

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Lincoln Laboratory

January 9, 1964

TO: Jack I. Raffel
FROM: I. E. Sutherland and W. R. Sutherland
SUBJECT: A New Display Proposal

BACKGROUND

During the development of Sketchpad, it became obvious that the spot-by-spot display then in use on the TX-2 was inadequate for advanced applications. It is now generally accepted that a better display can be built. Efforts in building better displays have emphasized more speed in line and curve drawing through Analog Techniques (Lincoln, Data Display, TRW, etc.) or through Incremental Digital Techniques (Digital Equipment Corp., and Project MAC). The developments so far have all used the conventional notion of data transmission to the display from consecutive registers in memory.

PROPOSAL

We propose that a new display control should be built which can take information from non-consecutive registers in memory. The display will also be able to store information back into memory. In short, the display we propose will have many of the properties of a small computer. We propose, of course, to use some form of high speed automatic line and/or curve generation.

Rah [The important new notion we are advancing in this paper is that a more flexible display can be built if the display control is given more capability.]
It is likely that the control system built to give the display the extra capability outlined here could also be used with other I-O units. The proposed system would be particularly useful for the XEROX unit.

HISTORY

Bert had the idea for this new type of control during the summer of 1963. The same or similar notions came to Ivan during a telephone conversation with Bob Savell during November 1963. At the present time Ivan has a PDP-4 and display system on order which incorporate some of the more elementary ideas set forth here. While specifying the system to be purchased from DEC, Ivan realized the tremendous potential of the more flexible display control system outlined in this memorandum. Subsequent joint discussion clarified our position.

DISPLAY SYSTEM AS A COMPUTER

Let us consider the similarities between a display system and an ordinary computer. In an ordinary computer, there are some central flip-flop registers which form the arithmetic element. In a display system, there are some flip-flop registers which contain the coordinates and other display information. In an ordinary computer, there is a register called the program counter which tells from which register in memory the next instruction is to be taken. In a display system, there is a register which tells where the next datum is to come from. In an ordinary computer, there are many different instructions which, when performed, cause different operations to be performed on the central arithmetic registers and the program counter. In an ordinary spot-by-spot display there is only one instruction; namely, "load the coordinate registers and flash."

We propose to build a display control with several instructions. Most of the instructions will be of the "immediate" type which require only one memory cycle, e.g., TX-2's "REX," PDP-1's "LAW". These new one-cycle instructions will load the various registers of the display with or without intensification, and will therefore serve the function of data "output." By storing the data in the same memory word as the instruction to transfer it, we get data transmission at one memory cycle per transmission.

The "display file" then will consist of a large number of immediate instructions which load the display's registers. The display will "perform"

the display file rather than merely receiving data from it. If jump instructions appear in the display file, the display will perform them also, so that the display file need not be performed in consecutive order. Notice that a jump instruction is really a load immediate instruction, but it is the "program counter" which is loaded.

We propose to include some kind of subroutine capability in the display. If the display can "perform" subroutines, then repetitive portions of the picture need only appear in the display file once. More important, the structure of the picture can be reflected in the subroutine structure of the display file where it can be detected during light pen interrupts. To take an example from Sketchpad: an instance stored as multi-level subroutines could easily be treated either as a single entity or as its component parts. Sketchpad is now able to treat an instance only as a single item, because to do otherwise with a straight-through display file is very difficult.

With subroutining in the display file, the display automatically gets character generation. Estimates based on the DEC-type 330 incremental display show that about 400 letters can be displayed, flicker free, using subroutines. The display file necessary to store subroutines for all the roman alphabet and numbers is estimated at about one thousand (1000) 18-bit words. Since subroutining will be multilayered, adding subroutines for simple words will require only a very few registers. The character set is entirely arbitrary, of course.

The display system will have some kind of conditional jump capability. Suppose, for example, that a "skip on memory bit" instruction (TX-2's SKM) is included. If the central computer arranges for particular bits in memory to be changed at regular times, then the display can condition its picture on those bits to produce moving figures. For example, if a particular subroutine produces a box during even seconds and nothing during odd seconds, it will form a "blinking box." Again, if a subroutine draws a succession of arrowheads whose position is time dependent in three or more phases, the line of arrows will appear to flow. We are convinced that by putting motion into computer pictures we can make their meaning much clearer to an observer.

SCOPE CONSIDERATIONS

The notions outlined above have tremendous value for any display system. As so far described, it makes no difference how the display implements line or circle drawing, or even whether it has line or circle capability at all. The important idea presented so far is that the display file is a series of instructions to the display which are performed. At the present time, each instruction happens to occupy two registers, the one containing the TSD instruction and the one addressed by the TSD. Our new concept of display file usage can be programmed now with existing hardware by doubling the size of the display file; each spot will have its own individual TSD. With a spot-by-spot scope however, the subroutining feature is almost worthless.

As everyone knows, there are better things than a spot-by-spot scope. At this point, let us distinguish between "line" and "vector" drawing scopes. A line scope is one which draws a line from a specified start point to a specified end point, i.e., the coordinates of the start and end points are given explicitly. A vector drawing scope is one which draws a straight vector from a start point in a specified direction for a specified distance. The start point of a vector may be left over as the end of a previous vector. The direction and length are commonly specified in a cartesian form (Δx and Δy).

Let us extend the distinction to circles. A line-type circle scope will be given a start point (on the circumference), the coordinates of the circle center, and some indication of the amount of circle to be drawn. A vector-type circle scope will be given the start point, initial slope, radius of curvature and amount of circle to be drawn. The start point again may be left over as the end point of a previous vector or circle. The extension to other curves is obvious. Larry's scope is a vector-type for vectors and a line-type for "circles".

How many things a display file subroutine can create depends on the type of display unit. For an ordinary spot-by-spot display, a subroutine can place dots in a fixed pattern at a fixed position on the picture. This is not very useful. For a display with line (as distinguished from vector) drawing capability, a subroutine can draw a line from any preset position to the given end point of the line, again not very useful. A display with vector (as distinguished from line) capability will be able to draw a symbol of fixed size and rotation at several arbitrary positions on the screen with a single subroutine. This is sufficiently useful to warrant implementation.

If a circle or other curve capability is built into the display, the specification of the circle or curve should be relative to an arbitrary starting point. That is, one should specify the radius of the circle and its initial slope, rather than specifying the specific coordinates for the center of the circle. It should be fairly easy to formulate curve generators which use such a relative specification. If we define as "relative" any display for which the only specific X Y coordinate specification is a starting point, then any relative display can generate and arbitrarily position symbols by subroutine.

If the display includes some kind of scale registers which change the size of the curves and vectors drawn, then a single subroutine can make different sizes of symbols. If the display contains rotation registers, then a single subroutine can produce symbols at different angles. We propose that size and rotation be omitted for now.

Notice that in a relative display system a subroutine may cause a total net displacement of the coordinate registers. This is very useful. For example, the subroutines which display the alphabet will include unintensified motion so that the coordinate registers are correctly set up for the next letter to be printed. Not only is the type set arbitrary, but you get the "IBM Executive" proportional spacing as well.

Particular subroutines may do different things during different iterations or at different times. For example, a particular subroutine might merely

cause a small displacement of the coordinate registers to the right every 7th time it is used. It will be very useful to be able to read back and store the final X Y coordinates after performing such a subroutine. For example, the small-displacement-every-7th-use subroutine, combined with an appropriate store instruction would make it possible to automatically move a line of text slowly across the screen. The kind of slow motion desired for the program "WINDOW" by L. M. Hantman, would thus be possible with almost no load on the central computer.

We propose, therefore, to include in the display control, a store instruction. This store instruction will insert the value of X Y into the appropriate portion of an instruction of the new load X Y (immediate) type. Similar store operations should be available for the other active registers of the display, e.g., intensity, etc.

It will be all but impossible to implement this store-type operation with any analog-scheme vector and curve drawing display. We insist, therefore, that the display itself must be of the incremental digital type, as, indeed, we have said before.

We have outlined a great deal of power for this proposed display. Fortunately, a great deal of the power needed is already built into the TX-2. Therefore, the specific equipment we have in mind will be fairly simple. In what follows, some attempt has been made to generalize the notion of "performing" a data output file to other IO units. Other input-output equipment (XEROX) may well profit from such a treatment.

We believe that the following points should now be clear:

- a. The notion of performing an output file is powerful. It can be applied to several types of scopes and even other I. O. units.
- b. The existing TX-2 hardware can be made to perform display files in a limited, wasteful fashion.
- c. Vector-type scopes are preferable to line-type scopes. Circle and curve drawing scopes should also be of the "vector" (relative) type.
- d. The power of performing a display file can never be fully realized without a read-back (store) capability. We must have digital incremental scopes for this reason alone, not to mention edge detection, pen tracking and stability.

- f. The TSD-data register pair is a cumbersome format for a single instruction. An increase of speed, saving of storage, and relatively simple implementation of a "data channel" are possible as outlined below.


SPECIFIC PROPOSAL

We propose that a single TX-2 op code be implemented as follows:

The operation is not addressable, nor indexable, nor deferrable. Configuration bits are used for variations and/or dismiss (maybe) The bits normally devoted to specifying an index register and the address (24 in all) are treated as data. The instruction has the "dismiss and wait" feature of the TSD.

When the instruction is performed, the 24 (1.1 to 3.6) bits are transmitted as data to the I.O unit whose sequence is active. The configuration bits are also sent to the I.O. unit to specify how the 24 bits of data are to be used. The instruction will dismiss unless held. When the operation started is complete (this may be at once, in the case of loading the intensity setting) the flag will again be raised and the next instruction will be performed.

We propose that the TSD in the display sequence should become an input or store rather than an output or load instruction. The TSD will place in the addressed register a "display X Y immediate no flash" with the current values of the coordinate registers correctly placed. A few miscellaneous functions such as "Flash current position," "Continue after light pen interrupt," etc., will be implemented in the IOS instruction.

All the remaining functions outlined for the display will, of course, using existing TX-2 instructions. Subroutining is possible, even arithmetic operations can be done at display time, if desired. The important new capability is that the display sequence performs the display file. 

DATA CHANNEL

Most of the instructions to be performed in the display file will be of the display immediate type. In order to gain a speed advantage, a "data

channel" could be implemented, meeting the following requirements. The data channel would be able to keep track of the display sequence program counter independently from X memory. The data channel would be able to perform only the display immediate instructions (which use the single newly-defined op code) by itself. If the data were coming from one memory (say U memory) and the other TX-2 sequences were concerned only with other memories, then the operation of the data channel would be concurrent (as contrasted with interleaved) with the rest of TX-2. Whenever the data channel came to an instruction it was unable to perform, it would raise the flag of the display sequence for help.

If the data channel is connected to a particular sequence, performing any load immediate instruction would dismiss the sequence and return I-O control to the data channel. If the data channel is not connected, the central machine would perform the instruction. Thus, the only effect of the data channel would be to free the central machine from such frequent I-O processing. An identical display file (program) will work either with or without the channel. For example, if one wishes to read tape while displaying, one will "borrow" the channel from the display for use with the tape. Meanwhile, display will continue, but with the central processor performing all the display instructions.

As mentioned above, Jump instructions are really load immediate instructions which load the program counter. It will probably be easy to make the data channel able to perform jumps. Slightly more difficult, but very useful, would be to make the data channel itself capable of performing subroutines, perhaps only a single level of subroutining, with help from TX-2 for more complicated structures. You can get more complicated, if you wish, but the additional gains become very small because the number of instructions of other types is probably very small.

For input rather than output units, the display (load) Immediate instruction should perhaps become a store immediate. Still only one memory cycle is required, but the programmer is given a responsibility to preload his table with "empty" store instructions. Since the store would affect only the right portion of the word, a single pre-set data file could be used for many transfers.

SCOPE NEEDED

As we have pointed out, a digital incremental display will be necessary to realize the full capability of the performed display file. We propose that an initial version of this display should be designed and implemented as soon as possible. The salient characteristics of the incremental display are outlined below.

The basic display should be built around counting decoders. Either a Lin-developed display could be used, or the DEC type 330 incrementing display (about \$18K) could be used. The counting decoders should be 10 bits each.

The display should have an "incremental" mode. The "display immediate incremental" instruction would cause the display to make six spots, each adjacent to the previous one. The direction of motion for each spot would be specified by four of the 24 bits transmitted by the display immediate instruction. Such direct incremental use of an incremental display has been explored by DEC and found to be useful.

The display should have a vector capability. The vector should be specified as ΔX and ΔY , with each Δ being 10 bits plus sign, for 11 bits in all. The vector capability could either be implemented with binary rate multipliers or with digital differential techniques. The binary rate multiplier method is cheaper, but will not generalize to circles.

The display might have circle capability. The details of implementing circle capability are not a fitting subject for inclusion here. If circle capability is not included, at least provision should be made for later expansion to circles.

We do not feel that rotation nor size change capability are necessary in the display. If a need for rotation or size change appears at a later time, it can be included then.

Full advantage of the digital properties of the display should be taken to ease the jobs of pen tracking and edge detection. Whenever the pen sees light, incrementing of the line or circle being drawn should stop so that the light pen sequence may note the coordinates which the spot has reached. A suitable IOS in the pen sequence should cause incremental motion to resume. Similarly, when incrementing brings the spot to the edge of the screen, a higher priority sequence should be called.

The display should be combined with a suitable manual intervention facility. The exact form of the knobs and so on to be provided is relatively immaterial. At a cost of only money and no design effort, Lincoln could purchase a carbon copy of the console DEC is building to Ivan's specifications.

LET US MOVE AHEAD WITH VIGOR

We believe that the display concepts outlined here, together with the hardware changes and additions necessary to implement them, will constitute a major step forward. Crudely speaking, we can gain an order of magnitude in display capability. Arbitrary character (or picture) sets can be easily implemented; flashing, winking, moving, flowing pictures can be used; and the display file (program) does not need to be radically changed to get a large change in the picture shown. The new system involves compiling a display program rather than listing a display table. We most strongly recommend that the essentials of this proposal be included in the TX-2 program now underway.

IES:WRS:smr

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

5 October 1964

TO: TX-2 Users
FROM: T. Johnson
SUBJ: Curve Display

Seq. 62, Curve Display, is now on-line. Enclosed is a draft of the scope's operations. At present, circles are not available. When this last problem is overcome, an addendum will be circulated to complete the description of the display's operation.

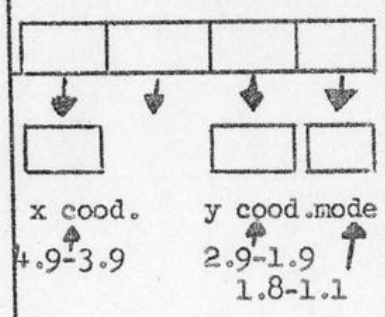

T. Johnson

TJ/jk

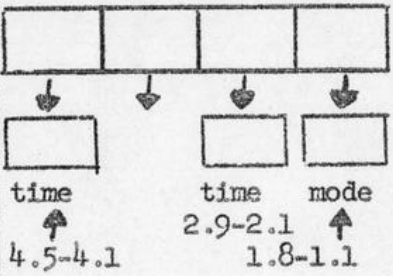
enclosure

The scope is a cartesian coordinate, high speed display with 10 bit precision in the vertical and horizontal axes. Points, lines, parabolas, hyperbolas and circles can be displayed repeatedly to maintain a viewable display.

OPERATIONS

<p>ICS₆₂³⁰⁰⁰⁰</p>	<p>Select Scope (connect)</p>	<p>This instruction gives a centered origin. (Other origins are not available.) Additional mode selections are accomplished with the TSD instruction.</p>
<p>$\alpha_{TSD} T_j$</p>	 <p>x coord. y coord. mode +.9-3.9 2.9-1.9 1.8-1.1</p>	<p>TSD copies from T_j to the scope buffer. The two 10 bit fields are interpreted as signed ones complement numbers. However, the scope is set so +0 = -0 for continuity. The mode bits determine whether the X, Y fields are to be used as coordinates, rates, radius, or arc length as discussed below.</p>
<p>CONTENTS OF T_j MODE FIELD ONLY (bits 1.8-1.1)</p>	<p style="text-align: center; border: 1px solid black; padding: 2px;">LOADING MODES</p>	
<p>23</p>	<p>display point</p>	<p>Used to display a point anywhere on the scope screen. (also can be used to load startpoint for any curve - see Mode 3)</p>

<p>21</p>	<p>display nearby point (high speed)</p>	<p>Used to display a point within 1/2" of the last position of the <u>beam</u> when faster plotting speeds are required (30 μs/pt compared with 70 μs/pt). Also can be used to load startpoint for any curve - see Mode 1.</p>
<p>3</p>	<p>Load Point</p>	<p>Used for loading any startpoint for any curve. The X, Y coordinates are stored in the scope for subsequent use in the <u>display curve</u> modes. The beam is moved to the specified start point, but not intensified.</p>
<p>1</p>	<p>Load nearby point</p>	<p>For loading curve startpoints within 2" of the last position of the beam when faster curve specification is desirable. Normally this mode does not save any time when more than one TX-2 sequence is running. The beam is moved to the startpoint, but not intensified.</p>
<p>2</p>	<p>Load rate (slope)</p>	<p>For loading the initial rates of curves (lines do not require this particular mode). The X field specifies the <u>complement</u> of $dx/dt _{initial}$ where t is time; the Y field specifies the complement of $dy/dt _{initial}$. Note $slope = \frac{dy/dt}{dx/dt} = \frac{Y \text{ rate}}{X \text{ rate}} = dy/dx$. Thus an infinite slope can be specified merely by setting $X_{rate} = 0$.</p>

		<p>The magnitude of X_{rate} and Y_{rate} determine, among other things, the initial brightness of the curve (see curve equations below). X_{rate} and Y_{rate} are stored in the slope for subsequent use in the <u>display curve</u> modes.</p>
<p>4</p>	<p>Load arc length (time)</p> 	<p>Determines the length of the curve starting at the initial position and rate. T (for time period) is stored in the scope for subsequent use in the <u>display curve</u> mode. Note the bit fields are different for <u>this mode only</u>. T (an integer) is split into two fields, the least significant bit occupies 2.1 and the most significant bit occupies 4.5. Thus, if T is less than 10 bits in length, bits 4.1-4.5 must be zero.</p>
<p>10</p>	<p>DISPLAY CURVE MODES Display line</p>	<p>The X field specifies X_{rate}, Y specifies Y_{rate}. Thus Mode 2 is by-passed for straight lines only. A line is displayed from the last position of the beam. The X & Y component lengths of the lines are $(X_{rate})(T)$ and $(Y_{rate})(T)$ respectively. At Maximum rate, $T = 1705_8$ for a component length equal to one scope <u>diameter</u>. The position of the beam at the end of the line is automatically stored in the scope for subsequent use as the start-point for another curve, if desired. The parametric equations describing the displayed</p>

		<p>line are:</p> $x = (X_{\text{rate}}) (t) + (x \text{ coord.})$ $y = (Y_{\text{rate}}) (t) + (y \text{ coord.})$ <p>for $0 \leq t \leq T$</p>
30	Blank line	Same as Mode 10, except the line does not show on the display.
11	Display parabola	<p>The X field specifies the <u>complement</u> of A, the rate of change of X_{rate}; the Y field specifies the complement of B, the rate of change of Y_{rate}. A parabola is displayed from the last position of the beam with an initial rate specified either from a Load rate mode (#2) or from the rate left in the scope at the end of another parabola or circle. (lines do not save rates). Care must be taken not to allow the rate to build up above its maximum by choosing the right parameters. For an orthogonal parabola displayed at maximum rate on one axis, and a rate of change of rate (A or B) of $1/4$ maximum on the other axis, $T = 1705g$ for a parabola extending over the diameter of the scope and curving to $1/2$ a diameter (with a slope of one). With A or B greater than $1/4$, the <u>changing rate</u> would grow to large. The position and rate at termination are automatically stored for subsequent use if desired. The parametric equations describing</p>

		<p>the parabola are:</p> $x = (A) (t^2) + (X_{rate})(t) + (X \text{ coord.})$ $y = (B) (t^2) + (Y_{rate})(t) + (y \text{ coord.})$ <p style="text-align: center;">for $0 \leq t \leq T$</p> <p>NOTE: if both A & B \neq 0, the parabola is rotated. If A = 0 & $Y_{rate} = 0$ with (X coord.) & (y coord) = 0</p> $y = (B) (t^2)$ $x = X_{rate} (t)$ $\therefore t = \frac{x}{X_{rate}}$ $y = \frac{(B)}{(X_{rate})^2} x^2$
31	Blank Parabola	Same as Mode 11, except the parabola does not show on the display.
12	Display sharp parabola	<p>Same as Mode 11, except the parametric equations are:</p> $x = (K)(A)(t^2) + (X_{rate})(t) + (X \text{ coord.})$ $y = (K)(B)(t^2) + (Y_{rate})(t) + (y \text{ coord.})$ <p style="text-align: center;">for $0 \leq t \leq T$</p> <p>where,</p> $K = \frac{1}{.01616_{10}} = \frac{1}{.010214_8}$ $\frac{1}{K} \approx 62$ <p>This allows sharply curved parabolas while A and B still remain fractions of one.</p>
32	Blank sharp parabolas	Same as Mode 12, except the curve is not displayed.

TO: APEX et al
FROM: Alex Vanderburgh
SUBJECT: Display package for Tim's Scope

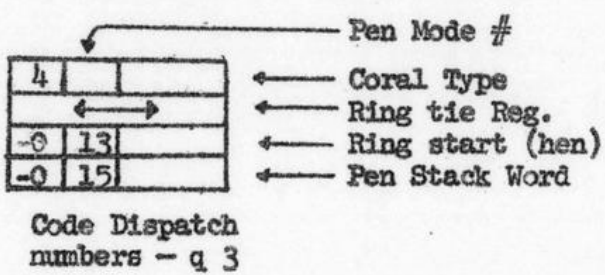
18 January 1965

1. A display package for curves, lines, points and characters is available for use on Scope 62 (Tim's).
2. At the moment it uses 3 CORAL block type numbers. The block format is diagrammed on the next page.
3. There are two character display code numbers:
 - a. 10 (3) for normal text. Right and Bottom edges are detected. Overflow from right is returned to center and down one. If bottom edge is detected, program jumps to "OFFB".
 - b. 11 (3) for fast text. - No edge detection.
4. Character codes 71, and 171 (WORD EXAM) give column spacing as specified by Q4. If current position is to the right, i.e., if beam must move to the left, it automatically moves down to avoid over printing.

Pseudo Character Code 203 specifies a new beam position to be taken from Quarters 4 and 2.

Character codes 14-17, Readin, Begin, No, Yes, 63 Black, 67 Red, 74-77, Lower, upper, Stop, Nullify (and 114-117 etc) are ignored.
5. A beam position resets script to normal. (So does carriage return.)
6. The program requires $376_{(8)}$ registers for instructions and 2300_8 for constants.
7. Pen Stack Controls and multiple scope switching are not yet included.

aly



INTERCONNECTING

