To the Graduate Council:

I am submitting herewith a thesis written by Robert G. Majure entitled "A Color Graphics System for a Power Plant Monitoring System." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Donald W. Bouldin

Donald W. Bouldin, Major Professor

We have read this thesis
and recommend its acceptance:

Robert E. Bachnkeimer

John Lyle Bridall

Accepted for the Council:

The Graduate School

A COLOR GRAPHICS SYSTEM FOR A POWER PLANT MONITORING SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Robert G. Majure

June 1984

ABSTRACT

Computer-based plant monitoring systems for the electric power industry are increasingly required to have sophisticated user interfaces. Computer color graphics is being used in most cases to meet the requirement.

The need for the color graphics system to be discussed arose from business opportunities at Technology for Energy Corporation. The author was assigned to design and implement the color graphics system after expressing an interest in doing it to satisfy the requirements of a thesis for The University of Tennessee, Knoxville. The author relied heavily on Technology for Energy Corporation's previous experience in color graphics in nuclear plants to design the new system. The system is now operational in the designated power plant and will be used in future monitoring systems.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
	System Overview	4
	Thesis Overview	4
II.	PURPOSE OF COLOR GRAPHICS IN REAL-TIME MONITORING SYSTEMS	8
III.	CHOICE OF A COLOR GRAPHICS SYSTEM	10
	Display Requirements	10
	Compatibility with Existing Systems	12
	Tesselator 8000 Color Graphics System	13
IV.	DISPLAY CREATION AND EDITING	20
	User-Interface	24
	Utility Graphics Software	25
	Picture Editor	38
	Picture Compiler	42
	Display Specification	43
	Picture Specification	43
	Picture Data Structures	46
	Static Definition Specification	47
	Static Definition Data Structures	47
	Dynamic Definition Specification	47
	Dynamic Definitions Data Structures	61
V.	REAL-TIME UPDATING OF DISPLAYS	67
VI.	SYSTEM LIMITATIONS	69
VII.	CONCLUSIONS	70

CHAPTER PAGE

LIST OF REFERENCES 71

APPENDIX 73

VITA 75

LIST OF FIGURES

FIGURE		PAGE
1.	Plant Monitoring System Layout	2
2.	Sample Data Flow Diagram	3
3.	TEC Power Monitor Data Flow Diagram	5
4.	TECGRAPH Data Flow Diagram	6
5.	Pixel Representation of the Character "A"	14
6.	Host Computer Tesselator Software Hierarchy	17
7.	Picture Hierarchy	23
8.	Vertical Bar Charts	35
9.	Horizontal Bar Charts	36
10.	Picture Editor Structure	40
11.	Typical Dynamic Field Specification	44
12.	Picture Definition Specification	45
13.	Dynamic Field Specification	48
14.	Dynamic Text Field Specification	50
15.	Typical Dynamic Text Field Specification	51
16.	Dynamic Bar Chart Specification	53
17.	Typical Dynamic Bar Chart Specification	54
18.	Dynamic Numeric Field Specification	55
19.	Typical Dynamic Numeric Field Specification	56
20.	Dynamic Symbol Field Specification	57
21.	Typical Dynamic Symbol Field Specification	58
22.	Dynamic Time Field Specification	59

FIGURE PAGE

23. Typical Dynamic Time Field Specification 60

24. Dynamic Date Field Specification 62

25. Typical Dynamic Date Field Specification 63

26. Example of a Color Graphics Display. 74

CHAPTER I

INTRODUCTION

Computer-based color graphics systems have numerous applications. The system described in this thesis was specifically designed for monitoring activities in a coal-fired electric power plant. Its purpose is to aid both operators and engineers in making decisions that concern plant efficiency and safety. The color graphics system formats the plant data, acquired by the monitoring system, in a manner that is more easily comprehended by the operators and engineers. Figure 26, in the Appendix, illustrates a color graphics display used in an electric power plant.

This particular plant monitoring system was designed and built for Carolina Power and Light's Skyland Unit One facility by Technology for Energy Corporation (TEC) in Knoxville, Tennessee. Figure 1 shows the layout of the monitoring system in the plant. The term "terminal" is used to denote a graphics interface which consists of a CRT screen and keyboard for information entry.

In discussing software, Yourdon stuctured design methods (Page-Jones, 1980) will be used when possible. The primary method to be used is the data flow diagram. A sample data flow diagram is shown in Figure 2. The circles represent processes in the system. No implication is made concerning the process implementation. It may be one program on a computer or a collection of programs on different computers. The arcs between the processes represent information flow. Each arc will have a descriptive name for the information represented.

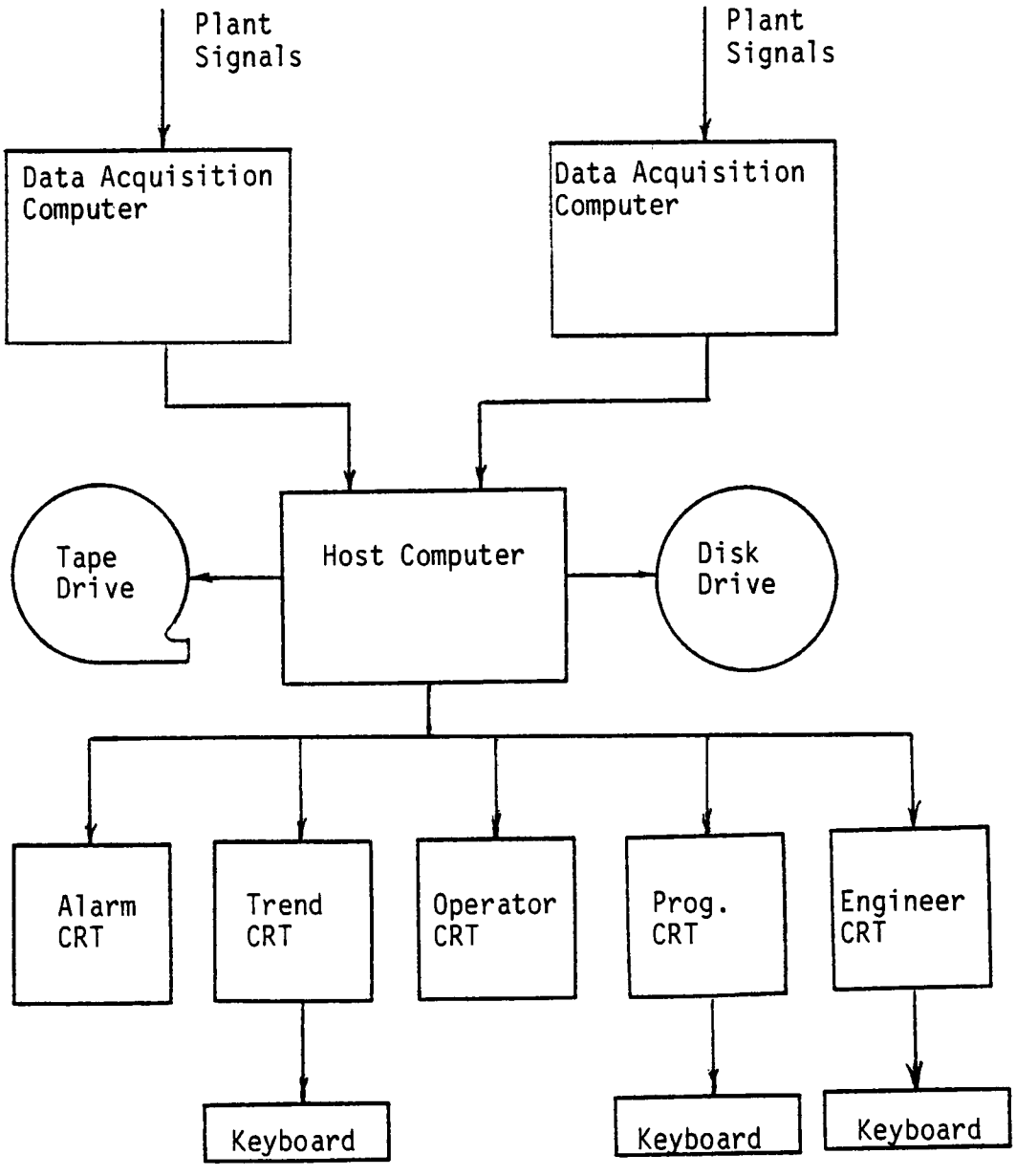


Figure 1. Plant monitoring system layout.

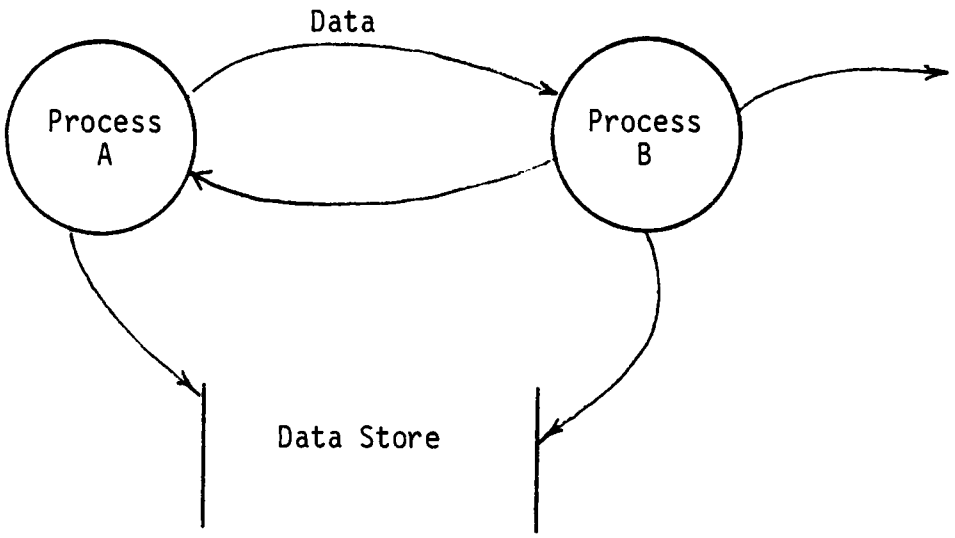


Figure 2. Sample data flow diagram.

Two parallel vertical lines represent a data store. A data store may be implemented as a disk file or as a global common on a computer.

System Overview

The plant monitoring system that is used consists of two main software packages that run on MODCOMP computers. The first package is called the TEC Power Monitor. This package is designed to provide three main functions in a monitoring system. A data flow diagram of the TEC Power Monitor is shown in Figure 3. The Plant Environment Data System's (PEDS) function is to collect data from the various sensors in the plant and place the data in the appropiate data stores. Each location in the data store is called a process point. Information Management Services' function is to format the collected data for presentation in the form of reports and messages. The Historian archives the data for future recall. The second software package is called TECGRAPH. It accesses the plant data stores and formats them in the form of color computer graphics. Figure 4 shows the data flow diagram for TECGRAPH.

The author specifically designed and implemented the Picture Editor and the Graphics Terminal Interface shown in Figure 4. The other elements of Figure 4 were adapted from a previous graphics system by the author and another project member.

Thesis Overview

The thesis is divided into seven chapters. Chapter II describes the purpose of using color graphics in a real-time monitoring system. Chapter III discusses the critera involved in choosing a color graphics system for this application. It also describes the ASEA

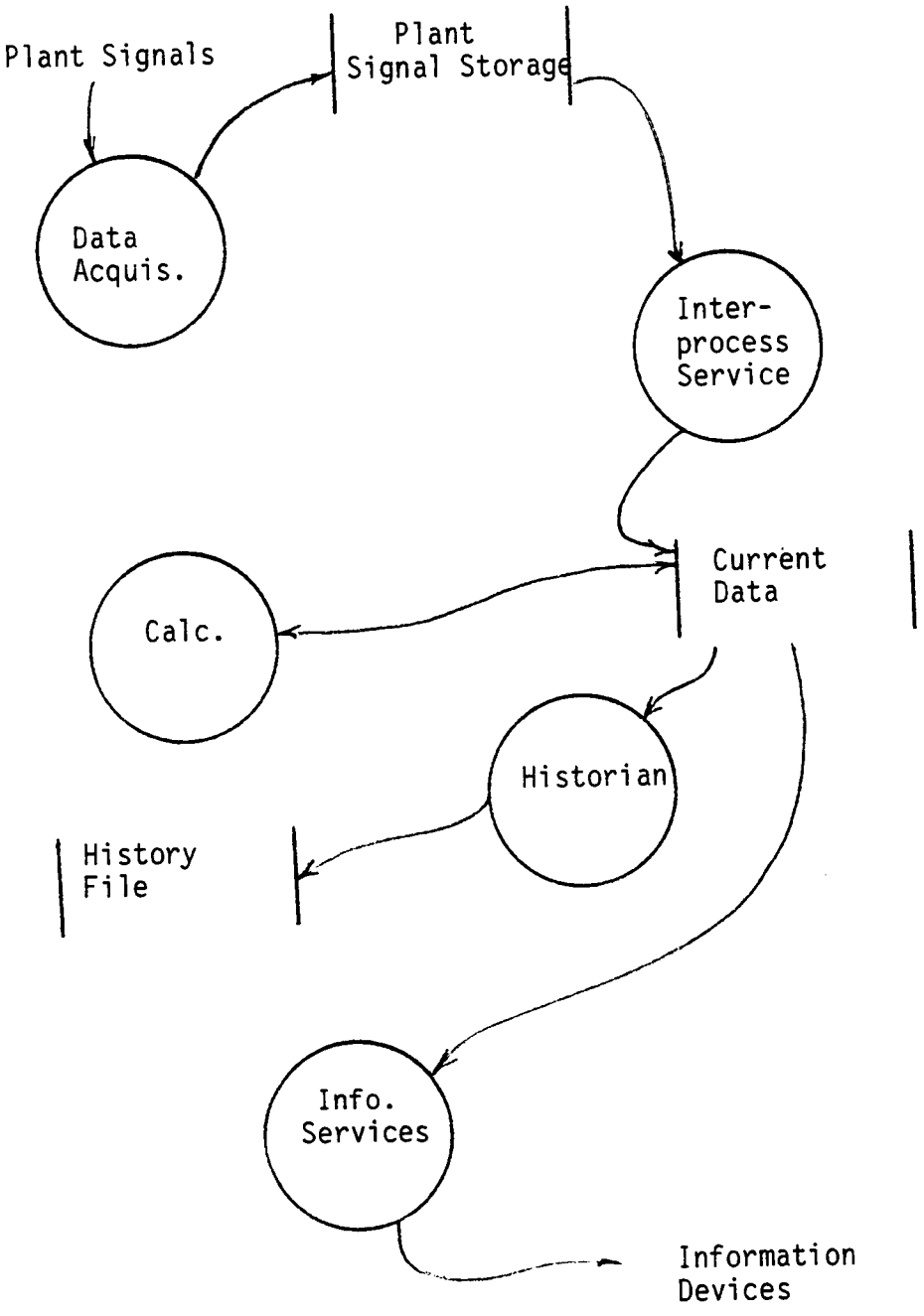


Figure 3. TEC Power Monitor data flow diagram.

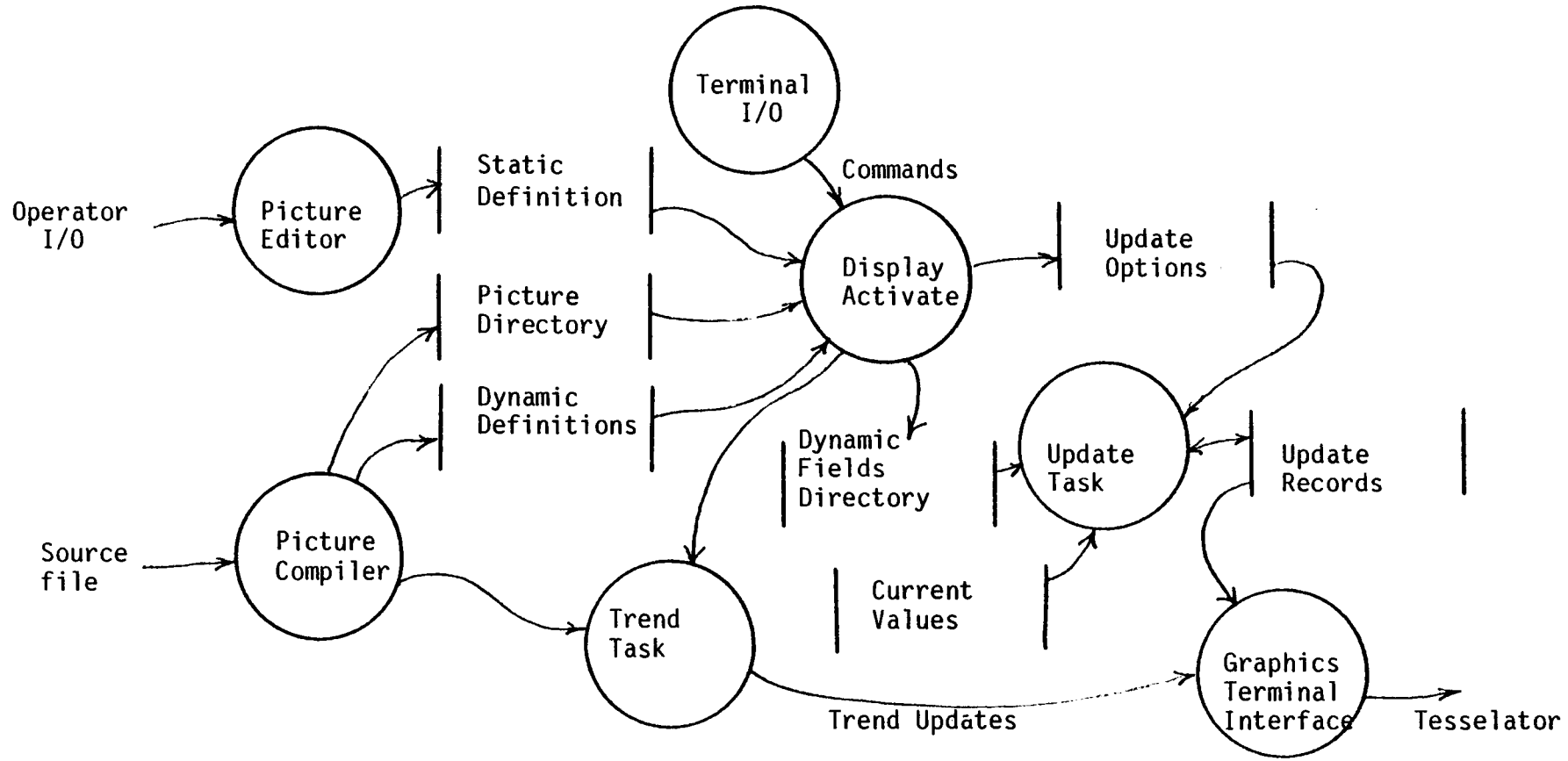


Figure 4. TECGRAPH data flow diagram.

graphics system for this application. It also describes the ASEA Tesselator 8000 Color Graphics System, which was the choice for this monitoring system. Chapter IV describes the display creation and editing process for the system. Chapter V describes the real-time updating of the dynamic portions of the displays. Chapter VI examines the limitations of this particular color graphics system and what could be done to improve it. Finally, Chapter VII presents the conclusions.

CHAPTER II

PURPOSE OF COLOR GRAPHICS IN REAL-TIME MONITORING SYSTEMS

There are numerous reasons for using color computer graphics in a real-time plant monitoring system. The ultimate reason is to improve plant efficiency and safety.

There are several factors concerning color graphics that lead to improved efficiency and safety. Information comprehension is a key factor. The plant operators and engineers must have accurate information with regard to plant parameters. An example would be the turbine parameters during plant startup. These parameters have limits, and if these limits are exceeded during startup, based on inadequate information from the plant sensors, serious damage could occur to the equipment. It is vitally important the operators have full knowledge of plant conditions. A color graphics display can help in this respect. A color graphics display aids information comprehension by formatting plant data in a manner that is easily understood by the plant operators and engineers. A display with meaningful symbols such as pumps, valves, and pipes, with corresponding plant data superimposed near the appropiate symbols is more readily comprehended than a table of alphanumeric information on a piece of paper.

Color is an important factor in information comprehension. The use of multiple colors allows the users easily to distinguish between various parts of a display. A typical color graphics system should have at least eight colors available for use. Resolution is the amount of detail available on the terminal screen. A suggested minimum is 512 by 256 pixels. A pixel is the smallest element of a

display. It usually looks like a dot to the observer. The more pixels in a display, the greater detail available for displaying graphical information.

Another advantage of color graphics in a monitoring system is the ability to provide a user-friendly interface. The term user-friendly means that the one who uses the system does not have to have extensive technical knowledge of that particular system. The user should be able to learn easily how to operate the system and be able to utilize the system with a minimum of frustration or error. A user-friendly interface is accomplished through the use of menus. A menu is usually constructed by partitioning the terminal screen into segments with descriptions in each segment. The user then can select a particular segment by positioning the graphics terminal cursor to that segment or by entering the description on a command line at the terminal.

Color graphics is a desirable feature in a monitoring system from a marketing standpoint. Almost no monitoring systems are being sold without this feature, according to both TEC's Sales Department and current advertising literature. Potential customers are impressed by the monitoring system having this "state-of-the-art" feature. Color graphics draws more attention to the monitoring system than would otherwise be possible.

CHAPTER III

CHOICE OF A COLOR GRAPHICS SYSTEM

This chapter explores the various factors that determine the choice of a particular system for a particular application. There are many graphics systems available in the market. Virtually none of them are suitable for every application. Therefore, one has to develop criteria for assessing how suitable a graphics system would be in the desired application.

Display Requirements

The first question to ask in an application such as a power plant monitoring system is: What are the display requirements? The requirements are:

1. Multiple colors.
2. Medium resolution.
3. No display flicker.
4. Dynamic fields capability.

For this application, at least eight different colors would be desirable. This is necessary since most power plants reserve the colors red, green, and yellow for dynamic fields, red meaning "alarm," green meaning "normal," and yellow meaning "alert." Another five colors are needed to construct the static portions of the displays. Medium resolution will suffice. In an application such as this, there is no need for an extremely high degree of display resolution as there is in a computer-aided design application. A resolution of 512 x 256 was decided as the minimum, since at lesser resolutions the

typical displays for this application become difficult to construct. An important requirement is that the display should not flicker Flickering occurs when the refresh rate of the graphics device is not fast enough. Refresh refers to the graphics device redrawing the screen. An accepted minimum refresh rate for which there is no flicker is 63 Hz. Finally, for a monitoring system, the graphics system should allow for dynamic fields capability. The dynamic fields in a graphics system are those segments of the display screen which change in real-time. Examples are bar charts, trends, numeric fields, text fields, and dynamic symbols. A bar chart, for instance, will change in height as the plant process point to which the bar is "tied" changes in value. Dynamic fields are required in any power plant application so it only makes sense to acquire this capability as part of the graphics system.

Performance criterion is primarily defined as the speed at which the graphics system will function under the required load. For this application, it was determined, primarily from nuclear plant requirements, that a new display should be drawn on the screen in less than two seconds. A delay that is longer than two seconds will frustrate the user and decrease desire to use the system. The dynamic fields should reflect plant data acquired by the computer within two seconds. This was determined to be sufficient since a two-second delay in operator response is not usually harmful.

The system must also perform properly in the application's physical environment. Since this application did not require the graphics hardware to be installed in a harsh environment, no special consideration was given to its ability to withstand such an environment.

Compatibility with Existing Systems

Compatibility with host computers is significant since the degree of compatibility determines the amount of labor required to interface the graphics system with the desired host computer. There are two brands of computers that are used in plant monitoring systems at Technology for Energy Corporation: MODCOMP and DEC. It was imperative that an existing interface between the graphics system and these computers be available. The hardware interface is usually one of the standard interfaces such as RS232. The software interface is the critical one. One must have software to handle the communication between the graphics system and the host computer operating system. This software is sometimes referred to as a handler. There must also be an interface between the application programs and the handler. A FORTRAN interface is highly desirable since FORTRAN is the primary language used in the monitoring system. A FORTRAN interface also enhances portability of the application software.

Compatibility with existing graphics software determines the cost to be incurred by adopting a new graphics system. The previous graphics system at TEC for plant monitoring systems was based on RAMTEK color graphics machines. The RAMTEK systems are dot-addressable. Dot-addressable means that each pixel on the screen must be controlled by the graphics software. This is both an advantage and a disadvantage. There is power and flexibility in being able to manipulate all the pixels on the screen, but there is also overhead in terms of a large amount of data transmission to the graphics system. It also complicates the graphics software design to have to control each pixel at all

times. A decision was made to use a symbol-oriented graphics system. In symbol-oriented systems, the host graphics software sends only symbol codes to the graphics device. These symbol codes represent many pixels of information. This decreases transmission overhead and software complexity considerably. As a comparison, consider the character "A". For a dot addressable graphics device to transmit an "A" as shown in Figure 5 would require 96 bits of information. In a symbol-oriented device, only eight bits of information need be transmitted. This change did necessitate some fundamental changes in the host computer graphics software. These changes will be discussed at a later point.

In assessing color graphics systems, cost is a prime consideration. For this application it was determined, based on market conditions, that a CRT, keyboard, and graphics generator should cost less than $20,000. This does not include the cost of software. The graphics generator should ideally be able to drive more than one CRT, thereby decreasing the cost of multiple CRTs.

Tesselator 8000 Color Graphics System

The graphics system that was chosen for this application is the ASEA Tesselator 8000 Color Graphics System. This system was designed specifically for process control applications. A power plant monitoring system can be thought of as a subset of a process control system. Therefore, the Tesselator appeared well suited for this application.

The Tesselator consists of several parts. The graphics generator consists of a central processor, graphics processor, communications interface, display memory, and display refresh circuitry. There are two Motorola MC68000 microprocessors in the generator, one for the

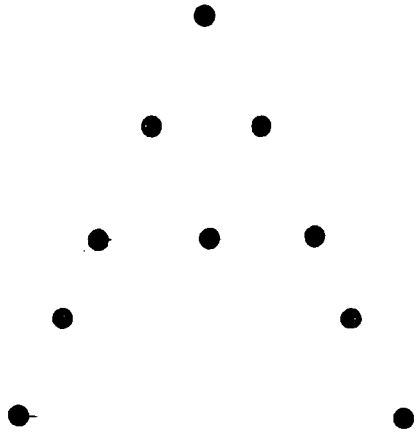


Figure 5. Pixel representation of the character "A".

central processor, and one for the graphics processor. There are two types of communications interfaces available. There is a high-speed link based on a MC68000 and a low-speed link that uses the standard RS232 link. For this application the low-speed interface was chosen due to lower cost. Each generator is capable of supporting up to three CRTs. For each CRT there is a display memory board and a display refresh board. Whatever is displayed on the CRT screen is kept in display memory and redrawn 63 times a second. The refresh rate of 63 Hz satisfied the requirement of no flickering of displays. The resolution of the displays is 712 x 336 pixels. This meets the criteria defined for resolution.

There are several options concerning what type of keyboard, for user entry, can be configured with the system. A general-purpose graphics keyboard that allows easy manipulation of the graphical symbols in the Tesselator was chosen for two of the three keyboards in the monitoring system. The other keyboard is a function keyboard for the plant operator. This keyboard allows the operator to select various functions in the monitoring system with a minimum of keystrokes. There is no capability for manipulating the Tesselator symbols with this keyboard. This is desirable since the plant operator should only be concerned with the plant operation.

A hard copy printer was also chosen for this application since it is desirable to have a permanent record of plant information in graphical form.

Host computer software for communicating with the Tesselator was provided. A FORTRAN subroutine, TESSIO, was provided to interface

with the application software. The software structure is in the form of layers as shown in Figure 6. The application program specifies the terminal number and then transmits or reads eight-bit control codes and symbol codes to or from the Tesselator. A symbol code represents a symbol such as a valve or an alphanumeric character. A control code represents functions such as "clear screen" or "set background color." Control codes are always preceded by the hexadecimal constant 1B. For example, to clear the terminal screen and write the word "TEST" at the current cursor location, the following hexadecimal constants are transmitted to the Tesselator with the desired terminal number specified:

1B 48 54 45 53 54

The user-interface is one of the most powerful features of the Tesselator. In addition to the keyboard, a trackball is available to manipulate the CRT cursor. One trackball was chosen for this application. By using the keyboard and the trackball, the user can define static displays at the terminal in a short period of time. The term "static" refers to that portion of a display that does not change with time.

The Tesselator has a feature called operator fields. The operator field is the main method of allowing the user to enter alphanumeric data at the terminal. An operator field is a single line of varying width on the CRT screen. The cursor can be positioned to the field by pressing the "next field" key on the keyboard. There is no limit, according to ASEA, on the number of fields on a CRT screen. By pressing the "next field" key, the cursor moves to the next nearest

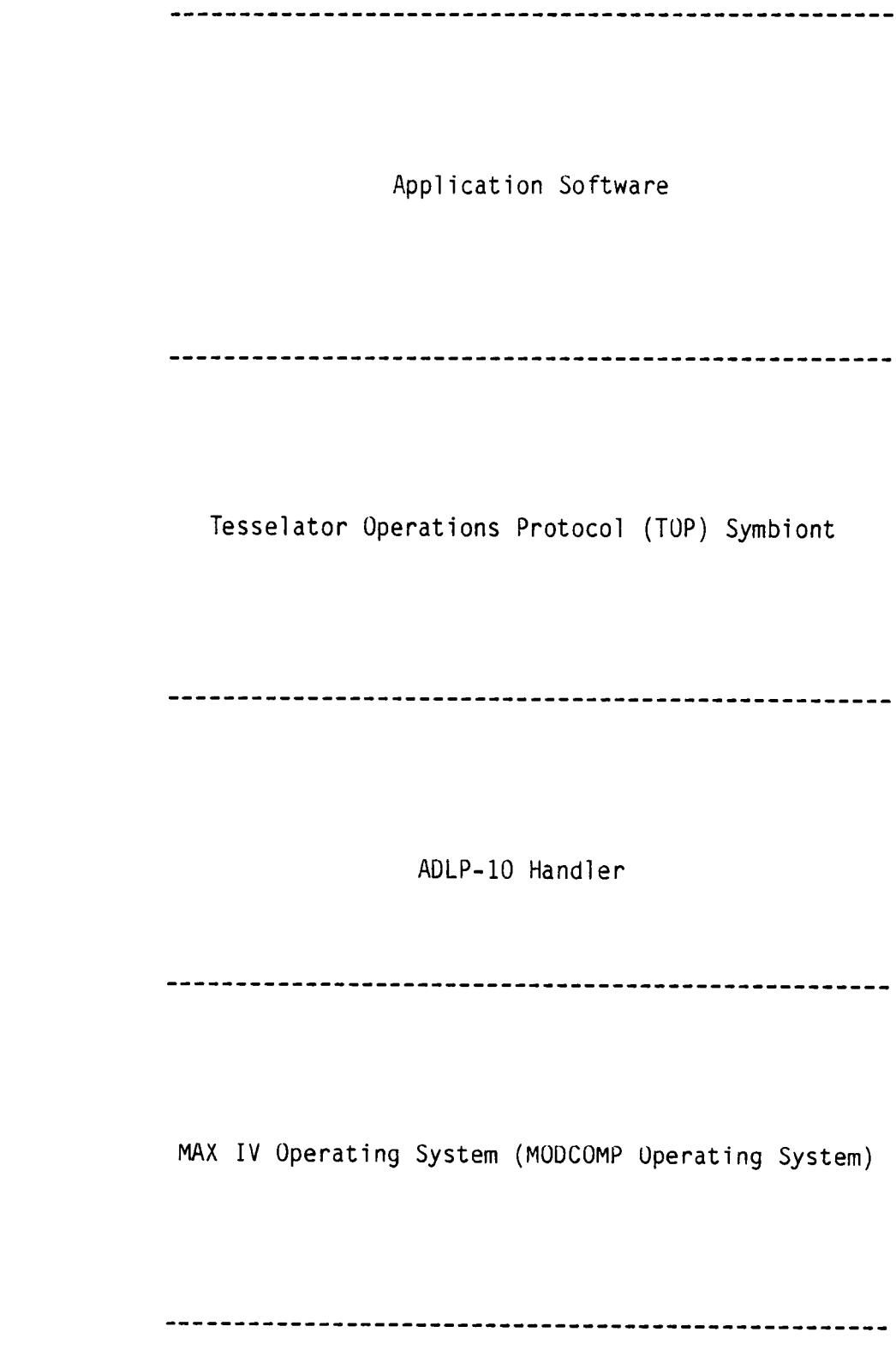


Figure 6. Host computer Tesselator software hierarchy.

operator field. In this manner, the user can move through the fields and enter or change the desired information. This is similar to user-interface techniques on many programmable instruments.

The Tesselator has a program that allows the user to create new symbols. This program is called the Symbol Editor. This is a very useful feature for defining symbols that are unique to a particular application. Once the symbol is defined it can be addressed in the same manner as the other symbols.

The Tesselator is also capable of real-time display. There are four features that provide this capability.

1. The computer field.
2. Bar-charts.
3. Trend plots.
4. Local display storage.

A computer field is similar to an operator field except that the computer field was designed for the host computer to manipulate. This allows the host computer to display alphanumeric information and update it in real-time. The computer field, like the operator field, automatically erases itself when new data is sent to it; it also justifies the data.

Horizontal and vertical bar charts are provided on the Tesselator. The host computer software needs only to send the location, size, type, and value of the bar chart. This decreases the overhead considerably from not having to send all the graphical information necessary for displaying a bar chart. Trend plots are plots of a data point versus time. This type of information allows the user to visually recognize trends in the process being monitored.

Trend plots are defined once for a display; then the host computer software sends only the new data for each plot and the Tesselator updates the plots.

Local display storage for up to 100 displays allows the host computer to store displays locally at the Tesselator for quick recall on the CRT screen without the communications overhead.

CHAPTER IV

DISPLAY CREATION AND EDITING

This chapter describes display creation and editing for the Tesselator-based TECGRAPH color graphics system. First, requirements for the system will be described. Then, the four fundamental entities of displays will be defined. The user-interface of the system will be described. Finally, the display specification will be described.

Display creation and editing is the process of the user defining what color graphics displays will be used in the monitoring system. The requirements for this aspect of the color graphics system must include the following criteria:

1. The user should not need detailed technical knowledge of the system.

2. The user should be able to create and edit displays in a short amount of time.

3. The user should be able to create and edit displays while the monitoring system continues to function normally.

The above requirements are necessary to achieve the goal of more efficient plant operation. If the user is required to learn the technical details of the system, he will not be inclined to use the system, or even purchase it. Technical details are defined as the details of the hardware or software required to run the monitoring system. This requirement can also be defined as requiring the system to be user-friendly. Creating and editing displays in a short amount of time is necessary so that the power plant's labor costs can be kept

to a minimum. If the system is expensive to use, it will be hard to market. Another factor in requiring quick creation and editing that the longer it takes to create or edit a display, the more likely that mistakes will be made by the user. This can have serious consequences. Therefore, every effort must be made to ensure that the user achieves exactly what he intended. The last requirement mentioned is that the monitoring system must continue to function properly when display editing is occurring. This is obvious since the monitoring system would be of little use if every time display editing occurred, the plant operators and engineers were deprived of plant data. The above requirements were set forth as criteria even before the Tesselator was chosen for the graphics system. The requirements stem primarily from customer requirements and competition.

The decision as to how the user would construct displays that would dynamically update in real-time is more complicated than just drawing the static lines, text, and symbols used in a static display. The author decided to use the following fundamental entities to construct displays for the graphics system:

1. Picture definitions.
2. Static definitions.
3. Dynamic definitions.
4. Symbol definitions.

This structure was chosen because it provided modularity in the software, compatibility with the previous TECGRAPH system, and suited the characteristics of the Tesselator.

The term "picture" will be used to describe a complete graphics display in a usable form for the plant personnel. It will have a name which can be entered by the user at the terminal. The picture will then be displayed at the terminal, and dynamic updating of the dynamic fields defined for that Picture will begin. The hierarchy of a picture is shown in Figure 7. A picture consists of one or more static definitions which will be overlaid on the screen. The picture also has one or more dynamic definitions, which in reality will be data structures that the graphics update software will use to display real-time data in the form of dynamic fields on the screen.

A dynamic definition may contain any combination of seven dynamic field types. The seven types are:

1. Dynamic text field.
2. Bar chart.
3. Numeric field.
4. Trend plot.
5. Dynamic symbol.
6. Time field.
7. Date field.

These seven types were determined to be necessary from the author's and others' experience with nuclear power plant requirements. Since the system may eventually be used in a nuclear power plant, the requirements were used in the coal-plant application.

A dynamic text field is a dynamic field that will display a specified text under the specified conditions. An example might be, if a process point were to exceed an alarm setpoint, the text "ALARM"

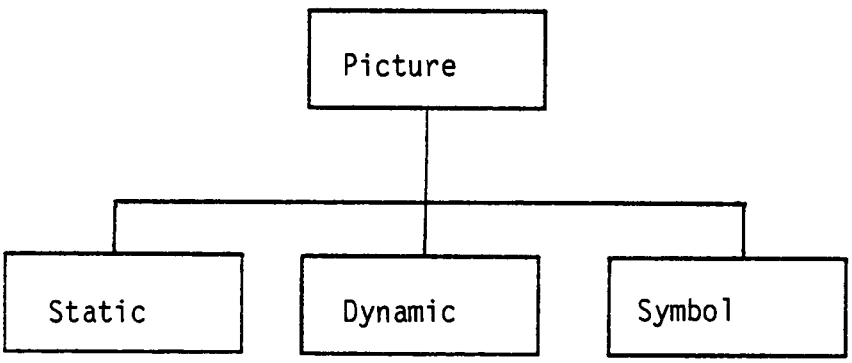


Figure 7. Picture hierarchy.

would appear in that field. A bar chart is a colored bar on the graphics screen whose value reflects the process point's value to which the field is assigned. A numeric field is similar to a dynamic text field except that a numeric field will contain a number that is equal to the process point value to which the field is assigned. A trend plot has been explained previously. A dynamic symbol is a graphics symbol defined on the Tesselator that assumes certain characteristics based on specified conditions. These characteristics can be size, color, etc. The date and time fields are similar to the dynamic text field except that the date and time will be updated in these fields. Specification of these fields will be discussed later.

The symbol definitions consist of the graphical symbols used in the static and dynamic portions of a picture.

This structure was determined to be the most flexible one while at the same time not overly complicating the software development process. The user can use the same static or dynamic definition with multiple pictures, thereby decreasing the time required to construct pictures.

User Interface

The design approach for the user interface for creating and editing pictures was to design and implement a program that would be completely interactive at the graphics terminal. This program is called the Picture Editor. The Picture Editor is represented by a process in the data flow diagram in Figure 4, page 6. The original idea was to have four parts to the Picture Editor: picture editing, static editing, dynamic editing, and symbol editing. This was to reflect the basic

structure of the graphics system. Because of schedule constraints, the picture and dynamic editing portion of the Picture Editor had to be dropped from the plans. Implementing an interactive interface that allowed the user to select any one of seven types of dynamic fields proved to be time-consuming since every type required a different software case. Picture and dynamic editing were implemented in another manner which will be discussed later.

The strategy the author adopted for providing a software interface with the Tesselator for the Picture Editor consisted of two types of software modules. The first type of module designed and implemented was a general-purpose type for implementing low-level utility functions. The second type consisted of modules specific to the Picture Editor program. A discussion of the second type will be deferred for now.

Utility Graphics Software

A set of necessary graphics utility functions was identified. This is the Graphics Terminal Interface shown in Figure 4 on page 6. The author then began the task of designing and implementing these modules while at the same time learning how to use the Tesselator. The set of functions is as follows:

1. Define an operator field.
2. Write to an operator field.
3. Read the contents of an operator field.
4. Define a computer field.
5. Write to a computer field.
6. Clear the screen.

7. Transmit a line of text to the screen.
8. Clear a line on the screen.
9. Read the contents of the work area.
10. Read the coordinates of the cursor position.
11. Protect a specified portion of the screen.
12. Read a function code from the keyboard.
13. Define a trend window.
14. Define a trend grid.
15. Define a trend.
16. Write a new trend value.
17. Draw a bar chart.
18. Display a dynamic text field.
19. Display a dynamic symbol.

Each of these functions was implemented as a FORTRAN subroutine. A naming convention of having the first two letters of the subroutine names be GF was adopted for software control. This set of utility functions has been found to be sufficient for all necessary manipulation of the terminal screens in this application. A description of each function will follow.

The operator field has been described previously. An operator field has certain characteristics associated with it. The subroutine GFDFOF, which is used to define the operator field, has the following arguments:

1. X coordinate of the left side of the operator field.
2. Y coordinate of the left side of the operator field.
3. The length of the operator field.

4. The foreground color of the operator field.

5. The background color of the operator field.

6. The symbol set to be used.

7. Array for storing the Tesselator codes.

8. The number of elements in the array.

9. The first available location in the array.

10. The next available location in the array after subroutine call.

The first two arguments simply tell the Tesselator where to create the operator field. The coordinates are specified in tessels. A tessel is the smallest distance increment on the terminal screen. There are 240 tessels in the X direction and 112 tessels in the Y direction. A tessel is a three-by-three block of pixels. The third argument is the length of the field in tessels. The length determines how many characters an operator will be able to enter in the field. The foreground color is the color of the characters in the field. The background color is the color of every part of the field except the characters. The symbol set is one of five available symbol sizes on the Tesselator. The characters entered in the operator field will have the attributes of the specified symbol set. Most of the utility graphics functions do not transmit data directly to the Tesselator. This was tried at first, but the speed at which graphical manipulations were accomplished was entirely too slow. It was found that storing the appropriate Tesselator codes in a single array to be transmitted to the Tesselator at one time greatly improved performance. The seventh argument is this array. The first available location pointer tells the subroutine where to start storing Tesselator codes in the buffer. The subroutine stores the codes

and then sets the ninth argument to the next available location in the array. This argument will then be used for the next utility routine to be called. This method has proved quite satisfactory since, in general, to accomplish a desired result on the screen, many calls to the utility routines must be made. In the future, the seventh, eighth, and ninth arguments will be referred to as terminal buffer information.

A subroutine for writing to an operator field is needed in some instances. In some applications a screen of operator fields with labels will be presented to the user. The operator fields will contain information which may be modified by the user. After modification, the operator fields will be read and processed. The arguments for the GFWROF subroutine are:

1. Text array containing characters to be written in the field.
2. X coordinate of the field.

The contents of the operator field are returned as symbol codes. For alphanumeric characters, the codes will be in standard ASCII form.

The subroutines associated with computer fields were designed and implemented, but it was later found that they were unnecessary for this application. The computer fields lack some characteristics that were deemed necessary and were therefore discarded in the design plans. The subroutines, however, are available for future use if needed.

Any graphics system needs to clear the terminal screen. The subroutine GFCLSC was designed and implemented for that purpose. The arguments are:

1. Logical terminal number.
2. Completion status.

The subroutine GFLNBF will construct a line of ASCII text for display on the terminal screen. This is useful for descriptions and headers to be displayed. The arguments for this subroutine are:

1. Array containing ASCII text to be displayed.
2. The symbol set to be used.
3. The foreground color of text.
4. The y coordinate of the line.
5. Terminal buffer information.

The GFCLLN subroutine clears a single line on the terminal screen. This is useful for eliminating operator fields in a display without disturbing the rest of the screen. The arguments are:

1. Logical terminal number.
2. Y coordinate of the line.
3. Terminal buffer information.

The GFRDWK subroutine reads the contents of the current work area on the terminal screen. A work area is defined as the portion of the screen that the user is allowed to manipulate. A work area can be defined as one of five types: (1) the entire screen, (2) a window, (3) a line, (4) a column, or (5) the operator fields. For most instances the graphics system software, the work area will be either a window or the operator fields. When the user constructs a static definition at the terminal, the lower three lines on the screen will not be available for use. This is necessary so that the command, dialog, and message lines will not be interfered with when a picture is selected. The user is prevented from working in this area by (1) defining the work area

as a window containing every portion of the screen except the bottom three lines and (2) setting the unprotected area to the work area. The GFRDWK subroutine reads the contents of this work area, which contains the static definition. The arguments for this subroutine are:

1. Logical terminal number.
2. Terminal buffer.
3. Number of bytes in display list representing the contents of the work area.

The GFPRO subroutine sets a specified window on the terminal screen as the unprotected area. The rest of the screen is protected from user manipulation. The computer can always write on the screen. The arguments for GFPRO are:

1. Logical terminal number.
2. Lower right x coordinate of the window.
3. Lower right y coordinate of the window.

The upper left coordinates of the window are always the upper leftmost corner of the screen. This is referred to as the home position.

The GFRDFC subroutine returns a function code from the user keyboard. The Tesselator keyboard has a feature known as function code. The function code is selected by pressing the "FC" key followed by 2 digits. The Tesselator sends the two digits to the host computer. The host computer software must have queued a read to the keyboard. The function code will then complete the read. Since the keyboards used in this system do not have a "RETURN" key, the function code is used frequently. The arguments for GFRDFC are:

1. Logical terminal number.
2. Hexadecimal function code.

Trend plots on the Tesselator require several commands to implement. A trend window must first be defined. The GFTRWI subroutine defines a trend window for a particular terminal. Up to 16 trend windows can be defined for one terminal. The window itself is not visible to the user. Each trend can be in a different window, or they can all be in one window. The arguments for this subroutine are:

1. The window number.
2. Upper left x coordinate of the window.
3. Upper left y coordinate of the window.
4. Lower right x coordinate of the window.
5. Lower right y coordinate of the window.
6. Overflow value flag.
7. Push flag.
8. Push all tessels flag.
9. Grid flag.
10. Number of tessels information in the window should be displaced by a push.
11. Map of trends (16-bit) to be included in window.
12. Terminal buffer information.

The overflow value flag indicates whether trend values that fall outside of the window will be displayed as a maximum window value or not at all. The push flag tells the Tesselator whether to push the contents of the trend window every time a new update occurs. The push all tessels flag indicates whether to just push the trend plot itself or everything in the window. The push can be visualized as a scrolling operation across the window. When a new value is to be

plotted, the oldest value in the window is scrolled out of the window. The grid flag tells the Tesselator if a grid background for the window is to be used. A grid is useful for observing particular points on the trend plot and estimating the time and value of it. The 16-bit map is simply a 16 bit-word with a bit allocated for each trend. If one desires to have trend number two included in this window, then the second bit should be a one.

The GFTRGD subroutine defines a grid background for a trend window. The arguments are:

1. Window number.
2. Pixel x coordinate of the grid's origin relative to the upper left x coordinate of the window.
3. Pixel y coordinate of the grid's origin relative to the upper left y coordinate of the window.
4. Color of grid.
5. The number of pixels between two vertical lines in the grid.
6. The number of pixels between two horizontal lines in the grid.
7. Total number of grid squares in the vertical direction.
8. Total number of grid squares in the horizontal direction.
9. Pixel distance between two marked points on the vertical grid lines.
10. Pixel distance between two marked points on the horizontal grid lines.
11. Terminal buffer information.

Only one grid per window may be defined.

Up to 16 trends per window may be defined. The GFDFTR subroutine defines a single trend. The arguments are:

1. The trend number.

2. Pixel x coordinate of the trend origin relative to the upper left corner of the window.

3. Pixel y coordinate of the trend origin relative to the upper left corner of the window.

4. Color of the trend.

5. Pixel distance between two trend values (step size).

6. Terminal buffer information.

The GFWRTR subroutine updates the trend with a new value. The value of the trend is specified from 20H to FFH. The real process value must be mapped into this range. The area of the trend must also be specified. The area is used to specify where the value will be plotted relative to the base line of the trend. The arguments for GFWRTR are:

1. The trend number.

2. The area.

3. The update value.

4. Terminal buffer information.

A subroutine was needed to display a bar chart. The GFBAR subroutine stores the appropiate Tesselator commands in the terminal buffer to display a bar. Each time a bar on the terminal screen is updated it must be completely redefined. This is not a serious problem since only 14 bytes of information are needed to define a bar.

The arguments for GFBAR are:

1. The bar direction.
2. The width of the bar in pixels.
3. The length of the bar in pixels.
4. The value of the bar in pixels.
5. The x coordinate of the bar.
6. The y coordinate of the bar.
7. The foreground color of the bar.
8. The color of the unused portion of the bar.
9. The blink attribute of the bar.
10. Terminal buffer information.

There are six available bar directions: (1) horizontal to the left, (2) horizontal to the right, (3) horizontal symmetric, (4) vertical up, (5) vertical down, and (6) vertical symmetric. Figures 8 and 9 show the possible configurations of the bar and defines the above arguments for the configurations.

The GFSTBX subroutines serve several functions in the graphics system. The mnemonic is derived from the term status box. This term is used in TECGRAPH to describe a dynamic text field with a colored box surrounding it. This is useful for highlighting certain dynamic fields on the terminal screen. This subroutine is used for all numeric, text, date, and time fields in the update software. If no "box" is desired, then a background color of black is specified, and only the characters will be seen. This is a simplification from earlier plans to have a separate subroutine for each dynamic field.

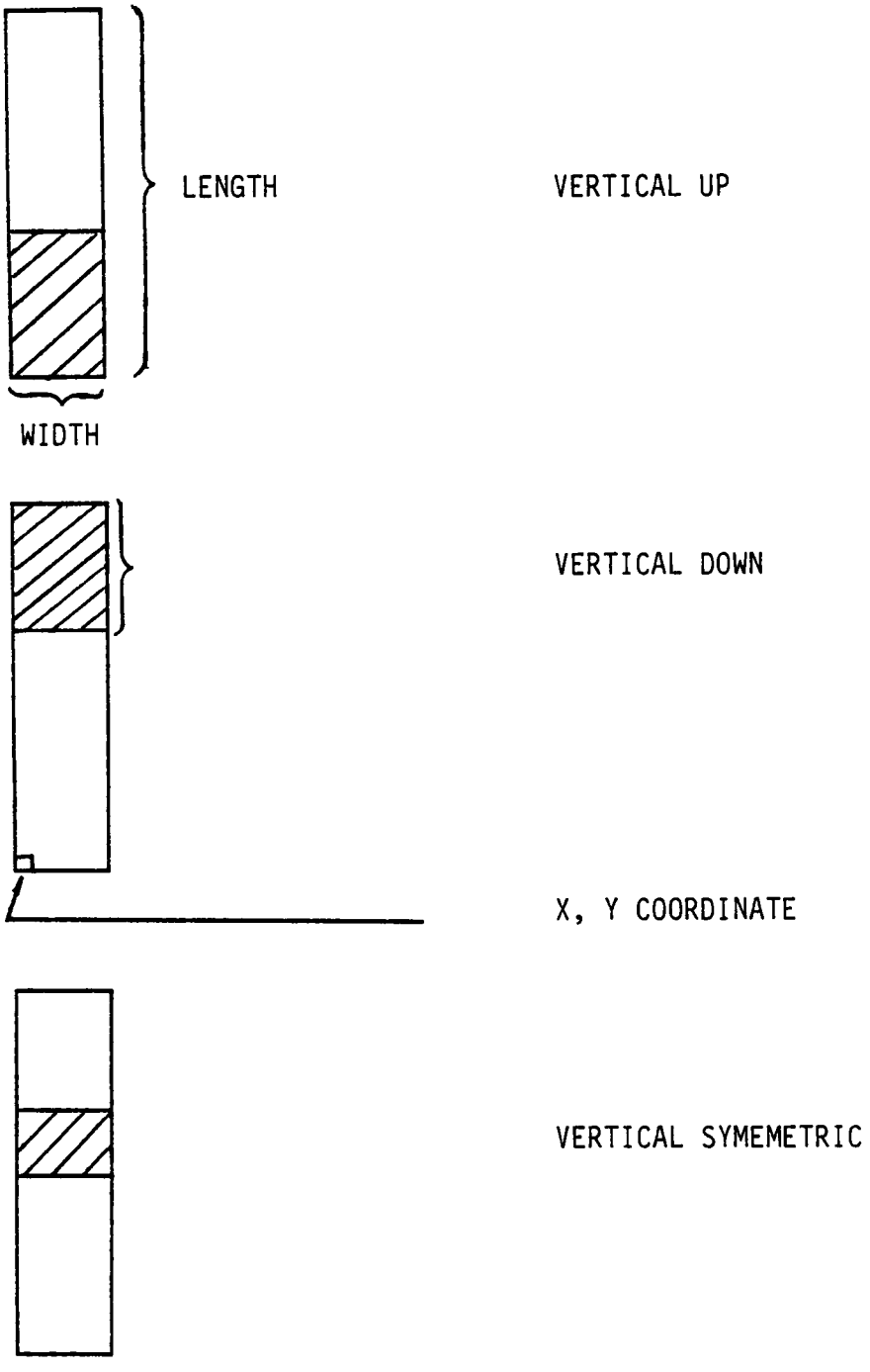


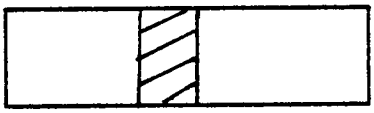
Figure 8. Vertical bar charts



HORIZONTAL TO THE RIGHT



HORIZONTAL TO THE LEFT



HORIZONTAL SYMMETRIC

Figure 9. Horizontal bar charts.

The arguments for GFSTBX are:

1. The upper left x coordinate of the box.
2. The upper left y coordinate of the box.
3. The lower right x coordinate of the box.
4. The lower right y coordinate of the box.
5. The writing mode for the text.
6. The text to be written.
7. The foreground color of the text.
8. The background color of the box.
9. The symbol set to use for the text.
10. The number of characters to be written.
11. Terminal buffer information.

There are six writing modes for this subroutine: (1) horizontal to the left, (2) horizontal to the right, (3) horizontal centered, (4) vertical up, (5) vertical down, and (6) vertical centered.

The last graphics utility function identified is the display of a dynamic symbol. An example of this would be a pump symbol that changes color depending on whether the pump is on or off. The GFSYMB subroutine fulfills this function. It is similar in nature to the GFSTBX subroutine. The arguments for the subroutine are:

1. The x coordinate of the symbol.
2. The y coordinate of the symbol.
3. The foreground color of the symbol.
4. The background color of the symbol.
5. The symbol set where the symbol is located.

6. The two-byte code that represents a symbol code or an operation to perform on a symbol.

7. Terminal buffer information.

Originally only a one-byte symbol code representing a particular symbol was to be allowed in this subroutine, but it was determined that an easy method of changing the color of large areas on a display could be accomplished using the dynamic color command on the Tesselator. The cursor can be positioned to the desired location on the screen and the dynamic color command can be issued, which will change the color of the symbols connected to that location. The term connected means that symbols such as lines, valves, and others, can be logically connected to each other when drawn by not leaving any space between the symbols. This variation has proved to be extremely useful for simulating flow in a pipe. The Tesselator has a color defined which alternates between red and yellow or dark blue and light blue. When the dynamic color command is used with these colors on a line, the line apppears to flow. In the monitoring system a dynamic symbol, can be defined at a certain location on a line, simulating a pipe in the plant. When the pipe is carrying steam or a liquid, the update software can call GFSYMB and change the color of that line to a dynamic color to simulate the flow.

Picture Editor

The other aspect of the software strategy for the user interface was to identify the structure of the Picture Editor program (PE). As has been mentioned previously, the Picture Editor consists of two main parts: static definition editing and symbol definition editing. The

software design strategy was based on designing a different subroutine for each logical division in the functionality of the program. The PESTPR subroutine was identified as the top level of the static definition editing portion of the Picture Editor. The PESYPR subroutine was identified as the top level of the symbol definition editing portion of the program. Figure 10 shows the structure of both the software and the functionality of the Picture Editor. Before continuing the software description of the Picture Editor, the functional requirements of the program will be described.

As has been mentioned previously, the Picture Editor needed to be interactive to allow easy creation of static definitions. The use of menus to guide the user through the available commands was determined to be the most efficient based on previous operator experience and current trends in industry. When the Picture Editor program is executed at the graphics terminal, the terminal screen is cleared, and a menu with the following three commands is displayed:

1. STATIC
2. SYMBOL
3. EXIT

These commands correspond to the top level functionality shown in Figure 10. The user may select one of these commands by positioning the cursor to the option on the screen and entering "FC 99" at the keyboard. If STATIC is selected, another menu is placed on the screen. This menu contains the commands for editing static definitions. The commands are:

1. GET

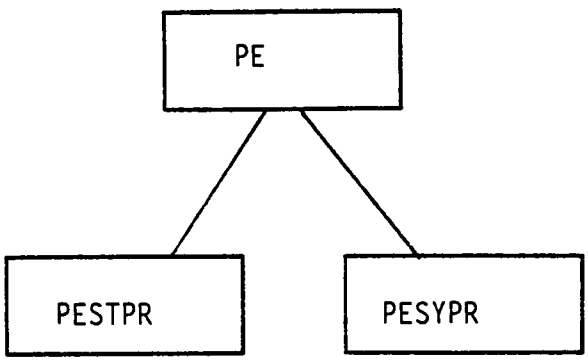


Figure 10. Picture Editor structure.

2. SAVE

3. LIST

4. DELETE

5. EXIT

The GET command allows the user to retrieve a particular static definition from the picture database and have it displayed at the terminal. This command can be executed several times so that several static definitions can be overlaid at once.

The SAVE command allows the user to store a static definition in the picture database. When SAVE is selected the Picture Editor prompts the user for the name that will reference the static definition in the database.

The LIST command allows the user to display a list of static definitions currently catalogued in the database.

The DELETE command allows the user to delete a static definition from the database.

The EXIT command returns the user to the previous level of the Picture Editor. This level is the one with the STATIC, SYMBOL, and EXIT commands.

If the SYMBOL command is selected, the Picture Editor executes the Symbol Editor program that is resident in the Tesselator. The user can then use the Symbol Editor's capabilities to edit or create a custom symbol. When the user is through editing symbols a "FC 99" should be entered on the keyboard to signify to the Picture Editor that the user

is through editing symbols. The Picture Editor displays a menu with the following commands:

1. SAVE
2. EXIT

If the user selects the SAVE command, the Picture Editor sends command a to the particular Tesselator control unit to read up the symbol set. The symbol set is a self-contained buffer of Tesselator symbol codes. This buffer is stored in the picture database. The symbol set is downloaded to the Tesselator each time the graphics software is initialized. There is only one symbol set for each Tesselator control unit, even though up to three terminals may be connected to the control unit. If EXIT is selected, the symbol set is not stored and the Picture Editor returns to the top-level menu.

Picture Compiler

The Picture Compiler is a program designed to compile a source file consisting of graphics definitions to build a picture for display. The Picture Compiler was designed and implemented for a RAMTEK-based color graphics system in an Emergency Response Facility for Florida Power and Light's nuclear plants. The original Picture Compiler was used to construct both the static and dynamic portions of the pictures. The dynamic part of it was retained for this application. The data structures used for the dynamic fields were changed, but the basic mechanism of the Picture Compiler remained the same.

The Picture Compiler program runs as a background task on the MODCOMP computer. The graphics source file is used as the input for the program. The source file consists of graphics mnemonics designed

by TEC. Figure 11 shows a sample graphics file for specifying dynamic fields. A detailed discussion of the meaning of the elements in Figure 11 will be postponed until the display specification has been described.

Display Specification

Specifying displays to be shown at the graphics terminals consists of defining (1) picture definitions, (2) static definitions, (3) dynamic definitions, and (4) symbol definitions. The specifying of the static and symbol definitions has been described previously. Specification of picture definitions and dynamic definitions will now be discussed.

Picture Specification

The picture definition is specified in a source file as shown in Figure 12. The picture definition name is an eight-character ASCII name that will be used as a command for a user to select the picture once the picture definition is complete. The access rites consist of a 16-bit word with bits corresponding to certain plant conditions for which the plant operators are allowed to select the picture for display. For this system bits five through eight correspond to plant modes four through one. Bits 13 through 16 correspond to plant numbers. If the bit is set, it indicates that the picture is valid for this condition. Up to eight static definition and dynamic definition names are then specified. These are the static and dynamic definitions that will be used by the graphics update software to display the picture. The END signifies to the Picture Compiler that this is the end of the picture definition source file.

```
DYNAMIC [dynamic definition name],


TEXT,
     P, F804,
     126, 92, 145, 93,
     2,
     Z8001, Z8001, 1, 0, HC, 2, 'CLOSED',
     Z8001, Z8000, 3, 0, HC, 2, 'OPEN'


BAR,
     P, H107,
     16, 37, 106, 38,
     HR,
     LIN, 0.0, 1200.,
     2,
     Z803F, Z8000, 3, 0,
     Z8110, Z8000, 1, 0


END
```

Figure 11. Typical dynamic field specification.

```
PICTURE, [picture name], [access rites],

        [static definition names],

        [dynamic definition names]

END
```

Figure 12. Picture definition specification.

Picture Data Structures

The picture definition is specified in a source file and compiled by the Picture Compiler. After compilation, the picture definition consists of an entry in the Picture Directory file and a corresponding entry in the Picture Definition file. The Picture Directory file contains the following information:

1. Picture definition name.
2. Picture definition record number.
3. Length of the picture definition.
4. Access rites of the picture.
5. Entry update flag.

The picture definition name is an eight-character packed ASCII array. The picture definition record number is the record number of the corresponding picture definition record in the picture definition file. The length of the picture definition is the number of words in the picture definition record. The entry update flag indicates whether the picture definitions residing in main memory need to be reloaded. This occurs when a picture definition is created or edited. The picture definitions are kept in main memory for faster access times.

The picture definition file contains the following information:

1. Up to eight static definition names.
2. Up to eight dynamic definition names.

The static and dynamic definition names consist of eight ASCII characters. When the picture is selected by the user, the graphics update software searches through the picture definition directory for

the requested picture. If the picture is found, then the update software goes to the appropriate location in the picture definition file and sends up to eight static definitions to the terminal screen and builds data structures for updating up to eight dynamic definitions.

Static Definition Specification

The static definition is specified by drawing it on the Tesselator terminal as has been described previously.

Static Definition Data Structures

The static definitions consist of an entry in the static definitions directory and a corresponding record in the static definitions file. A static definition record is the Tesselator codes sent by the Tesselator when the subroutine GFRDWK is called by the Picture Editor. The static definitions directory has the following information for each entry:

1. Eight-character static definition name.
2. Static definition record number.
3. Length of the static definition record in words.
4. Entry update flag.

These parameters are similar to the picture definition directory parameters.

Dynamic Definition Specification

Each dynamic definition is specified by a source file which is compiled by the Picture Compiler at run-time. The source file for a dynamic definition is shown in Figure 13.

[dynamic field type],

[variable type], [variable ID],

[window coordinates],

[number of update options],

[status word mask], [status word compare value],

[foreground color], [background color], [write direction],

[symbol set], [other information]

Figure 13. Dynamic field specification.

There are seven different dynamic fields available for use. Figure 14 shows the specification of the dynamic text field. The variable type is one of three types: (1) P for process variable, (2) S for system variable, and (3) T for terminal variable. A process variable is a point in the system database that reflects a point in the plant. A system variable is one that may reflect information such as time and date. A terminal variable reflects information available in the terminal common block. The variable ID is the identifier for the variable that is "tied" to this field. The coordinates are for the dynamic text window. Up to eight update options can be specified. The status word mask is a 16-bit word that is used as a mask for the process point's 16-bit status word. This allows only the desired bits of information from the status word to affect the dynamic field. The status word contains bits indicating such information as whether a point is in alarm or not or whether a point is active. The status word compare value is a 16-bit word used for comparison with the masked version of the status word. If the masked version of the status word is equal to this compare value, then the update option is used for updating the dynamic text field. This allows the dynamic text field to assume up to eight different characteristics depending on plant conditions.

Figure 15 shows a typical dynamic text field specification. The specification indicates that the dynamic text field is to be "tied" to the process point F804. Its window coordinates are 126, 92, 145, and 93. There are two update options. The status word mask for both options. is the hexadecimal number 8001. The comparable value for the two

TEXT,

[variable type], [variable ID],

[text window coordinates],

[number of update options],

[status word mask], [status word compare value],

[foreground color], [background color], [write direction],

[symbol set], [ASCII text]

Figure 14. Dynamic text field specification.

```
TEXT,

     P, F804,

     126, 92, 145, 93,

     2,

     Z8001, Z8001, 1, 0, HC, 2, 'CLOSED',

     Z8001, Z8000, 3, 0, HC, 2, 'OPEN'
```

Figure 15. Typical dynamic text field specification.

options are 8001 and 8000. The first option is used if the last bit in the status word is a one. This would indicate that the valve is closed. The text "CLOSED" would appear on the screen in red letters with a black background. HC indicates that the text is to be centered horizontally in the window. Symbol set two should be used to write the text. The second option indicates that the text "OPEN" should be displayed in green letters.

The dynamic bar chart is specified in a similar manner as the dynamic text field. Figure 16 is the specification for a bar chart. The main difference between it and the text field is that a scale is specified for the bar chart. It can be a linear or logarithmic scale. Figure 17 shows a typical bar chart specification. It has a linear scale with a minimum value of 0 and a maximum value of 1200.

The dynamic numeric field specification is shown in Figure 18. The format is specified the same way FORTRAN numeric formats are specified. Default update options can be specified in place of the status word mask and compare values and associated colors. This is useful for making all the numeric fields in the system have the same characteristics. Figure 19 shows a typical dynamic numeric field.

At this time, the dynamic trend specification has not been implemented. It will look similar to the other specifications when completed.

The dynamic symbol specification is shown in Figure 20. A typical symbol specification is shown in Figure 21.

The time and date dynamic fields are special cases of the dynamic text fields. Figure 22 and Figure 23 show the time and date field specifications respectively. The format can be one of several time

BAR,

[variable type], [variable ID],

[window coordinates],

[bar direction],

[linear or log scale], [minimum value], [maximum value],

[number of update options],

[status word mask], [status word compare value],

[foreground color of bar], [color of unused portion of bar]

Figure 16. Dynamic bar chart specification.

```
BAR,

     P, H106,

     16, 37, 106, 38,

     HR,

     LIN, 0.0, 1200.0,

     2,

     Z803F, Z8000, 3, 0,

     Z8110, Z8000, 1, 0
```

Figure 17. Typical dynamic bar chart specification.

```
NUMERIC,

     [variable type], [variable ID],

     [window coordinates],

     [write direction], [symbol set],

     [status word masks], [status word compare values],

     [foreground colors], [background colors],

     [format field], [format]
```

Figure 18. Dynamic numeric field specification.

```
NUMERIC,

     P, F145,

     126, 92, 145, 93,

     HL, 1,

     DEFAULT, F, 6.1
```

Figure 19. Typical dynamic numeric field specification.

```
SYMBOL,

     [variable type], [variable ID],

     [x coordinate], [y coordinate],

     [number of update options],

     [status word masks], [status word compare values],

     [foreground colors], [background colors],

     [symbol set], [symbol code]
```

Figure 20. Dynamic symbol field specification.

```
SYMBOL,

     P, F146,

     126, 92,

     2,

     Z8001, Z8001, 12, 0, 1, Z1B22,

     Z8001, Z8000, 3, 0, 1, Z1B22
```

Figure 21. Typical dynamic symbol field specification.

```
TIME,

     [variable type],

     [window coordinates],

     [write direction], [symbol set],

     [number of update options],

     [status word masks], [status word compare values],

     [foreground colors], [background colors],

     [time format (HMS, HM)]
```

Figure 22. Dynamic time field specification.

```
TIME,

     I,

     126, 92, 136, 93,

     HR, 1,

     2,

     Z8001, Z8001, 3, 0, HMS,

     Z8001, Z8000, 7, 0, HM
```

Figure 23. Typical dynamic time field specification.

and date formats such as hours/minutes/seconds and month/day/year. Typical time and date field specifications are shown in Figures 24 and 25 respectively.

Dynamic Definitions Data Structures

The dynamic definitions directory has entries defined similar to the static definitions directory except the names are of dynamic definitions. The dynamic definitions are specified in a source file and compiled by the Picture Compiler at run-time into dynamic definition records. A dynamic definition record consists of up to 75 update record templates. There is an update record template for every dynamic field in the definition. There are seven different types of update record templates corresponding to the seven dynamic field types.

The dynamic text update record template consists of the following information:

1. Record type (1 for dynamic text).
2. Upper left x coordinate of the text window.
3. Upper left y coordinate of the text window.
4. Lower right x coordinate of the text window.
5. Lower right y coordinate of the text window.
6. Text foreground color.
7. Text background color.
8. Write direction.
9. Symbol set.
10. Number of characters to display.
11. ASCII text to be displayed.

```
DATE,

     [variable type],

     [window coordinates],

     [write direction], [symbol set],

     [number of update options],

     [status word masks], [status word compare values],

     [foreground colors], [background colors],

     [date format (M/D/Y, Y/M/D, D-M-Y)]
```

Figure 24. Dynamic date field specification.

```
DATE,

     I,

     126, 92, 136, 93,

     HR, 1,

     2,

     Z8001, Z8001, 3, 0, M/D/Y,

     Z8001, Z8000, 7, 0, D-M-Y
```

Figure 25. Typical dynamic date field specification.

Each update option has an update definition record associated with it. Each update definition record for the dynamic text field has the following information:

1. The status word mask.
2. The status word compare value.
3. The text to be stored in the update record template if this update definition record is used.

The dynamic bar chart has the following update record template:

1. The record type (2 for dynamic bar chart).
2. The bar height.
3. Upper left x coordinate of the bar.
4. The upper left y coordinate of the bar.
5. The lower right x coordinate of the bar.
6. The lower right y coordinate of the bar.
7. The foreground color of the bar.
8. The color of the unused portion of the bar.
9. The bar direction.

The update definition record for the bar is as follows:

1. The status word mask.
2. The status word compare value.

The dynamic numeric field update record template is the same as the dynamic text field template. The update definition record for the numeric field contains the following information:

1. Status word mask.
2. Status word compare value.
3. The text foreground color.

4. The text background color.

5. The format (1=I, 2=F, 3=E, 4=A).

6. The field width in characters.

7. The number of places to the right of the decimal point.

8. The ASCII text to be placed in the field.

The dynamic trend plot update record template and update definition record format have not yet been designed.

The dynamic symbol update record template contains the following information:

1. Record type (5 for dynamic symbol).

2. Symbol x and y coordinates.

3. Symbol foreground color.

4. Symbol background color.

5. Symbol set.

6. Symbol code.

The dynamic symbol update definition record contains the following information:

1. Status word mask.

2. Status word compare value.

3. Symbol foreground color.

4. Symbol background color.

5. Symbol set.

6. Symbol code.

The dynamic time and date fields have the same update record template as the dynamic text field. The update definition record for the time and date field contains the following information:

1. Status word mask.
2. Status word compare value.
3. Text foreground color.
4. Text background color.
5. Time or date format.
6. Number of characters for A format.
7. ASCII text for A format.

CHAPTER V

REAL-TIME UPDATING OF DISPLAYS

The dynamic fields in a picture are updated continuously as new data is acquired by the Plant Monitoring System. The author adapted the Update Task for this system using the utility subroutines mentioned previously. Another member of the project adapted the Display Activate task for the system. These tasks were in use with a somewhat different graphics system at TEC using the RAMTEK graphics devices. The basic structure of the tasks did not change significantly in the transition.

The Display Activate task is activated any time a "select picture" command is entered at the terminal. The task then reads the command string to determine if the string is the name of a valid picture. If it is, the task reads up the static display produced by the Picture Editor and the dynamic definitions produced by the Picture Compiler. From this information, the Display Activate task forms the update definition records for the various dynamic fields and transmits the static definition to the Tesselator. The Update task then reads the update definition records and constructs a buffer of Tesselator codes to be sent to the Tesselator. The Update task uses a subset of the graphics utility functions to construct the buffer. GFSTBX is used for the numeric, dynamic text, date, and time fields. GFSYMB is used for the dynamic symbol field. GFBAR is used for the dynamic bar chart. The update definition records are produced by the Update task

from the update options and the dynamic fields directory for that picture. The Display Activate task loads those data structures when the picture is selected.

The trends are constructed and displayed by the other tasks in Figure 4 on page 6. These tasks are not yet finished. The set of graphics utility routines consisting of GFTRWI, GFTRGD, GFDFTR, and GFWRTR, designed and implemented by the author, will be used by these tasks.

CHAPTER VI

SYSTEM LIMITATIONS

There are two main limitations in using the Tesselator for a color graphics system. The Tesselator does not have the capability of drawing arbitrary vectors. It would be desirable to specify a starting and ending coordinate and draw a straight line between the coordinates. This is necessary for doing parameter-versus-parameter plots. The second limitation is that the Tesselator is not software-programmable. This would help decrease the cost of an overall monitoring system by not requiring a separate host computer. Despite these limitations, no other graphics system had better credentials than the Tesselator.

CHAPTER VII

CONCLUSIONS

Though the color graphics system is not presently in its final form, it is on-site and functioning at Carolina Power and Light's Unit One facility at Skyland, North Carolina. The trend plot software is presently being developed and should soon be available. The author became familiar with, and hopefully in some areas advanced, the state-of-the-art in color graphics in monitoring systems. This system is competitive with many of the systems produced by larger companies such as Honeywell, Foxboro, and Aydin. This color graphics system will remain as the base for future plant monitoring systems for Technology for Energy Corporation.

LIST OF REFERENCES

LIST OF REFERENCES

1. Page-Jones, Meilir. The Practical Guide to Structured Systems Design. New York: Yourdon Press, 1980.

APPENDIX

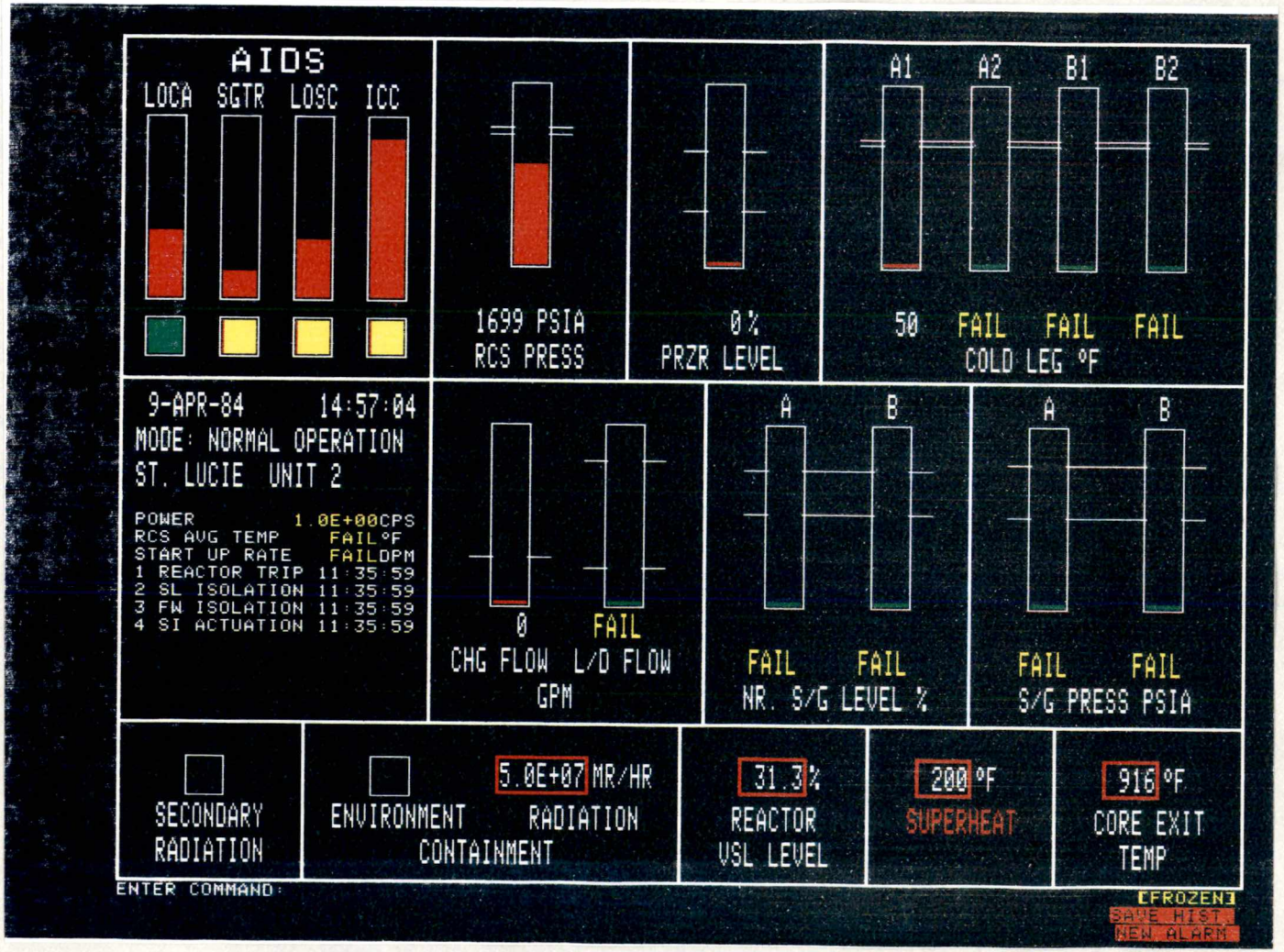


Figure 26. Example of a Color Graphics Display

VITA

Robert Gregg Majure was born in Jackson, Tennessee on January 26, 1960. He was graduated from Cheatham County High School, in Ashland City, Tennessee, in June 1978. He entered Tennessee Technological University in June 1978 and began study toward a Bachelor of Science degree in Electrical Engineering. This degree was awarded in June 1982.

He began work as an Engineer with Technology for Energy Corporation in Knoxville, Tennessee. In September 1982 he began study toward a Master of Science degree in Electrical Engineering at The University of Tennessee, Knoxville. He received this degree in June 1984.

The author is a member of Eta Kappu Nu, Tau Beta Pi, and IEEE. He was named as one of the top ten electrical engineering students in the United States in 1982 by Eta Kappa Nu.