 \end{cases}$
IDENTITY	$X'_j = Y'_j = 2\frac{j-1}{n-1} - 1$
δ	$Y'_j = \begin{cases} 1 \text{ if } Y_j = 0 \text{ or } Y_j Y_{j+1} < 0 \\ 0 \text{ otherwise} \end{cases}$

* Y_j is the value of the j^{th} element of the Y register before the operation; Y'_j is the value afterward. $S[Y]$ is the scale of the Y register. Similarly for X_j, X'_j, $S[X]$. Unless otherwise specified, $1 \leqslant j \leqslant n$. "A" is used to denote the predicate operand, i.e. the vector addressed immediately after the single predicate operator. It may also be a constant vector, entered by typing a number on the numerical keys instead of specifying an address. In that case, the computer decides that you are through entering the number and, accordingly, executes the operation in question as soon as you push any non-numerical key. Note that the reading (from core or drum) involved in LOAD, STORE, etc. is always non-destructive, i.e., just after the operation you have two copies of the data, one in the old location and one in the new.

Table 1b

Single Predicate Operators of Level II

Operator	Action Taken*
+	$Y_j' = Y_j + A_j$
-	$Y_j' = Y_j - A_j$
·	$Y_j' = Y_j \cdot A_j$
/	$Y_j' = Y_j / A_j$
LOAD	$Y_j' = A_j \quad A_j' = A_j$
STORE	$A_j' = Y_j \quad Y_j' = Y_j$
EVALUATE	$Y_j' = \frac{Y_{k+1} - Y_k}{X_{k+1} - X_k}(A_j - X_k) + Y_k$ where $k = k_j$ is such that $X_k \leqslant A_j \leqslant X_{k+1}$
SUBSTITUTE	$X_j' = A_j$
DISPLAY	$Y_j' = A_j$. The points (X_j, Y_j') connected by straight line segments, are plotted on the scope.

* Y_j is the value of the j^{th} element of the Y register before the operation; Y_j' is the value afterward. S Y is the scale of the Y register. Similarly for X_j, X_j', S X . Unless otherwise specified, $1 < j < n$. A is used to denote the predicate operand, i.e. the vector addressed immediately after the single predicate operator. It may also be a constant vector, entered by typing a number on the numerical keys instead of specifying an address. In that case, the computer decides that you are through entering the number and, accordingly, executes the operation in question as soon as you push any non-numerical key. Note that the reading (from core or drum) involved in LOAD, STORE, etc. is always non-destructive, i.e., just after the operation you have two copies of the data, one in the old location and one in the new.

Table 2a

Predicate Free Operators of Level III

Operator	Action Taken*
SQ	$X'_j + iY'_j = (X_j + iY_j)^2$
SQRT	$X'_j + Y'_j = \exp \frac{1}{2} \log (X_j + iY_j)$
*	$X'_j = X_j \quad Y'_j = -Y_j$
INV	$X'_j + iY'_j = (X_j + iY_j)^{-1}$
Δ	$X'_j = X_{j+1} - X_j$, $Y'_j = Y_{j+1} - Y_j$, $j=1, 2, \ldots, n-1$; $X'_n = 2X'_{n-1} - X'_{n-2}$, $Y'_n = 2Y'_{n-1} - Y'_{n-2}$
Σ	$X'_j = \sum_{k=1}^{j} X_j$, $Y'_j = \sum_{k=1}^{j} Y_j$
LSHIFT	$X'_j = X_{j+1}$, $Y'_j = Y_{j+1}$ $j=1, 2, \ldots, n-1$; $X'_n = X_1$, $Y'_n = Y_1$
RSHIFT	$X'_j = X_{j-1}$, $Y'_j = Y_{j-1}$, $j=2, 3, \ldots, n$; $X'_1 = X_n$, $Y'_1 = Y_n$
REFLECT	$X'_j = Y_j$, $Y'_j = X_j$

*See legend for Table 1. Here "A" denotes a complex predicated operand, if an address is specified following the operator, or a real, constant vector (with zero imaginary part) if the operand is specified via the numerical keys.

Table 2a (cont'd)

Operator	Action Taken*
ENLARGE	$X'_j = 2X_j$, $Y'_j = 2Y_j$; $S[X'] = S[X] - 1$, $S[Y'] = S[Y] - 1$
CONTRACT	$X'_j = \frac{1}{2} X_j$, $Y'_j = \frac{1}{2} Y_j$; $S[X'] = S[X] + 1$, $S[Y'] = S[Y] + 1$
MAX	$Y'_j = \text{Max}(Y_1 \ldots Y_n) = Y_j$; $X'_j = X_j$
MOD	$X'_j = \sqrt{(X_j)^2 + (Y_j)^2}$, $Y'_j = 0$
SIN	$X'_j + iY'_j = \sin(X_j + iY_j)$
COS	$X'_j + iY'_j = \cos(X_j + iY_j)$
LOG	$X'_j + iY'_j = \log_e(X_j + iY_j)$
EXP	$X'_j + iY'_j = e^{(X_j + iY_j)}$
ARG	$X_j = \theta_j$, $Y'_j = 0$ where $X_j + iY_j = R_j e^{i\theta_j}$
ID	$X'_j = Y'_j = 2\frac{j-1}{n-1} - 1$
δ	$X'_j = \delta(X)\,\delta(Y)$, $Y'_j = 0$

*See legend for Table 1.

Table 2b

Single Predicate Operators of Level III

Operator	Action Taken*
+	$X'_j + iY'_j = X_j + iY_j + A_j$
-	$X'_j + iY'_j = X_j + iY_j - A_j$
.	$X'_j + iY'_j = (X_j + iY_j) \cdot A_j$
/	$X'_j + iY'_j = (X_j + iY_j)/A_j$
LOAD	$X'_j + iY'_j = A_j$
STORE	$A'_j = X_j + iY_j$
SUBSTITUTE	$X'_j = A_j$
DISPLAY	The points (X'_j, Y'_j) are plotted on the scope and connected with straight lines, where $X'_j + iY'_j = A_j$.

*See legend for Table 1. Here A denotes a complex predicated operand; if an address is specified following the operator, or a real, constant vector (with zero imaginary part) if the operand is specified via the numerical keys.

Table 3

Operators of Level IX

Key	Action Taken
DISPLAY	Converts operand keyboard to typewriter
+	Carriage roll-up
-	Carriage roll-down
RS	Carriage rolled up and shifted left so next character will appear in upper left corner of scope.
EVAL	Displays decimal value of first component of Y.
MOD	Displays binary scale of Y, S[Y] .

TRW SPACE TECHNOLOGY LABORATORIES
THOMPSON RAMO WOOLDRIDGE INC.
ONE SPACE PARK · REDONDO BEACH, CALIFORNIA

EXECUTIVE OFFICES

13 January 1965

Dr. Douglas Engelbart
Stanford Research Institute
Stanford, California

Dear Dr. Engelbart:

We cordially invite you to join with us in opening demonstrations marking the inauguration of TRW Space Technology Laboratories' new four-station On-Line Computing Center to be held on Wednesday, 27 January 1965. The first demonstration will be at 9:30 in the morning, and afternoon demonstrations will start at 1:30. Luncheon will be served at noon. You are invited to witness one of the morning or afternoon demonstrations and join us for lunch. Dr. Simon Ramo, Vice Chairman of the Board of TRW, will speak at the luncheon.

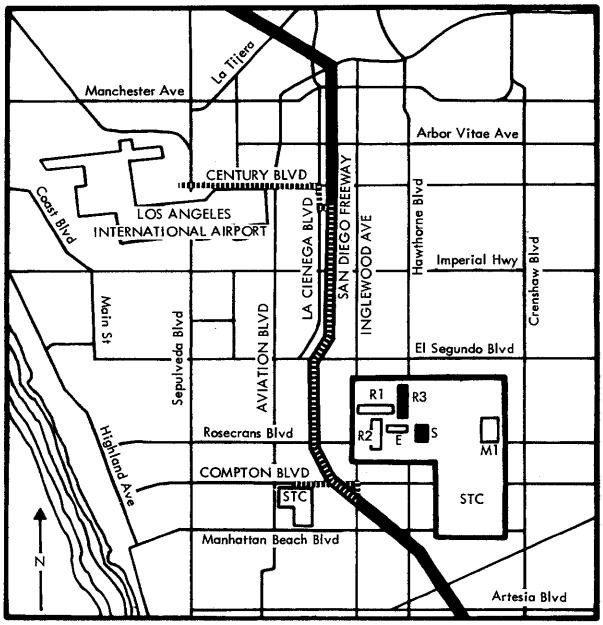
This facility represents a new approach to the problem of communication between men and computers, and we believe it will be of interest to you. A report briefly describing the system is enclosed along with additional information concerning this event. We sincerely hope that you will be able to attend.

Sincerely yours,

Ruben F. Mettler

Ruben F. Mettler
President

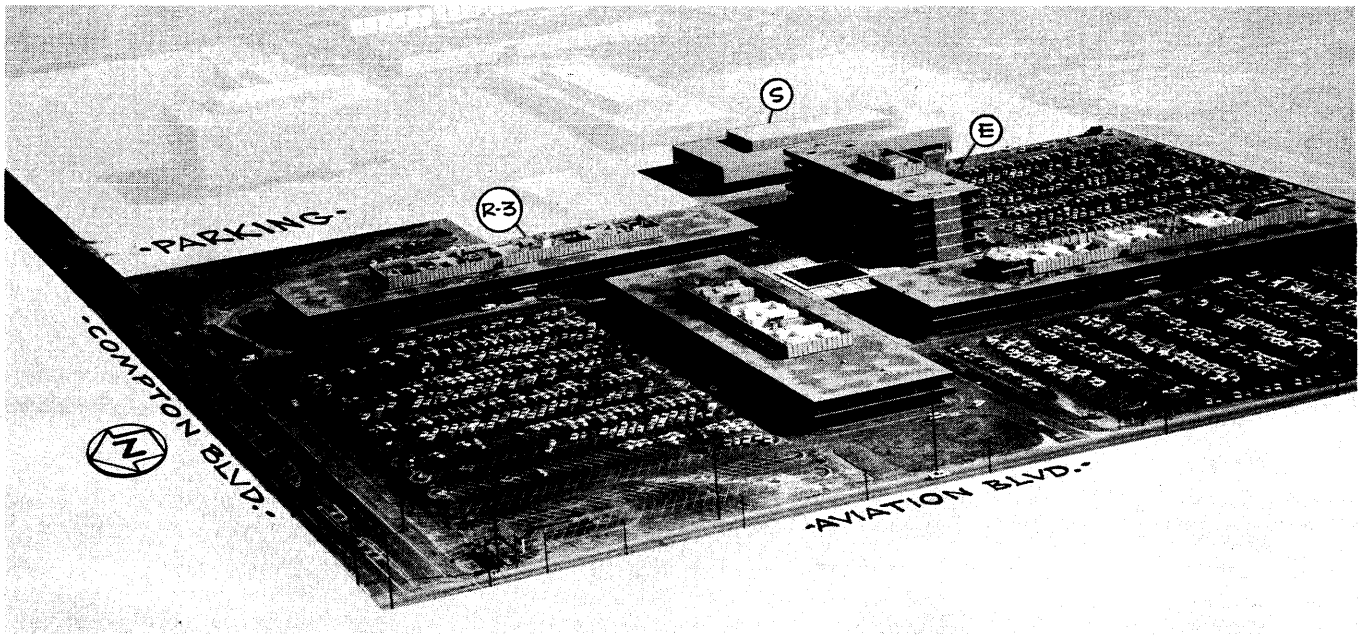
Enclosures



STL Space Technology Center is located at Aviation and Compton (Marine) Boulevards in Redondo Beach, California. It is approximately five miles southeast of the Los Angeles International Airport. The San Diego Freeway (Inglewood Avenue ramp) is just a few blocks east of the Space Park complex.

The facility, pictured here, is composed of six buildings. Building R3, where the demonstrations are to be held, and Building S, where luncheon will be served and the ceremonies will take place, are labeled for easy identification.

Visitors are requested to go to the lobby in Building R3 for passes to the activities. If you plan to attend the afternoon demonstration, please arrive by 11:30 so you may attend the luncheon. Parking will be provided in the outlined area.



TRW Space Technology Laboratories

ON-LINE COMPUTING CENTER

Demonstrations

This event will include four one-hour tour-demonstrations of STL's new four-station On-Line Computing Center. Demonstrations will be held at 9:30 and 10:30 in the morning and at 1:30 and 2:30 in the afternoon. In each of them you will see the system in operation and you may, if you wish, operate it yourself. Professor Glen Culler, who conceived and designed this type of on-line system, and others involved with its development, will be on hand to answer questions and to discuss in detail any points of particular interest to you.

A luncheon will be held at noon with a program in which Professor Culler will speak on the motivation and basic philosophy underlying this on-line system and its future extensions and developments. Dr. Simon Ramo, Vice Chairman of the Board of Thompson Ramo Wooldridge Inc., will discuss the impact of this new technique.

If you wish, you may invite one or two people from your organization to attend. To assist us in planning for this event, we have enclosed a reply card and would appreciate a response from you by January 25. Maps of the area and the STL complex, as well as an agenda for the day and a document containing a general description of the system, are enclosed.