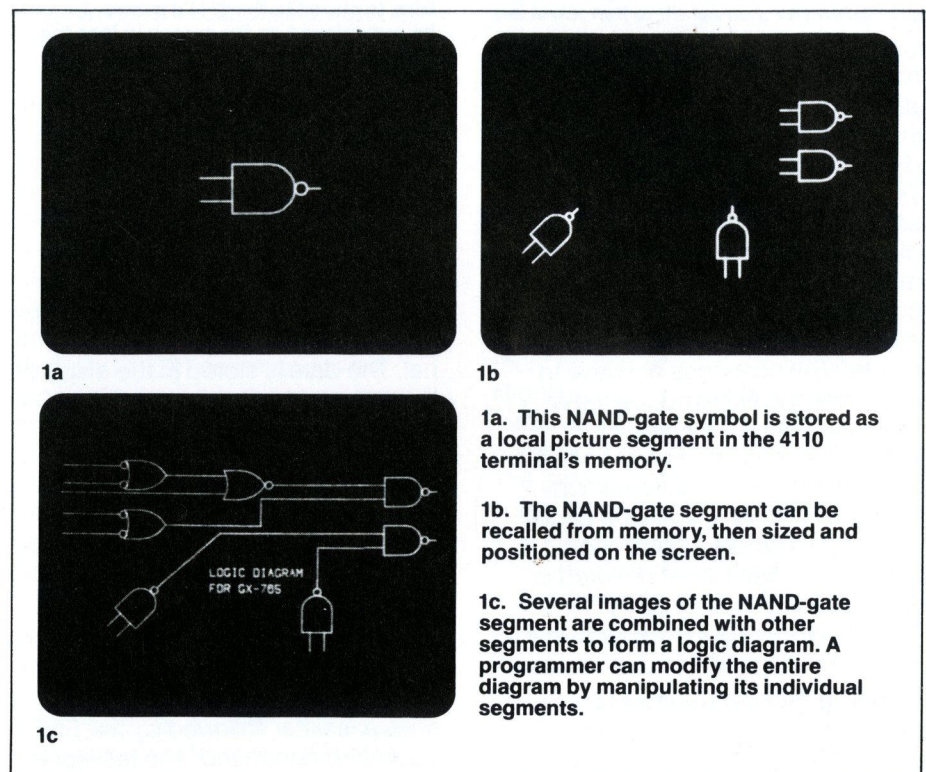


LOCAL PICTURE SEGMENTS

An architect sits before an outline of a building that is displayed on a computer terminal. After studying the doors, windows, and columns that fill one wall of the building, the architect decides to move a window across the screen to a more desirable location within the outline of the building. If defined as a local picture segment, the window can be repositioned quickly and independently of the rest of the picture.

A local segment is a specific part of a picture that is displayed on a terminal's screen. Though basically a sequence of moves and draws, the segment is treated as an entity by the terminal (Figure 1a). It can be stored in local memory, recalled, and then sized and positioned on the screen (Figure 1b). Because segment operations are performed locally, the user saves the time and money it takes to retransmit graphics data from the host system.

In SIGGRAPH-ACM's proposed core system of computer graphics standards, a picture segment is defined as an ordered collection of output primitives defining an image which is part of the picture on a view surface. As a chemist works with a compound by breaking it down into its component molecules, which are ordered collections of primitives called atoms, the graphics programmer divides a computerized picture into segments, ordered collections of primitives like lines and text strings. The programmer can then modify the whole picture by identifying, manipulating, and displaying individual segments (Figure 1c).



1a. This NAND-gate symbol is stored as a local picture segment in the 4110 terminal's memory.

1b. The NAND-gate segment can be recalled from memory, then sized and positioned on the screen.

1c. Several images of the NAND-gate segment are combined with other segments to form a logic diagram. A programmer can modify the entire diagram by manipulating its individual segments.

Many graphics terminals, whether the display technology is storage tube or raster-scan, are "blind" to the concept of picture segments. The segments are stored and manipulated on the host computer, then sent to the terminal for display as strings of graphics data. When a displayed segment is modified (for example, repositioned on the screen), the host must perform the modifications with its own CPU, then transmit the graphics data back to the terminal. When a picture displayed on the screen is erased and redrawn, the host must again transmit the graphics data to the terminal. The host, in these cases, is doing all the work, much of it repetitive, and the terminal user is paying for it in time-sharing costs and phone line delays.

A new family of graphics terminals, the Tektronix 4110 Series, provides a local graphics capability to alleviate the host's burden.

The 4110 Series terminals allow you to store, manipulate, and redisplay picture segments locally. The host does not have to continually transmit the same picture information every time a segment is modified or the screen is erased. The terminal can perform many of the modifications and redraw the picture from memory.

Tektronix[®]
COMMITTED TO EXCELLENCE

As an illustration of the benefits of local picture segments, consider an application in which an engineer designs an electronic circuit on the screen of a graphics terminal. Initially, the host sends the graphics data to display a menu of components (resistors, capacitors, diodes) on the terminal's screen. On the 4014, if the engineer erased the screen and then wished to redisplay the menu of components, the host would again have to send to the terminal every vector in every component.

On the new 4110 Series terminal, however, each component displayed on the screen can be stored in local random-access memory (RAM) as a numbered segment. Then, if the screen is erased, the segments can be repainted (redisplayed) on the screen from this memory. The host need send only a command to the 4110 that says, in effect, "Display the specified segments." The host does not have to retransmit the entire sequence of graphics data. The result is a marked increase in repaint speed and a decrease in host communications traffic. An integrated circuit mask, for example, containing 26,000 short vectors and stored in RAM as one or more segments, can be redrawn on the 4114 screen in less than half a second (Figure 2).

Creating Picture Segments

To display a picture on a terminal's screen, the host sends the terminal strings of graphics information that define the picture. On a 4110 terminal, the user creates a picture segment by "capturing" a string of graphics data and storing it in memory separately from other strings of graphics data. The process is similar to putting the data in parentheses, which, in this case, is the function of "Begin Segment" and "End Segment" commands.*

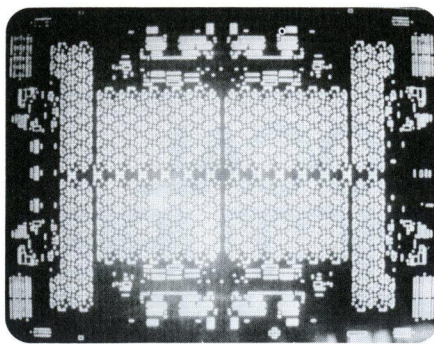


Figure 2. After this integrated circuit mask, which contains 26,000 short vectors, is stored in the 4114's memory, it can be redisplayed on the screen in less than half a second.

To create a segment, then, the host first sends a "Begin Segment" command to the 4110 terminal. The command includes a parameter that specifies the number of the segment (1 through 32767). After traversing its data base for the desired graphics information, the host sends a string of data to the terminal. The data is stored in the display list as the specified segment. An "End Segment" command terminates the definition of the segment.

In the electronic circuit design application, for example, the user can create a separate segment for each component displayed in the menu. First, the host sends the 4110 Series terminal the command "Begin Segment (1)". Next, the host sends the string of graphics data that defines a resistor, followed by an "End Segment" command. The resistor is stored in memory as segment 1. To continue, the host sends the command "Begin Segment (2)", followed by the string of data that defines a capacitor. After receiving an "End Segment" command, the terminal stores the capacitor in RAM as segment 2. In this way, the whole picture—the menu of electronic components—is divided into separate segments, each of which is stored in memory and can be specified independently of the other segments.

Manipulating Picture Segments

The 4110 Series terminals give the user the capability to modify the image of each picture segment (the segment as displayed on the screen) by changing such dynamic segment attributes as writing mode, visibility, highlighting, position, scale factors, rotation, and segment class. In addition, several other segment manipulation features—user-defined graphic input cursors, segments within segments, and disk storage—are included to minimize host processing and communications and optimize the 4110's contribution to faster, more efficient graphics.

Note that changing the dynamic attributes of a segment affect only the segment's image on the screen. The segment definition that is stored in memory cannot be edited after a segment is created. All graphic primitives within a segment (lines and text strings) are static, unchangeable parts of the segment. Any attributes of the graphic primitives within a segment (line style, graphics text font and size, etc.) are also static and unchangeable. For instance, if the resistor segment in the previous example is defined with a line style of dashed lines, then it will always be drawn with dashed lines.

*The command names in this feature note are descriptive of the command functions. Each command name represents a sequence of ASCII characters that is sent from the host to communicate with the terminal. See the **4110 Series Command Reference** for further information.

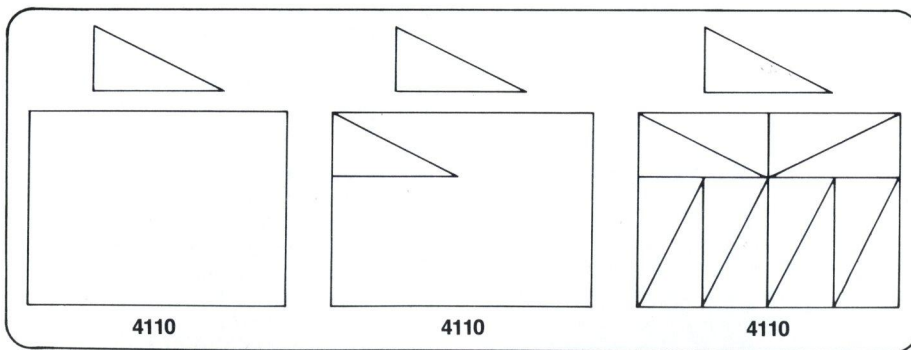


Figure 4. In this illustration of a blank nesting layout application, the manufacturer tries to cut as many of the triangular patterns as possible from the rectangular piece of material (left).

The operator at a 4110 terminal first moves the image of the pattern within the outline of the material (center).

By moving and rotating subsequent images of the pattern within the displayed outline, the operator can fit the maximum number of patterns in the piece of material (right).

By means of 2-dimensional image transformations, the user has the flexibility to create a picture segment and, after it is displayed, to modify its size, shape, and position to fit the application. In a blank nesting layout application, for example, a piece of material is displayed on the 4110 screen along with the outline of a pattern (Figure 4). The material might represent a size of fabric and the pattern an article of clothing. The object is to cut as many patterns as possible from the material with a minimum of waste. If the pattern is stored in memory as a picture segment, the terminal operator can position and rotate it within the displayed piece of material. By transforming subsequent images of the pattern as necessary, the operator can fit the maximum number of patterns in the piece of material. The resulting picture shows the manufacturers how to cut the fabric most efficiently.

User-defined graphic input cursor. For graphic input (GIN) operations, the user can replace a 4110 terminal's standard crosshair cursor with any locally retained picture segment, an arrow for example. The host program issues a "Set GIN Cursor" command to assign a segment as the graphic input cursor. The first parameter of the command is a special code that specifies the graphic input device (keyboard thumbwheels, graphic tablet, or plotter) and the graphic input func-

tion (locator, pick, or stroke). The second parameter specifies the number of the segment to be assigned as the GIN cursor.

This feature offers a convenient means to move a displayed segment across the screen. In the previously described circuit design application, the component picked from the menu by the engineer could be assigned as the graphic input cursor. The host program sends the command "Set GIN Cursor (0)(n)" which assigns the component (segment n) as the GIN cursor when the keyboard thumbwheels are enabled for the locator function (device code 0). Then, after the host enables the keyboard thumbwheels for the locator function, the component is displayed on the screen in refresh and can be moved to the desired position in the circuit by rotating the thumbwheels.

Segments within segments. The 4110 Series terminals allow the user to include a copy of one or more segments within another segment. Once the graphics data for a segment are sent from the host, the locally retained segment can be used like a cookie cutter to produce identical images within other segments. The host program issues an "Include Copy of Segment" command to include an image of the locally retained segment in a segment that is being defined. The first and only parameter of the command specifies the number of the retained segment to be copied.

For example, assume we're creating a segment that defines a map of several city blocks. The host first sends the 4110 terminal the com-

mand "Begin Segment (1)"; followed by the graphics data to draw a manhole cover, and terminated by an "End Segment" command. The manhole cover is now stored in memory as segment 1. Next, the host sends the command "Begin Segment (2)", followed by the graphics data to draw the outline of the map, which is displayed on the screen.

Now, if we need 50 manhole covers in the map, we can include 50 copies of segment 1 in segment 2 without sending any further graphics data from the host. For each manhole cover that we position in the map, the host sends the command "Include Copy of Segment (1)". The terminal then includes the image of the manhole cover (as it is displayed on the screen) as part of the definition of segment 2. After alternately positioning the 50 manhole covers in the map and sending the terminal 50 "Include Copy of Segment (1)" commands, the host terminates the definition of segment 2 with an "End Segment" command. Segment 2, as it is now stored in memory, contains not only the original outline of the map, but 50 manhole covers, all copies of segment 1, as well.

Classes of segments. With 4110 Series terminals, the user can modify related picture segments as a unit by grouping the segments into a segment class. A segment can be a member of one or more of a maximum of 64 classes. The host program issues a "Set Segment Class" command to add a segment to (or remove a segment from) a class. The first parameter of the command specifies the number of the segment. The second parameter is the removal array. It lists the segment classes from which the specified segment is to be removed. The third parameter is the addition array. It lists the segment classes to which the specified segment is to be added. The command "Set Segment Class (n)(3)(2,4)", for example, directs the terminal to remove segment n from class 3 and add segment n to classes 2 and 4. Segment n becomes a member of classes 2 and 4 and, if segment n was a member of class 3, it is no longer a member of class 3.

To illustrate the utility of segment classes, consider an application that displays a circuit diagram of a microprocessor-based oscilloscope calibration generator on a 4110 Series terminal. The circuit diagram consists of three basic modules: the microprocessing unit control circuitry, the analog and the timing circuitry. Each module contains a variety of components (ROM/RAM, amplifiers, phase-locked loops, etc.) that are defined in the application as individual picture segments.

By grouping the segments in the displayed diagram into three separate segment classes, one for each module, the user can, for example, collectively highlight all the segments in the timing module or collectively erase all the segments in the analog module. Data communications traffic is reduced because the host need send only a minimum of commands to the terminal specifying the segment class, not numerous commands specifying each and every segment.

Segments on disk. A 4110 Series terminal equipped with the optional flexible disk drives adds the capacity to save locally retained picture segments on a disk. The host, then, transmits the graphics data in a segment only once. Even if the segment is deleted from the 4110 memory (when, for instance, the terminal is turned off), the host can command the 4110 to load the segment into memory directly from the disk. Loading the segment from disk is faster and less costly than retransmitting the data from the host.

By storing and processing picture segments locally, the 4110 Series terminals cut host CPU costs and data communications traffic. Freed from the frustrations of time-sharing delays, the operator at the terminal experiences improved interactivity and can better use computer graphics to get more work done in shorter sessions.

Creating A Sample Segment

The two following examples show how to create a picture segment on a 4110 Series terminal. The code on the left contains the descriptive command names that are listed in the **4110 Series Command Reference**. The command names represent sequences of ASCII characters that are sent from the host to communicate with the terminal. The FORTRAN code on the right calls subroutines from IGL (the TEKTRONIX 4010C01 PLOT 10 Interactive Graphics Library). IGL, available as an option, can save users the time and expense needed to write their own graphics subroutine package.

Both examples of code create the same segment (pictured below). This segment, defined in the code as segment 1, is a square with diagonals, 200 units on a side. Its pivot point is at (100,100), the center of

the square, where the diagonals intersect. The pivot point serves to define the segment's position. Note that the pivot point is set before beginning the segment definition. This is necessary because once a segment has been created, its pivot point cannot be changed.

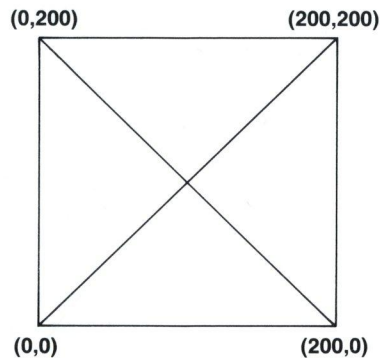
The <delete-segment> command (or DELSEG subroutine) deletes any segment 1 which might already exist. It is not possible to create a new segment with a number identical to that of an existing segment.

The <begin-segment> and <end-segment> commands (or OPNSEG and CLOSEG subroutines) begin and end the segment definition. Any commands which occur between these two commands are included in the definition of the segment.

```

<set—pivot—point: (100,100)>
<delete—segment: 1>
<begin—segment: 1>
  <enter—vector—mode>
    <xy: (0,0)>
    <xy: (200,0)>
    <xy: (200,200)>
    <xy: (0,200)>
    <xy: (0,0)>
    <xy: (200,200)>
  <enter—vector—mode>
    <xy: (0,200)>
    <xy: (200,0)>
<end—segment>
CALL GRSTRT (4114,1)
CALL WINDOW (0.,4095.,0.,3120.)
CALL W22NDC (100.,100.,PXPVT,PYPVT)
CALL SET2PV (PXPVT, PYPVT)
CALL DELSEG (1)
CALL OPNSEG (1)
CALL MOVE (0.,0.)
CALL DRAW (200.,0.)
CALL DRAW (200.,200.)
CALL DRAW (0.,200.)
CALL DRAW (0.,0.)
CALL DRAW (200.,200.)
CALL DRAW (200.,0.)
CALL MOVE (0.,200.)
CALL DRAW (200.,0.)
CALL CLOSEG
CALL GRSTOP
STOP
END

```



For further information,
contact:

**U.S.A., Asia, Australia,
Central & South America,
Japan**

Tektronix, Inc.
P.O. Box 4828
Portland, OR 97208
For additional literature, or the
address and phone number of the
Tektronix Sales Office
nearest you, contact:
Phone: 800/547-6711
Oregon only 800/452-6773
Telex: 910-467-8708
Cable: TEKTRONIX

**Europe, Africa,
Middle East**


Tektronix Europe B.V.
Postbox 827
1180 AV Amstelveen
The Netherlands
Telex: 18312

Canada

Tektronix Canada Inc.
P.O. Box 6500
Barrie, Ontario L4M 4V3
Phone: 705/737-2700

**Tektronix sales and service
offices around the world:**

Argentina, Australia, Austria,
Belgium, Bolivia, Brazil,
Canada, Chile, Colombia,
Costa Rica, Denmark, East
Africa, Ecuador, Egypt, El
Salvador, Federal Republic of
Germany, Finland, France,
Greece, Hong Kong, Iceland,
India, Indonesia, Iraq, Israel,
Italy, Ivory Coast, Japan,
Jordan, Korea, Kuwait,
Lebanon, Malaysia, Mexico,
Morocco, The Netherlands,
New Zealand, Norway,
Pakistan, Panama, Peru,
Philippines, Portugal,
Republic of South Africa,
Saudi Arabia, Singapore,
Spain, Sri Lanka, Sudan,
Surinam, Sweden,
Switzerland, Syria, Taiwan,
Thailand, Turkey, Tunisia,
United Kingdom, Uruguay,
Venezuela, Zambia.

Copyright © 1981, Tektronix, Inc. All rights reserved. Printed in U.S.A. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specification and price change privileges reserved. TEKTRONIX, TEK, SCOPE-MOBILE, and  are registered trademarks of Tektronix, Inc.; TELEQUIPMENT is a registered trademark of Tektronix U.K. Limited.

Tektronix®
COMMITTED TO EXCELLENCE