MITRE Technical Report

MTR-4725

Trend Monitoring System (TMS) Graphics Software

J. S. Brown

APRIL 1979

CONTRACT SPONSOR	NASA/JSC
CONTRACT NO.	F19628-79-C-0001 T5295F
PROJECT NO.	8470
DEPT.	D72

THE MITRE CORPORATION

HOUSTON, TEXAS

Department Approval: Edwin S. Hamilton

MITRE Project Approval: Edwin S. Hamilton

ABSTRACT

At NASA's Johnson Space Center MITRE has installed a prototype bus communications system, which is being used to support the Trend Monitoring System (TMS) as well as for evaluation of the bus concept. As a part of the work on the TMS project, MITRE implemented a set of FORTRAN-callable graphics subroutines for the host MODCOMP computer, and designed an approach to splitting graphics work between the host and the system's intelligent graphics terminals. This document describes the graphics software in the MODCOMP and the operating software package written for the graphics terminals.

This page intentionally
left blank

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

This page intentionally
left blank

GLOSSARY

Completion Time	Time between request for a function to be performed and the completion of that function.
Curve	In TMS, the locus of one measurement in a plot. Up to 6 curves may be displayed in a plot.
Display	A complete picture on the graphics terminal screen. In this document, used synonomously with "plot".
Display List	A series of display processor instructions which specify what the display processor causes to appear on the screen. Generally broken into pictures.
Display Processor	The MG-552 component of the MEGATEK terminals, which runs asynchronously to the terminal's NOVA and which draws and refreshes the display.
Graphics Software	FORTRAN-callable routines in the MODCOMP which are used to build a display on the graphics terminal.
Graphics Terminal	A TMS MEGATEK 5000 intelligent graphics unit.
Picture	A contiguous subpart of a display list composed of display processor instructions which define a logical subportion of the entire display.
Plot	See "display".
Response Time	Time between request for a function to be performed and the beginning of the response to the function.
Terminal Program	The complete set of software which executes in a TMS graphics terminal, and which both builds displays and responds to keyins by the terminal user.

This page intentionally
left blank

TREND MONITORING SYSTEM (TMS) GRAPHICS SOFTWARE

SECTION I
INTRODUCTION

1.0 BACKGROUND

At NASA's Johnson Space Center (JSC) in Houston, Texas, the Orbiter Data Reduction Complex (ODRC) has the responsibility of handling non-real-time data reduction for measurements gathered during tests of manned spaceflight missions, such as the Space Shuttle. The data reduction involves extraction of requested data from magnetic tapes, calibration of the raw measurements and conversion to engineering units, and display of the data in any of a variety of output formats and media. For the most part, the ODRC's workload is generated from written processing requests and has a desired turnaround time ranging from several hours to several days.

In the course of planning ODRC support for the Operational Flight Tests (OFT) of the Space Shuttle, it was recognized in 1977 that a need existed to monitor certain thermal parameters of the vehicle in near real time. NASA/JSC's Engineering and Special Development Branch (FD7) of the Institutional Data Systems Division (IDSD) consequently designed and implemented an interactive graphics system built around a MODCOMP IV/35 host minicomputer and several MEGATEK 5000 intelligent graphics terminals. The system, termed the Trend Monitoring System (TMS), gathers data samples from tapes as they become available and then presents plots of measurements against time, as the plots are requested by users at the terminals.

One of the constraints of the TMS was that the terminals and the host computer would have to be separated by a distance of about 1600 feet, and that furthermore, the speed of data transmission would have to be high. Plots may be composed of as many as 25,000 bytes of data and must be displayed in a few seconds. Conventional communications systems to meet these requirements of a high data rate over an extended distance are not readily available.

MITRE has developed a coaxial cable bus communications system [1], however, which can provide a digital communications bandwidth of up to 307.2 Kbps over a distance of several miles. Consequently, NASA chose to implement a bus system to support the TMS for two reasons: (1) to meet the communications need for the TMS, and (2) to provide a test bed for evaluating the bus concept for IDSD communications.

MITRE has provided both engineering and implementation support for the hardware and software required to support the prototype bus. This work is documented in a series of reports ([2], [3], [4], [5], [6], [7], [8]).

As a collateral effort to implementation of bus hardware and software for the TMS, MITRE was assigned the responsibility of determining what portion of the graphics functions should be performed in the host computer and what portion in the intelligent terminals (which are based around Data General NOVA/3 minicomputers and which can be programmed). MITRE was also tasked with the design and implementation of the graphics software in both the MODCOMP and the MEGATEK computers.

1.1 Functional Requirements for Graphics Software

The functional requirements for the TMS graphics software fall into two separate classes: (1) requirements for the MODCOMP application program graphics interface, and (2) requirements for the graphics terminal human interface.

When the TMS was initially planned, studies and tests (such as [9]) were conducted using a then-available IMLAC graphics terminal. Several demonstration programs were written to use the GRAPHELP graphics support package designed by the Harry Diamond Laboratories for the IMLAC [10]. As system development for the TMS continued, it became clear that the ultimately selected terminals for the system might not be IMLAC terminals, but in order for applications software development to proceed during terminal procurement, in October 1977 a graphics interface based on the GRAPHELP package was defined. The interface was composed of a subset of the GRAPHELP calls, augmented by several special entry points (such as BIAS, CURVE1, and

CURVE2) defined specifically for the TMS environment. This interface for all applications programs is included as Appendix I to this document.

The user interface to the graphics terminal was likewise defined at an early stage of system development. Requirements analysis indicated that the functions of displaying of plots and command entry were needed, together with annotation of the plot by user typing and full editing capability (cursor movement, tabulations, character insertion and deletion, etc.) for all keyed input. A need for a capability to slide (skew) certain plots with respect to other plots was also identified, as was a requirement for a joystick which could be used to determine coordinate values of particular points on the screen. These and other user interface requirements were documented in specifications such as [11] and [12] and are summarized in Figure 1.1-1.

Editing

- Provide cursor movement, character insertion and deletion, and line deletion

Plot Display

- Construct plots of thermal data from lists of data points
- Upon command, slide (skew) designated curves in the X-direction and update a corresponding bias time
- Provide a shorthand method of requesting the next plot group in a sequence

Plot Annotation

- Provide for user typing of comments on the plot
- Provide for tabulation among the comments

ASCII Messages

- Retain the last 20 lines of ASCII command input and ASCII output in a scrolled display

Joystick

- Provide a graphics cursor controlled by the terminal's joystick
- Provide a facility to determine the user coordinates of a point on the display from the joystick cursor position

Copying and Display Control

- Provide a means for transferring the screen contents to a VERSATEC hard copier
- Provide a means to make the ASCII messages and/or the plots temporarily invisible upon command

Figure 1.1-1 Overview of TMS Graphics Software Functional Requirements

A further functional requirement was placed on the TMS by the specification that output from any plot request (which may result in up to 5,000 data points) must begin to appear on the screen within 15 seconds ([12], paragraph 3.3.7]). No statement of completion time (time to fully satisfy the plot request) was made in TMS requirements. This response time requirement together with the interfaces mentioned above, provided the framework within which the developed graphics and terminal software were constrained to operate.

1.2 Document Purpose and Overview

This report documents the general structure of the MODCOMP graphics software (termed "graphics software" in the remainder of the paper) and of the MEGATEK terminal program (termed "terminal program"). Listings, detailed flow diagrams, and module interface definitions are given in [13]. The purpose of the document is to provide an introduction to the graphics software and terminal program and to discuss in detail certain background information necessary to maintain or modify the programs. Since the graphics functions supported both in the MODCOMP and the MEGATEKs are general purpose, the software can be used in the future to support applications other than TMS, if desired.

Section 2 deals with considerations involved in the decision of how to split the graphics workload between the MODCOMP and the MEGATEKs and also describes the general protocol designed for communication of requests and graphics commands between the two machines. (The TMS includes, of course, lower levels of protocol [invisible to the graphics software and terminal program] for accomplishing actual transfers of data; these other protocols are described in detail in [4], [5], and [6].)

Section 3 documents the structure and operation of the MODCOMP graphics software, while Section 4 deals with the terminal program. Appendix I contains, as mentioned, a description of the MODCOMP application program interface to the graphics software, and Appendix II gives details of the graphics pictures used by the terminal program.

SECTION II
TMS WORKLOAD DISTRIBUTION

2.0 DESIGN CONSIDERATIONS

Although the TMS has several functions, such as loading of data tapes, production of data availability reports, archiving and updating data bases, and producing plots of thermal data, requirements studies indicated that by far the most frequently used function would be display of plots. Investigations during the conceptual design phase of the system [9] indicated that most of the requirement for CPU processing power would come from producing plots. The fetching of data for the plots from the TMS data bases was anticipated to require ten or fewer disk I/Os per plot. It was judged that the principal scarce resource on the MODCOMP was likely to be CPU time.

A further consideration in the decision of how to split the graphics processing workload between the MODCOMP and the MEGATEKs was a preference to build the system with some capability to add additional graphics terminals, if desired. This expansion can be absorbed only if the MODCOMP CPU is less than fully utilized with the initial complement of terminals.

These desires, together with an intention to arrange the workload split so as to meet a response time goal of 15 seconds for the most complex plots, indicate that as much of the plot processing as feasible should be off-loaded onto the intelligent terminals.

The analysis in paragraph 2.1 shows how time estimation was done during the study of how to split the workload. Paragraph 2.2 contains a discussion of how memory constraints affect the workload split decision. How the workload is split in the TMS is briefly stated in paragraph 2.3, and the protocol used to support the split is presented in paragraph 2.4.

2.1 Completion Time Estimates

A part of determining how to split the workload between the MODCOMP and the graphics terminals is the estimation of the time required to produce a display. Paragraph 2.1.1 outlines an estimate of the CPU time for producing

a display, and paragraph 2.1.2 gives an approximation of transmission time for the display commands. A summary of the estimated time to complete a display is then developed in paragraph 2.1.3.

2.1.1 CPU Calculations Required

The principal computation to be done on each of the 5,000 points in the maximally complex display is the conversion of data from user units (such as degrees Fahrenheit and time) to raster positions on a display screen, and the generation of a display list command for each point. The data conversion in its most simplified form takes the form of the following equation:

NRASTERS = (USER UNITS - ORIGIN VALUE) * SCALING FACTOR

where NRASTERS = number of rasters (integer)

USER UNITS = value of one coordinate of one of the 5,000 data points

ORIGIN VALUE = user units value associated with the plot origin

SCALING FACTOR = number of rasters per user unit

This conversion must be done for both the X and Y coordinates of each of the 5,000 points, or for a total of 10,000 values.

Since the values on the right side of this equation are all MODCOMP floating point numbers, the equation requires a floating point difference and a floating point multiply, plus a floating-point-to-integer conversion if the calculation is done on the MODCOMP. If the calculation is done on the NOVA in the MEGATEK, on the other hand, an additional conversion (of USER UNITS) from MODCOMP floating point format to NOVA floating point format is required before the equation can be evaluated (the other floating point values are assumed to be converted only once per plot).

The speeds and capabilities of the two processors for this type of calculation differ, as summarized in Table 2.1-I.

Table 2.1-I
MODCOMP and NOVA CPU Power Comparisons

Factor	MODCOMP	NOVA
Memory cycle time	500 ns	700 ns
Floating point multiply	5.22 μs	250 μs[1]
Floating point difference	6.24 μs	150 μs[1]
Conversion from own floating point to integer	6.96 μs	100 μs[1]
Conversion to NOVA floating point from MODCOMP floating point	---	34 μs[2]

NOTES: [1] Estimated from FORTRAN benchmark, which uses library subroutines for floating point operations

[2] Estimated from instruction timings in conversion routine

A rough estimate of the minimum processing time f 5,000 data points based on these figures is then 0.2 seconds for MODCOMP processing and 5.3 seconds for NOVA processing. To these numbers should be added (for estimation) about 0.7 seconds for other MODCOMP processing (loop control, I/O wait time, calculation time other than in the main loop, etc.). Other NOVA processing, such as for display list housekeeping, requires about 2 seconds for a maximal plot.

2.1.2 Transmission Time

A second component of the response time is the time to transmit data between the MODCOMP and the MEGATEK. The maximum aggregate throughput of the communications bus is 307.2 Kbps, but because of handler delays, network contention, application program time between I/Os, etc., the maximum effective rate is no greater than about 200 Kbps for a single MODCOMP user program writing data to a graphics terminal over the bus. If data conversion is done in the NOVA, two floating point values (of 32 bits each) must be transmitted for each data point; if conversion is done in the MODCOMP, on the other hand, two integer values (of 16 bits each) must be sent. In either case, about 5 16-bit words of overhead are required for every 60 16-bit words of data. Transmission of the data then requires about 0.9 seconds if conversion is done in the MODCOMP (about 10,800 words) and 1.7 seconds if conversion is done in the NOVA (about 21,700 words). To each of these figures should be added about 0.2 seconds for transmission time of plot background information (grids, axes, etc.). These figures for data transmission speeds all assume that the recipient of a message can absorb the message as quickly as the sender can generate the message, so that the sender is not kept waiting.

2.1.3 Time Calculations

The foregoing rough analysis is summarized in Figure 2.1-1 and indicates tnat if I/O and CPU work are not overlapped, a 5,000-point plot would require about 4 seconds if data conversion were done on the MODCOMP and 10 seconds if conversion were done on the NOVA. The TMS is designed to process each plot transaction serially (see [11]) so that one plot is completed

before another is begun; as a result, it is estimated that overlap effects would reduce these times by less than 20%. The figures indicate that either method is likely to meet the stated response time (in both cases the completion time, which is what is shown in Figure 2.1-1, is significantly less than the required response time). Performing data conversion on the MODCOMP, however, would result in a noticeably faster plot completion.

Component	Conversion Done on MODCOMP	Conversion Done on NOVA
Raster calculation (CPU time)	0.2 sec (MODCOMP)	5.3 sec (NOVA)
MODCOMP overhead	0.7	0.7
Transmission time (data)	0.9	1.7
Transmission time (background)	0.2	0.2
NOVA overhead	2	2
Total	4. sec	9.9 sec

Figure 2.1-1
Summary of Completion Time Estimates for a Maximal Plot (No Overlap Assumed)

Reference [12] indicates that the TMS is designed to support up to 8 terminals (paragraph 2.2), each of which may be used to produce up to about 400 displays per 8-hour shift (paragraph 3.3.6). This workload corresponds to about 400 plot requests per hour, or about one request every 9 seconds. The MODCOMP time for the maximally complex plot (the sum of the times shown in the first 4 lines of Figure 2.1-1), even if no overlap is obtained, should be less than 3 seconds, regardless of where the data conversion is done. Less than one third of the system's time is thus anticipated to be required for producing plots; the remaining time is available for other TMS functions and for a workload cushion.

2.2 NOVA Memory Space Constraints

The NOVA computer in a TMS MEGATEK terminal contains 32,768 words of MOS memory, which must be used for the NOVA operating system,

the terminal program, buffers, and the MEGATEK graphics processor display list. More detail about the usage of memory and about the graphics processor is found in paragraph 4.2 and in [14], but it should be noted here that each absolute vector command requires 2 words of display list, and each character requires one word of display list. The functional requirements mentioned in Section 1 call for up to 1440 characters of scrolled ASCII data (20 lines of up to 72 characters each) and imply about 320 characters of plot annotation. About 2000 words of display list are needed to draw some plot backgrounds and to contain the miscellaneous cursors, etc. which must be displayed. The 5,000 points in the most complex plot each require 2 words of display list (one absolute vector command per point). Together, these requirements add up to about 14,000 words of display list.

Buffer space must also be provided for some portion of the information sent from the MODCOMP to the NOVA, so that the NOVA can have room to keep unprocessed commands before they are inserted into the display list. The buffer space could be as small as one word, of course, but this would require a large number of I/Os between the BIU and the NOVA, and also would tend to slow down the MODCOMP (the sender of information) to the speed of the NOVA. This slowing down occurs because the TMS design [11] processes plot transactions sequentially, so that one request must be completed before another is begun.

The buffer space situation requires, therefore, that the NOVA and its BIU be able to accept data from the network at least as fast as the MODCOMP produces it (if the NOVA is not to be the choke point). The speed with which words can be accepted is a function of the number of words which must be processed. If data conversion is done at the NOVA, approximately twice as many words (over 21,000, as opposed to less than 11,000) must be transmitted and buffered. Because the NOVA is slower than the MODCOMP, it is most desirable to provide NOVA buffer area for the entire response from the MODCOMP, so as not to cause the MODCOMP to be waiting for the NOVA to process the response.

Analysis of the size of the operating system modules required suggests that the maximum practical buffer area for network data is about 12,000 words in the NOVA plus 1,200 words in the BIU. If conversion is done in the MODCOMP, about 12,000 words must be transmitted to the NOVA (10,800 words of data points plus about 1,200 words of background information). If conversion is done in the NOVA, however, as discussed earlier, about 23,000 words must be transmitted. The entire response can be absorbed at one time only in the first case.

2.3 Workload Division Decision

On the basis of the rough analysis outlined in paragraphs 2.1 and 2.2 it was decided to perform the raster conversion function in the MODCOMP. An approach of going slightly further than just conversion, and also producing display processor instructions was actually taken in the design of the software The generation of the display processor commands described in [14] requires little more than OR-ing a set of control bits into the upper part of each raster count word, so the additional work to build the actual instructions is small.

The preliminary analysis results were sustained in tests of the completed software. When the MODCOMP is otherwise idle, the display of the maximally complex plot is completed in less than 4 seconds after the request is made.

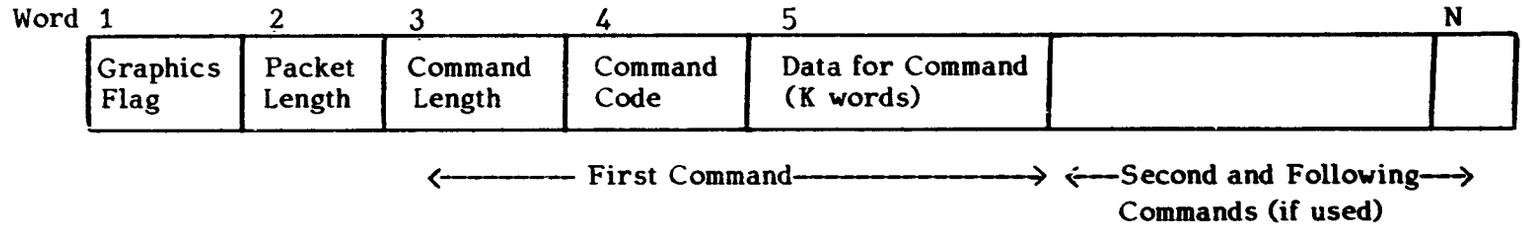
2.4 Protocol

It was recognized that communication between the MODCOMP and a graphics terminal to accomplish plotting needed to take into account two main factors. First, the communication protocol should be flexible, so that possible future modifications to the workload split could be easily accomodated. Second, the protocol should be able to operate successfully without exclusive use of the communications link between the MODCOMP and the terminal. The second requirement (which is equivalent to stating that graphics commands must be separable from other traffic) is necessary because any MODCOMP program (such as the TMS Monitor described in [10]) can write an ASCII message to the terminal without going through the graphics software. Traffic from the

graphics terminal to the MODCOMP, however, is controlled by a single program and consists only of ASCII messages.

Further analysis of the MODCOMP/MEGATEK logical interface revealed a need for two categories of graphics commands: (1) commands to add certain MEGATEK display processor instructions to the terminal's display list (see paragraphs 3.1 and 4.3), and (2) commands specifying supplementary information or state changes. In the second category are included commands such as those to define axis scaling factors and clipping windows (see paragraph 3.2), to indicate the beginning and end of construction of a display, to request reinitialization of the graphics terminal program, and to call for erasure of the screen. Graphics commands of both categories and ASCII messages may be sent to the graphics terminal in any order, but all portions of a single command or message are sent together without interruption.

To meet the requirements for flexibility and for separability, graphics commands from the MODCOMP to a terminal are placed in graphics packets, which are structured as shown in Figure 2.4-1. The packet can contain one or more graphics commands (the current TMS implementation places only one command in a packet). Each command is composed of a command length count, a command code, and zero or more words of supplementary information.



Where Graphics Flag = Hexadecimal 1111 (ASCII DC1 DC1)
Packet Length = N–2
Command Length = K+1
Command Code = Integer specifying a graphics function (see Section 3)

Figure 2.4-1 Graphics Packet Structure

This structure satisfies the flexibility requirement by allowing easy movement of computational work between the MODCOMP and the NOVA, if desired. For example, the present implementation performs raster conversion in the MODCOMP and sends MEGATEK display list absolute vector commands to the graphics terminal; a special type of graphics packet (in the first of the two categories mentioned above) is used for these absolute vector instructions. If it were desired, however, to move the raster conversion to the NOVA, only the following two steps would be required:

1. Definition of a new command code which means that the values included with the command are MODCOMP floating point user values which are to be plotted, and modification of the MODCOMP graphics software to use the new code
2. Addition of software in the NOVA to handle the new command code.

After these steps were taken, the work would be performed at the graphics terminal rather than at the MODCOMP.

This packet structure also satisfies the separability requirement by including a special graphics flag as the first word of each graphics packet; this flag allows the terminal program to distinguish graphics messages from other traffic. The graphics flag is chosen to be a pair of ASCII characters which is unlikely to occur within an arbitrary ASCII message which might be sent by any MODCOMP program to the graphics terminal. The terminal can then identify with high certainty whether the next word in its input stream is the beginning of a graphics packet (the length of which is specified by the second word of the packet) or the beginning of an ASCII message (the length of which is unknown). An ASCII message is scanned character by character until a graphics flag is found, at which point the length of the ASCII message is known. The message is then displayed in the scrolled display mentioned in paragraph 1.1.

Distinguishing between the categories of graphics messages is accomplished using the command code.

SECTION III
MODCOMP GRAPHICS SOFTWARE

3.0 INTRODUCTION

The MODCOMP graphics software is intended to provide a general graphics capability which could be used by any application program which wishes to create displays for the MEGATEK terminals. Presently, however, the only production usage of the software is in the TMS. The plot (PLO) function of the TMS calls the graphics software through several entry points to create first a plot background -- a grid, axes with annotation, a bias time, and general plot labelling. The graphics package is then called separate times to produce up to 6 curves, each of which is the trace of a single measurement from Space Shuttle instrumentation. The PLO function is expected to be by far the heaviest user of the graphics software in TMS production.

The graphics software is also used by the TMS data availability report (DAR) function to produce a bar graph of mission periods for which TMS data has been received. The DAR function can also use the graphics software to display a tabular listing of the periods for which data is available.

The following two paragraphs describe the general characteristics of the graphics software implementation (paragraph 3.1) and the concepts of scaling and clipping supported by the software (paragraph 3.2).

3.1 Graphics Software Implementation

The MODCOMP graphics software is implemented for the TMS as a series of FORTRAN subroutines. There is a separate subroutine for each of the routines described in Appendix I, and there are 6 additional utility subroutines defined for used by the plot package. Table 3.1-I lists the utility routines and their functions; more detailed documentation of the calling sequences and parameters is found in [13].

Table 3.1-1
Graphics Software Utility Subroutines

Subroutine	Purpose
DRAWR	To draw a line (either clipped or unclipped) from the present beam location to a point specified in user units. Utility routine for DRAW and DRAWC.
MEGBUF	To buffer display list commands to be sent to a graphics terminal, and to write the commands out either when the buffer becomes full or when called with a special empty-the-buffer code.
RDRAWR	To draw a line in user units relative to the current beam position. Utility routine for RDRAW and RDRAWC.
TIMCVT	To convert a time in seconds into an ASCII string specifying ±hhh:mm:ss.
WRPKT	To write a buffer of information from the MODCOMP to a graphics terminal.
WRSCA	To build and send a graphics packet to the NOVA to update the axis scale factors and clipping windows.

In the development of the MODCOMP graphics software, the original GRAPHELP entry points and parameter names (described in [10]) were retained as far as possible. As a result, some of the parameters in the calling sequences described in Appendix I are unused. Throughout the subroutines conditionally compiled code for measuring CPU time usage and for printing debug output has been included. This code is normally not a part of the production TMS, but can be included by a procedure described in [8].

The approach of implementing more complex routines (such as plotting a series of points in the routine DATAQ) in terms of less complex routines (such as plotting a single point in the routine DRAW) was generally followed, as is shown in Table 3.1-II. The only major circumstance in which this type of hierarchical programming was not used is in the heavily used routines CURVE1 and CURVE2. In the TMS, these routines are called once for each curve in a plot, and at each call may be instructed to plot 800 or more points. Initially, the routines were implemented to use ABSVEC to draw the line to each point, but measurements showed that a noticeable reduction in plotting time could be achieved if the CURVE routines were changed to generate the MEGATEK display list instructions themselves. The CURVE routines still do call ABSVEC and SYMBOQ routines for generation of the (infrequent) curve annotation labels.

Because of this hierarchical implementation, only 12 graphics command codes (see Table 3.1-II) are used in graphics packets sent to a terminal. A graphics command number is actually defined for each of the graphics functions listed in Appendix I; the number is the same as the last part of the paragraph number in which the function is described. The TMS terminal program contains stub (null) code for each of the command numbers not shown in Table 3.1-II.

Three special command codes (97, 98, and 99) are also shown in Table 3.1-II. These codes are used in handling arbitrary display processor commands (category 1 commands, as described in paragraph 2.3) and in generating the individual curves of each plot. A type 97 graphics packet is sent by the MODCOMP just before the beginning of each curve and a type 98 packet is sent after the end of each curve. These packets allow the graphics terminal to determine which display list commands make up a curve, so that the curve can be skewed

upon command (see paragraph 1.1). Type 99 graphics packets are the only category 1 packets and contain only display list instructions which are already in a form acceptable to the MEGATEK graphics processor. Whether the instructions are considered a part of a curve or part of the plot background depends on whether the type 99 packet is sent between a type 97/type 98 bracket or not.

Table 3.1-II
Implementation of MODCOMP Graphics Routines

Entry Point	Graphics Routines Called	FORTRAN Library Routines	Command Code	Comments
ABSVEC	MEGBUF			
AUTOFR		ENCODE		No NOVA support needed
AXPREC				No NOVA support needed
BIAS	WRPKT		4	
BLINKQ	WRPKT		5	Command ignored in NOVA
CURVE1	ABSVEC, SYMBOQ, WRPKT	WRITE4	97,98,99	
CURVE2	ABSVEC, SYMBOQ, WRPKT	WRITE4	97,98,99	
DATAQ	DRAW			
DRAW	DRAWR			
DRAWC	DRAWR			
DRAWR	ABSVEC			
ENTGRA	WRPKT		11	
ERASEQ	ERSALL			
ERSALL	WRPKT		13	
EXITGR	WRPKT		14	
GETSCA				No NOVA support needed
GRID	ABSVEC			
INIT	WRPKT		17	
INTENS	WRPKT		18	Command ignored by NOVA
MEGBUF		WRITE4	99	
NUMBRQ	AUTOFR, SYMBOQ			
PLOT	ABSVEC			
QCALE	WRSCA			
RDRAW	RDRAWR			
RDRAWC	RDRAWR			
RDRAWR	RELVEC			
RELVEC	MEGBUF			
SCALE	QCALE			
SCREEN	ENTGRA, ERSALL, EXITGR			
SETPDQ	INTENS		27	
SETWIN	WRSCA			
SYMBOQ	MEGBUF			
TIMCVT				No NOVA support needed
WILDCR				No NOVA support needed
WRPKT	MEGBUF	WRITE4		
WRSCA	WRPKT		21	
XAXIS	ABSVEC, AUTOFR, SYMBOQ, TIMCVT			
YAXIS	ABSVEC, AUTOFR, SYMBOQ, TIMCVT			
YOURSC	WRSCA			

3.2 Scaling and Clipping of Displays

An important function performed by the graphics software is the insulation of the application program from the need to be concerned with exact screen locations for the plotting of data points. Provided that scaling factors have been defined by calls to certain graphics routines (QCALE, SCALE, YOURSC), calls to display routines (CURVE1, CURVE2, DATAQ, DRAW, DRAWC, RDRAW, and RDRAWC) can be made using only data values in user units. In a call to one of the scaling routines the application program defines for one of the plot axes the length of the axis, the number of user units represented by the axis, and the user unit value to be associated with the end of the axis nearest the origin. A call to INIT specifies where the end of the axis is to be placed on the screen.

On the MEGATEK, vectors which extend past the edge of the physical screen are not clipped in hardware, but rather wrap around to the opposite edge of the screen. The wraparound also produces extraneous retrace lines across the entire screen face, so it is desirable to clip vectors so that they do not cause this problem. The MODCOMP graphics software defines a clipping window which is always less than or equal to the entire screen face. Calls to the scaling routines (QCALE, SCALE, and YOURSC) each implicitly set one dimension of the clipping window, according to the axis being referenced. (The axis itself is clipped to force it to fit on the screen.) The routine SETWIN can also be used to specify any clipping window that lies wholly within the screen area.

The clipping comparisons are performed only in calls to CURVE1, CURVE2, DATAQ, and DRAWC. These routines all specify point locations in user units relative to a fixed point (the plot origin established in a call to INIT), so the clipping can be done knowing only the origin location and the value of the desired point. Clipping for relative vectors (as called for in RDRAWC), on the other hand, requires that the graphics software continually keep track of the beam location in order to do clipping. Because this accounting is costly in CPU time (the cost to keep a record of beam position must be paid even in routines which do not make relative moves) and because relative moves

are not used in the TMS, the clipping of relative vectors is not included in the current graphics software implementation.

This page intentionally
left blank

SECTION IV
MEGATEK TERMINAL PROGRAM

4.0 INTRODUCTION

The MEGATEK terminal program controls the entire operation of the graphics terminal, including the handling of graphics packets from the MODCOMP, the sending of commands to the MODCOMP, and the responding to keyboard inputs from the MEGATEK. This section includes first a description of the MEGATEK hardware (paragraph 4.1), and then a view of the overall terminal program structure (paragraph 4.2). A discussion of graphics pictures is found in paragraph 4.3, and the master display list and command buffer is described in paragraph 4.4. Paragraph 4.5 outlines the intertask communications structure, while paragraph 4.6 gives a broad discussion of the function of each task.

4.1 MEGATEK Terminal Hardware

The Trend Monitoring System terminals are MEGATEK 5000 intelligent graphics terminals. The MEGATEKs are each built around a Data General NOVA/3 computer which has 32,768 16-bit words of semiconductor (volatile) memory. With the exception of a single MEGATEK configured as a development terminal with two floppy disk drives, the terminals do not have any nonvolatile storage (storage which retains its contents when power is removed). The NOVA in each terminal is fully programmable and, in the TMS, is operated under the Real Time Operating System (RTOS - see [15].

Connected to the NOVA in each MEGATEK is a standard alphanumeric keyboard with 17 function keys and a joystick control. The character codes generated by each of the keys are determined by the contents of an encoding matrix stored in a PROM in the keyboard. The PROMs and wiring in the TMS keyboards have been set up as described in [16].

Each MEGATEK also contains a high-resolution CRT (monitor), which is capable of resolving 4,096 rasters in both the X- and Y-directions. VERSATEC copiers provide the capability to make hard copies of the displays;

up to four MEGATEKs can be connected through a multiplexer to a VERSATEC copier. In the TMS, normally two terminals are connected to a multiplexer, which can be cabled to a single copier. A NOVA is attached to a MEGATEK MG-562 copier multiplexer through a MG-556 RASTERIZER interface board which plugs into the NOVA chassis.

The heart of the display generation system in a MEGATEK terminal is the MG-552 display processor. The display processor is supplied as two circuit boards -- the MG-552 Microprocessor Board and the MG-552 Vector Generator Board -- which plug into the terminal's NOVA chassis. The microprocessor in the display processor runs asynchronously to the NOVA and is treated as an I/O device by the NOVA CPU. The display processor uses Direct Memory Access (DMA) capability to fetch display processor instructions from a buffer in NOVA memory. The contents of the buffer are termed a display list; the last word in the display list is always a display processor HALT instruction. A complete discussion of the types and formats of display processor instructions is found in [14].

The display processor refreshes the display independently of the NOVA CPU by periodically retracing the display list. The refresh process may be set to operate at nominal 30 Hz, 45 Hz, or 90 Hz rates, or may be set to operate in single cycle mode (one pass is made through the display list and no refreshing is done) or in free-run mode (a new pass through the display list is begun as soon as the preceding pass is completed). The higher refresh rates generally reduce display flicker, but cause more contention for memory access cycles. In the TMS, the display processor is operated in the free-run mode.

In the TMS an additional circuit board is installed in each NOVA to provide a DMA interface between the computer and a Bus Interface Unit (BIU) connected to the communications system. The hardware portions of this interface are described in [3] and [4], and the software is outlined in [5] and [6]. A bus handler is also included in the graphics terminal program, but since the handler is documented in [5], it will not be further discussed in the following paragraphs.

4.2 Overall Structure of the Terminal Program

The TMS terminal program operates as a multitasking program under the RTOS operating system [15]. The program is composed of five major tasks -- CSMN (main task), KB (keyboard task), CI (command interpreter task), CS (curve sliding task), and JS (joystick task) -- which communicate with each other using flags and buffers stored in FORTRAN COMMON areas. Three user interrupt handlers (developed as a part of the TMS) are also employed; they are the KB70 handler to handle character-by-character interrupts from the keyboard, the communications bus handler (described in [5]), and the CLK user clock interrupt handler. Synchronization among the tasks and handlers is accomplished through the XMT and REC semaphore manipulation instructions offered as a part of RTOS Services. Some of the mechanisms of intertask communications are discussed in paragraph 4.5; more details on the manner in which tasks communicate are found in [13].

Supporting the main task programs and handlers are 29 subroutines written during TMS development and 3 subroutines taken (with minor modifications) from the MEGATEK graphics library [17] supplied by the vendor with the terminals. (The MEGATEK subroutines contain user interrupt handlers for the display processor and for the joystick.) The 3 MEGATEK subroutines are all in assembly language, while 17 of the TMS subroutines are in FORTRAN and 12 in assembly language. During execution, of course, a call of a subroutine is considered by RTOS to be a part of the activity of the task which calls the routine.

Figure 4.2-1 shows the relationships among the tasks and interrupt handlers in the terminal program. The broad flow of information among the parts of the program is shown as a set of labeled arrows connecting the symbols. The figure also indicates the RTOS task priorities assigned to each task (a smaller number implies a higher priority). Each of the tasks is discussed further in paragraph 4.6.

When the tasks and handlers are mapped together, the resulting TMS terminal program (including buffers) occupies most of the available NOVA

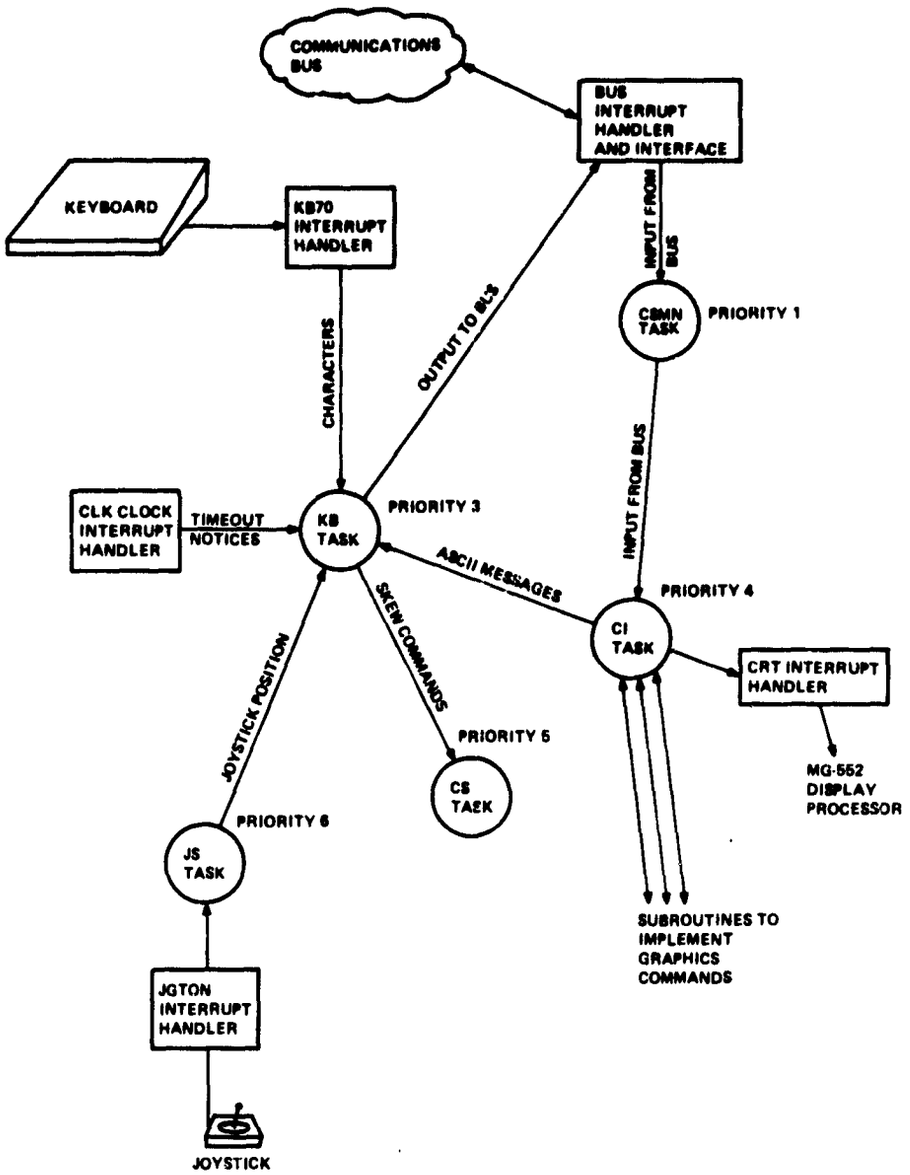


Figure 4.2-1
Tasks and Handlers of the TMS Terminal Program

memory. Figure 4.2-2 shows an approximate overall memory map for the initial production version of the terminal program. Slightly under 900 (decimal) words remain for future expansions of the code.

Length (words[1])	
288	Low-memory portions of operating system
481	Terminal program named COMMON areas (forced to appear together because they are all referenced in the BLOCK DATA subprogram)
9,600[2]	Code for terminal program
4,624	Code for FORTRAN and system library routines used by the terminal program
13,900	Blank (unlabelled) COMMON used by terminal program for master display list and command buffer
3,000[2]	High-memory portions of operating system
31,900[2]	TOTAL

NOTES: [1]All numbers are in decimal base.
[2]Approximate figure

Figure 4.2-2 Memory Map for TMS Terminal Program

A BLOCK DATA subprogram is used to initialize values in named common areas. Unlabelled common is used for the large buffer area because of the way the RTOS loader works. The loader requires a certain memory area not occupied by code or named COMMON in which the loader can be placed. The only portion of program memory in which the loader can lie is blank COMMON, since that area cannot be initialized by a BLOCK DATA routine.

4.3 Graphics Pictures

A convenient logical way to subdivide the graphics terminal display list is into units termed "pictures". A picture is a group of display list locations, always in sequential memory locations in the TMS, which is preceded by a special 5-word header and followed by a special 4-word trailer (see paragraph 4.3.2). The concept of the picture in the TMS program is based on the ideas described in [17], with minor modifications. Pictures are linked together with display list jump instructions (successive pictures need not occupy successive blocks of memory locations) in picture trailers, except for the last picture, which has a display processor HALT instruction in its trailer. The display processor is thus started (with an I/O operation) at the header of the first picture in the chain, and the processor runs through all the pictures in the display until it encounters the HALT. The processor then retraces the display list at the prescribed refresh rate, as discussed in paragraph 4.1.

4.3.1 Picture Allocation

The portion of the TMS terminal program dealing with picture management requires that all pictures be allocated from a single contiguous region of memory, which is introduced to the allocation routines by a call to the routine BIN5. The first two words of the region are initialized to a pair of display processor HALT instructions, so that the display processor can be started without problems. The terminal program then calls the BINIT routine to create a new picture and in the call specifies both the maximum length of the picture and its type. The maximum length can be either a positive integer or zero; if the length is zero, the picture is assumed to be open-ended and is allowed to grow toward

the end of the display list region until another picture is defined, at which time the first picture is closed.

The type of the picture can be either "miscellaneous" or "curves". A curves picture includes the display list instructions to generate (with annotation) exactly one of the curves in a display. A miscellaneous picture includes display list commands for generating such things as the axes, grids, and background textual information in a display.

The TMS picture allocation routines always link pictures together (see paragraph 4.3.2) so that all miscellaneous pictures come before all curves pictures in the picture chain, and so that within picture types the ordering is chronological. The terminal program also distinguishes between permanent pictures (picture numbers 1 through 8) and nonpermanent pictures. Permanent pictures are those which are always needed by the terminal program (such as those for cursor display or for display of the last 20 lines of ASCII text), but which may at times be invisible (they are always allocated, however). All permanent pictures are miscellaneous type pictures. Nonpermanent pictures are associated only with particular displays on the screen and are all deallocated when the screen is erased or a new display is generated. Figure 4.3.1-1 shows a summary of the terminal program pictures and their sizes.

Picture Number	Function	Controlling Task	Size in Words	Comments
1	---	CSMN	30	Not used
2	Bias time window	CS	30	
3	Joystick window	JS	30	
4	Joystick cursor	JS	30	
5	Current command line	KB	100	
6	Display annotation	KB	400	
7	---	---	---	Not used
8	Last 20 lines of ASCII output or command input (scrolled)	KB	1,500	Controlled through SCROL subroutine
9-32	Miscellaneous or curves pictures	CI	>9	Curves pictures may also be modified by the CS-called curve sliding routine CSLIDE

NOTE: Pictures 1-8 are termed "permanent" pictures (always allocated, but sometimes erased or turned off). Pictures 9-32 are termed "non-permanent" pictures and are deallocated when the screen is erased.

Figure 4.3.1-1 MEGATEK Graphics Pictures

Further information about the picture management routines (including BIN5, BINIT, and others), and about the arrays and approaches used in allocating and deallocating pictures is found in [13].

4.3.2 Picture Linkage

The header and trailer mentioned above are shown in Figure 4.3.2-1. During terminal program operation it is frequently necessary to make portions of the display temporarily invisible (as when the user depresses the CLEAR TEXT or CLEAR CURVES function keys). A single picture can be made invisible by changing the jump address in the second word of the header to be either the address of the first word in the trailer, or the contents of the fourth word of the trailer. All pictures beyond a certain picture (picture A, say) in the display list can be made invisible by placing a HALT instruction in the first or second word of picture A's trailer. Both of these methods are used in the TMS.

Header		
Word 1	JUMP (130000K[1])	Display list jump command
2	*+1 (address of word 3 of header)	
3	SPECIAL FUNCTION (170002K)	Reset display origin
4	SET TRANSLATION (160000K)	Set display origin (used in sliding curves)
5	0	
Trailer		
Word 1	SPECIAL FUNCTION (170002K)	Reset display origin
2	SPECIAL FUNCTION (170000K)	Null operation
3	JUMP (130000K)[2]	Jump to next picture in chain
4	address of next picture[2]	

NOTES: [1]The suffix K (as in Data General documentation) denotes an octal number.

[2]This word is replaced with a display list HALT (170010K) in the last picture in the picture chain.

Figure 4.3.2-1 Picture Header and Trailer

When a picture is to be erased, rather than made temporarily invisible, two approaches can be taken. First, the picture trailer can simply be moved in memory so that it immediately follows the picture header. This approach is taken when permanent pictures are erased.

Second, the entire picture can be removed from the display processor chain (by changing the display list jump instructions) and the picture number and memory space can be marked as unallocated. This approach is followed with nonpermanent pictures when an ERASEQ, ERSALL, or SCREEN MODCOMP graphics software call is processed.

4.3.3 Picture Contents

In the current version of the TMS graphics software and terminal program, outside of the picture headers and trailers described in paragraph 4.3.2, pictures contain only the following display processor instructions:

1. Absolute vector commands (2 words)
2. Hardware character commands (1 word)
3. Long relative vector commands (2 words)
4. Jump commands (2 words)
5. Special function commands (1 word)
6. Short relative vector commands (2 words)
7. Jump-to-subroutine commands (2 words)

MODCOMP application programs can generate only the first, second, and third types of instructions, but in the TMS, the third instruction type is not used. The fourth through the seventh instruction types are employed only in permanent pictures to generate various cursors. In miscellaneous pictures, the absolute vector and character commands can occur intermixed in any order. In curves pictures, however, in order to simplify the logic of curve sliding (see [13]) character commands occur only in even-length sequences; if an odd-length character string is needed, the MODCOMP graphics software pads the string with a character command for a null character. Characters are displayed as MEGATEK size 2 (36 by 54 rasters) for such functions as

axis annotation, ASCII messages, entered commands, and the joystick and bias time windows.

A detailed description of the contents of each picture is found in Appendix II.

4.3.4 Picture Management

The MEGATEK Corporation supplies a very flexible FORTRAN-callable graphics library [17] with the terminal, and initial work on the TMS terminal program used that library. As time passed, however, and it became clear that space in the NOVA memory was at a premium, the MEGATEK library was largely discarded (except for the modules CRT, JGTON, and JTXX, for managing the display processor and joystick interrupts). The picture structure and general approach of the MEGATEK library was retained in the TMS routines.

When a picture is being updated with the addition of new commands, it is generally desirable to exclude the display processor for the area being updated. If the display processor fetches words from an area which is undergoing arbitrary updating, the contents of words in the area may be unknown or may be partial display list commands, so that the display processor may execute, for example, a random jump. After such a jump, the display will probably contain random lines, and the display processor may halt with an error condition (such as encountering a return-from-display-list-subroutine without a corresponding jump-to-subroutine). Consequently, certain types of picture changes, such as the update of user-typed display annotation, are performed only after excluding the display processor from the picture by making the picture invisible.

The display processor can also be excluded from an area to be changed by stopping the display processor by an I/O instruction. During testing of the terminal program, however, it was discovered that when a halt I/O instruction is issued by the NOVA to the display processor, the display processor appears to continue to the end of the display list before halting.

The time before this is accomplished may be several hundred microseconds, so the terminal program in some cases must delay before making display list modifications to insure that the display processor will not be confused by the updates. During initial allocation of pictures in the routine BINIT, the display processor is halted to avoid undesired side effects.

An exception to the practice of stopping or excluding the display processor from a picture being updated is made when the commands in a picture are being updated in place and when it is known that no malformed display processor instructions can occur. For example, when pictures are being made invisible as described above, the changing of the header is performed without blocking out the display processor. Similarly, when, for example, a displayed digit is being replaced by another digit, the instruction to generate the new digit can be stored over the old instruction without excluding the display processor.

A consequence of the method of managing pictures is that in a multitasking environment, calls to the picture management routines must be done with care. In general, if two tasks each attempt to modify the same picture without external synchronization to prevent simultaneous updates, the picture may not be properly changed. In the TMS, therefore, as indicated in Figure 4.3.1-1, each picture is controlled by exactly one task, with one exception. When curve sliding is being performed, the CS task may modify one or more of the curves pictures, which are normally under the control of the CI task. In this case, the CS uses a special semaphore (MKEY - see paragraph 4.5) to lock CI from making changes to the buffer until CS has finished.

4.4 Master Display List and Command Buffer

Because of space constraints in NOVA memory, the display list and the buffer area for receiving graphics packets and ASCII messages from the MODCOMP both occupy the same array IBUF in blank COMMON. This array is referred to as the "master display list and command buffer", or simply as the "master buffer". The graphics packets have been designed so that the

length of any addition to the display list resulting from a graphics packet is always equal to or less than the length of the graphics packet itself. (The same condition holds for arbitrary ASCII messages from the MODCOMP.) Consequently, the master buffer is managed by placing permanent pictures at its low-order end and reading from the bus into an area which is a specified padding distance behind the last display list word. As a graphics packet is processed, the display list is extended by moving words from the packet to positions below the packet. As this is done, previously processed words from this or earlier graphics packets may be overwritten. Figure 4.4-1 illustrates the state of the buffer at some arbitrary time and indicates several of the pointers used by the program.

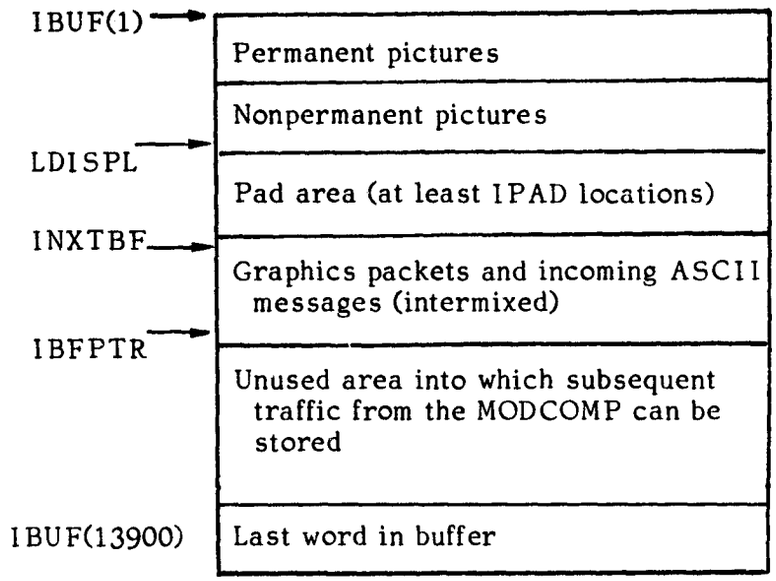


Figure 4.4-1 Master Buffer Structure

The pointers (in blank COMMON) LDISPL, INXTBF, and IBFPTR are used to indicate the index in IBUF of, respectively, the last display list word, the next buffer position to be processed by IC, and the last filled buffer position.

As commands are processed, the pad area eventually grows since most of the graphics packets are longer than the change in display list length that they cause. Eventually, of course, the unused area at the bottom of the buffer may be exhausted, and since traffic from the MODCOMP is only placed into that area, buffer compactions are sometimes needed. When the CI task detects that a buffer compaction is advisable or when the CSMN task finds that there is no room for additional MODCOMP input, the PACK subroutine of CI is called to move all unprocessed words in the buffer down so that the first word lies IPAD (currently 100) words behind LDISPL, and the other pointers are adjusted accordingly. If compaction cannot provide enough space for all the MODCOMP input, a display list overflow has occurred (generally caused by requesting a display in which too many points [more than 5000] are plotted). An advisory message is placed on the screen and the input for which there is no room is discarded.

4.5 Intertask Communication

Communication among the tasks and handlers in the terminal program is handled through shared buffers, flags, and semaphores. The use of each of these communications media is documented in detail in [13]. Buffers are used only for passing data among the tasks and handlers, rather than for influencing control, so they will not be further discussed here. The uses and identification of semaphores in the terminal program are given in paragraph 4.5.1, while a list of flags (together with the usage of the flags) is included in paragraph 4.5.2.

4.5.1 Semaphores

The RTOS operating system provides a semaphore mechanism which is useful for synchronizing the different tasks and interrupt handlers in the terminal program. A semaphore S is a designated word in a COMMON area (accessible to each of two tasks to be synchronized). Ordinarily, S controls access to a critical resource or process which must not be used by more than one task at a time. An example of such a process in the TMS is compaction

or other length changes to the display list buffer, since both the CSMN task and the C1 task use the same set of buffer pointers (see Figure 4.4-1).

In order for S to be used as a semaphore, it is first initialized to a nonzero value. A task seizes the semaphore (or performs a P operation, as it is termed by some authors) by calling the RTOS routine REC with S as a parameter. If S has a nonzero value, RTOS performs an indivisible operation which returns the value of S to the task, sets S to zero, and allows the task to continue. If S has a zero value, the task's execution is blocked until another task releases the semaphore as described below.

A task T1 releases a semaphore at the end of using the critical resource or process by calling the RTOS routine XMT with S and a nonzero value X as parameters (this corresponds to a V operation, as described by some authors). The semaphore S is then set by RTOS to contain the value X. If other tasks are awaiting the release of the semaphore, RTOS performs an indivisible operation to select one of those tasks, T2, to return the value of S to T2, to set S to zero, and to mark T2 as able to continue. If no tasks are waiting for S, T1 simply continues. Whenever a task is blocked or unblocked in this fashion, the RTOS scheduler is invoked so that the highest priority task is selected to run.

Semaphores are used in the TMS terminal program not only for protecting critical resources and processes, but also for scheduling. RTOS does not provide a capability for clock-driven time slicing, and a lower priority task, once activated, will run until it calls an operating system routine to do an I/O or until certain system events occur. These events include the occurrence of certain types of interrupts and the blocking of the task on a semaphore.

Similarly, a higher priority task T1 must block itself to allow a lower priority task to execute. T1 accomplishes this by storing a zero directly into a semaphore S, and then calling REC with S as a parameter. T1 thereby guarantees that it will block itself and allow a lower priority task T2 to execute. When T2 has completed its work, it calls XMT to release S. The

RTOS scheduler is invoked at that point and returns control to the highest priority ready task, which is now T1. Semaphores are used in this manner by, for example, the CI task (through its ERASR subroutine) to block CI until CS and JS (lower priority tasks) have accomplished certain blanking functions.

Figure 4.5.1-1 gives a list of the semaphores used in the TMS terminal program, together with a brief description of the use of each semaphore.

4.5.2 Flags

A number of memory locations in COMMON areas are used by the terminal program as flags to communicate among the tasks and handlers. Figure 4.5.2-1 summarizes the flags, their users, and their explanations. In general, the flags are set by one task (Task A, say) to request another task (Task B) to perform some operation. When Task A needs to delay until Task B has completed the operation, the usual procedure is to set the request flag, and then for Task A to block itself using a semaphore S. When Task B has completed the request, it resets the flag and releases the semaphore to reenable Task A.

4.6 Functional Descriptions of Terminal Program Tasks

The following paragraphs give a brief verbal description of how the five tasks of the terminal program operate and cooperate with each other. Further detail on these tasks, their subroutines, and the terminal program bus, keyboard, and clock handlers is found in [13]. While reading the following paragraphs, the reader may find it helpful to refer back to Figure 4.2-1 of this paper.

4.6.1 CSMN Task

The CSMN task is the main task of the terminal program. When the terminal program is initiated, the CSMN task receives control and creates the permanent pictures. It then initializes pictures 5, 6, and 8 before installing the keyboard interrupt handler and starting the other four tasks.

Semaphore	Common Block	Task Usage					Handler Usage			Usage in Blk. Data Routine	Explanation
		CSMN	KB	CI	CS	JS	KB70	CLK	BUS	BLDTA	
CICS	ERASE			P*	V					W	Used by ERASR (called by CI) to block CI until CS has made bias time window invisible.
CIJS	ERASE			P*		V				W	Used by ERASR (called by CI) to block CI until JS has made joystick window invisible.
CIKB	ERASE		V	P*						W	Used by ERASR (called by CI) to block CI until KB has erased curve annotation.
CIKSM	SEMA		V	P						W	Used by CI to block itself until KB has scrolled an ASCII message onto the screen.
CISEM	SEMA	V	V	P						W	Used by CI to block itself until the next event; CI awakened by CSMN when new input arrives from the bus and by KB when curves are to be made visible or invisible.
CSSEM	SEMA		V	V*	P					W	Used by CS to block itself until the next event; CS awakened by BIAST (called by CI) when bias time is set and by KB when curve sliding or making curves visible or invisible is requested.
ICMSEM	COMPAC	P/W	R	V*						W	Used by CSMN to block itself for a display list compaction or erasure request; CSMN awakened by PACK (called by CI) when erasure or compaction is completed. Semaphore also read by KB as starting point of dump generated by DUMP request key (not active in production software).
JSRDY	JSPOS		P			V				W	Used by KB to block itself until JS makes joystick coordinates available.
KBSEM	SEMA	P		V			V	V		W	Used by KB to block itself until the next event; KB awakened by CI to scroll an ASCII message onto the screen, by KB70 when a character is typed, and by CLK when the TALLOW flag has reached zero.
MKEY	SEMA	P/V		R/P*/V*	P/V					W	Used by various tasks to single thread accesses which change the display list/command buffer (IBUF array of blank common) in a way that affects other tasks.

NOTES:
1. Key to Task Usage abbreviations:
   P = semaphore is seized (using a REC system call)
   V = semaphore is released (using an XMT system call)
   W = semaphore value is changed (not using REC or XMT)
   R = semaphore value is examined (not using REC or XMT)
2. An asterisk beside a usage code means that a subroutine called by the task, rather than the task itself, uses the semaphore.

Figure 4.5.1-1
Semaphores Used in the TMS Terminal Program

Flag	Common Block	Task Usage					Handler Usage			Usage in Blk. Data Routine BLDTA	Explanation
		CSMN	KB	CI	CS	JS	KB70	CLK	BUS		
FLCC	FLAGS		W	R	R	R				W	Set to 0 if curve pictures are to be visible and 10 (octal) if they are to be invisible.
FLCHAR	FLAGS		R/W				W			W	Set by KB70 to (translated) code of last keyboard character; read and reset by KB.
FLCPY	FLAGS		W			R/W				W	Set to 1 by KB to request JS to copy joystick coordinates into JSX and JSY of common block JSPOS; read and reset by JS.
FLERM	FLAGS	W/W*		R/W/W*	W*	W*				W	Set to error message number for KB to display; read and reset by CI.
FLMSG	FLAGS		R/W	W						W	Set to 1 (nonblinking) or 2 (blinking) if ASCII message is present in ASCMSG array of common block BUFFRS; read and reset by KB.
FLSKEW	FLAGS		W		R/W					W	Set to +1 (skew right) or −1 (skew left) to request curve sliding; read and reset by CS.
FLTC	FLAGS		W			R				W	Set to 1 by KB to request JS to display a text cursor with its graphics cursor.
FLTD	FLAGS	W	R	W		R				W	Set to 1 by CI to disable curve sliding, joystick window while display is being built; reset by CSMN (after certain error conditions) and CI.
IALLOW	TCK		R/W					R/W			Set by KB to 3 to count a time delay between carriage returns; read and end-off shifted by CLK.
ICMREQ	COMPAC	W		R/W*						W	Set by CSMN to 1 to request display list compaction or to 2 to request forced display list erase by CI (after certain error conditions); reset by PACK (called by CI).
ICSKEW Array	CURVES			W	R*					W	Ith location set to 1 if ith curve is slidable; read by a routine (CSLIDE) called by CS.
IERCS	ERASE			W*	R/W					W	Set to 1 by ERASR (called by CI) to request CS to make bias window invisible; read and reset by CS.
IERJS	ERASE			W*		R/W				W	Set to 1 by ERASR (called by CI) to request JS to make joystick window invisible; read and reset by JS.
IERKB	ERASE		R/W	W*						W	Set to 1 by ERASR (called by CI) to request KB to erase curve annotation; read and reset by KB.
JSENBL	JSPOS			W*		R					Set to 1 by UPSCA (called by CI) to note to JS that scaling values have been defined which permit joystick window display.
NEWCMD	COMPAC	R/W	W								Set to 1 by KB whenever a carriage return is handled from the keyboard; reset and read by CSMN after certain error conditions in order to delay until next command is entered.
NOBIAS	BIAS			W*	R					W	Set to 1 by ERASR and SETPDQ (called by CI) to indicate no bias time has been defined, and set to 1 by BIAST (called by CI) when bias time is defined.

NOTE: An asterisk beside an R (for Read access) or W (for Write access means that the flag is accessed by a subroutine called by the main task program rather than by the task code itself.

Figure 4.5.2–1
Flags Used in Intertask Communication

After the initialization phase, CSMN enters an infinite loop in which it issues a wait-mode read to the bus (the first call to the bus interface routines inserts the bus interrupt handler into the system's table). When a response is returned to the read, CSMN is reactivated and checks whether there is room for the input in the master display list and command buffer. If there is room, the input is inserted, the CI task is signalled by releasing the CISEM semaphore, and the loop is repeated. If there is not room, CSMN requests that CI compact the buffer and waits until that work is done. If there is then room, CSMN inserts the input and repeats its loop, but if there is still not room CSMN generates an error message and discards the input. Any additional input received before the next command is sent to the MODCOMP is discarded. When the next command is sent, the screen is erased and the normal loop is repeated.

4.6.2 KB Task

The KB task is the second highest priority task. After initialization, KB enters a loop to check for work to be done; if no work is found, the task blocks itself on the semaphore KBSEM until another task reawakens KB by releasing KB. KB can be awakened by the keyboard interrupt handler when a key is struck by the terminal user, or by the CI task when an ASCII message is to be placed in the scrolled display of ASCII lines.

KB handles most of the key responses internally, since they relate most frequently to editing commands for the command line or editing commands for display annotation. KB does communicate with the JS task, however, when the terminal user annotates the screen, since the user employs the joystick to position the cursor to the desired point for the annotation.

Certain keyins are passed by KB to other tasks or handlers. In the case of requests to skew curves, KB uses the CSSEM semaphore to signal the CS task of the request and blocks itself until the curve sliding can be performed. When the terminal user enters a command, followed by a carriage return, KB passes the ASCII characters of the command to the bus interface to be sent to the MODCOMP. The KB task then adds the command line

to the scrolled display of the last 20 lines of ASCII text and clears the command line portion of the display.

4.6.3 CI Task

The CI task is the third-highest-priority task in the terminal program and has the responsibility of processing all graphics packets and ASCII messages received from the MODCOMP. After initialization, CI enters an infinite loop in which it first searches for work to be done. If unprocessed input is found, CI determines whether the input is a graphics command or an ASCII message. If the input is an ASCII message, it is passed to KB for insertion into the scrolled ASCII text picture, but if the input is a graphics packet, one or more subroutines are called by CI to process the graphics function. CI continues checking for unprocessed input until no more work is found, at which time it blocks itself on the semaphore CISEM until new input is received and placed in the master display list and command buffer by the CSMN task.

4.6.4 CS Task

The CS task is the fourth-highest-priority task in the terminal program and has as its responsibility the management of the bias time window and the sliding of curves upon request by the terminal user. CS begins by initializing the bias time window (picture 2) and then enters an infinite loop. In the loop CS first checks whether the bias time window should be visible and sets the window accordingly, and then checks whether there is an outstanding request to slide the curves. If a sliding request is found, CS calls its subroutine CSLIDE to perform the actual display list manipulation. When CS finds no work to do, it blocks itself on the CSSEM semaphore until it is awakened by another task.

4.6.5 JS Task

The JS Task is the lowest-priority task and executes continually when there is no higher priority work in progress. When the JS task is started,

it initializes the pictures for the joystick window and the joystick cursor (pictures 3 and 4), and then enters an infinite loop. In the loop, JS first checks whether the joystick window should be visible, and sets the window accordingly. If the window is to be visible, JS obtains the joystick coordinates from the MEGATEK JGTON module and updates the window. If the coordinates are to be returned to KB (as indicated by the setting of a flag), JS provides the coordinates and awakens KB by releasing a semaphore. As the last test in the loop, JS checks to see whether a text cursor should be displayed along with the graphics cursor, and if so, modifies the cursor picture accordingly.

The actual tracking of the joystick location and updating of the cursor position is handled by the joystick interrupt handler included in the MEGATEK JGTON module.

APPENDIX I
TMS MODCOMP APPLICATION PROGRAM GRAPHICS INTERFACE

SECTION I.1
INTRODUCTION

At the beginning of the Trend Monitoring System (TMS) project, an IMLAC graphics terminal was available for study. This terminal was supported by the GRAPHELP FORTRAN subroutines, which were written at the Harry Diamond Laboratories and supplied through IMLAC. As the TMS work continued, a subset of the GRAPHELP routines was identified for use in TMS. This document describes the calling sequence and error checks in detail for each of those routines, plus three added routines (BIAS, CURVE1, and CURVE2). In general, GRAPHELP compatibility has been retained, with three exceptions: 1) the definition of AUTOFR has been changed slightly; 2) the formal parameter IPEN has been replaced with the formal parameter LINETP in every case; and 3) two additional parameters have been added to XAXIS and YAXIS calls, and the effect of the calls has been slightly changed.

The routines are discussed in alphabetical order in Section I.2. Table I-I, however, groups the routines into sets according to their use.

TABLE I-I. GROUPING OF GRAPHICS CALLS BY USE

Scaling

GETSCA
QCALE
SCALE
YOURSC

Graphics Control

BLINKQ
ENTGRA
ERASEQ
ERSALL
EXITGR
INIT
INTENS
SCREEN
SETPDQ

Drawing

ABSVEC
CURVE1
CURVE2
DATAQ
DRAW
DRAWC
NUMBRQ
PLOT
RDRAW
RDRAWC
RELVEC

Axes and Grids

AXPREC
GRID
XAXIS
YAXIS

Annotation

SYMBOQ

Miscellaneous

AUTOFR
BIAS
WILDCR

SECTION I.2
DESCRIPTION OF ROUTINES

I.2.0 INTRODUCTION

Each of the paragraphs from I.2.1 through I.2.33 describes one of the graphics entry points which can be called by a MODCOMP application program. The descriptions outline the purpose, call sequence, parameter definitions, and side effects (where appropriate) for each routine. Paragraph I.2.0 gives information about the method of identifying the location of points on the graphics display screen, about error checks and defaults in the graphics software, and about conventions used in the routine descriptions.

I.2.0.1 Reference to Locations on the Screen

There are seven different ways which can be used to specify the location of a point on the display terminal screen:

1) In "inches"
   a. Absolute "inches" (measured from the lower left corner of the screen)
   b. "Inches" from the plot origin
   c. Relative "inches" (with respect to the current beam position)

2) In user units
   a. User units from the plot origin
   b. Relative user units (with respect to the current beam position)

3) In rasters
   a. Screen position (offset from the center of the display screen)
   b. Raster position (offset from the lower left corner of the display screen)

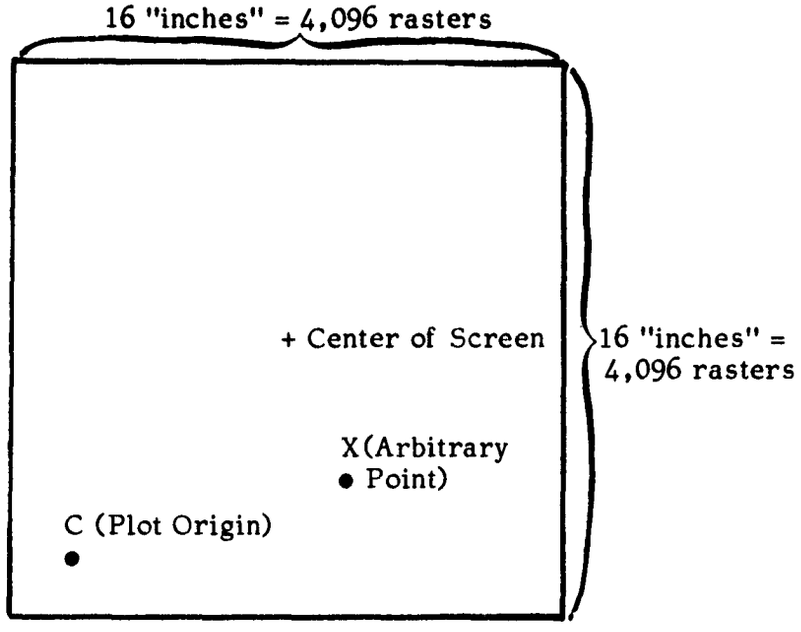
The fundamental logical unit of measure on the screen face is "inches". The "inch" is an arbitrary measurement unit derived from assuming that the screen is INMAX "inches" wide. The actual MEGATEK display screen is about 13 inches wide and is approximately square; there are 4,096 raster lines in both the X- and Y-directions. For convenience in relating "inches" to rasters, in the TMS graphics software, INMAX has been chosen to be 16 (a convenient power of two near the actual size). One "inch" is thus equivalent to 256 raster positions. The first and third types of measurements in "inches" are then easily interpreted. The second "inch" measurement relies on the plot origin definition, which is established either by default or by a call to the graphics routine INIT (paragraph I.2.17).

User units are the terms in which an application program normally works. Scaling factors to convert between user units and "inches" are established by calls to SCALE, QCALE, or YOURSC, and the graphics software then can convert the user units to "inches" before plotting. Calls to SCALE, QCALE, and YOURSC also relate user units to actual points on the screen by specifying user unit values (VLOX, VLOY) which are to be associated with the coordinates of the plot origin ((XORG,YORG) in "inches") established in an INIT call.

Positions in rasters rely on the relationship mentioned above, that one "inch" is equivalent to 256 rasters. Certain of the MEGATEK display list instructions (see [13]) deal with raster counts which are relative to the center of the screen (screen position). In other contexts, however, it is easier and less confusing to consider raster counts as relative to the lower left corner of the screen (raster position).

These methods of specifying the location of a point on the screen are used in the following routine descriptions, in the body of this report, and in the detailed documentation in [12]. Figure I.2.0.1-1 illustrates the methods of specifying a location on the display screen. In certain portions of the following descriptions, distances and sizes are given in millimeters (mm).

These distances are to be interpreted as actual millimeters; the mm measure is used to avoid possible confusion between actual inches and the arbitrarily defined "inches".



Given: Plot Origin C has user units coordinates (VLOX, VLOY) and "inches" coordinates (XORG,YORG)

Scale factors (in user units per "inch") are SFX and SFY for the X- and Y-axis, respectively.

There are 256 rasters per "inch"

Then: The following coordinates of an arbitrary point X are equivalent:

= (X,Y) in user units
= (X-VLOX)/SFX, (Y-VLOY)/SFY)), or (xi,yi), in "inches"
= (xi+XORG, yi+YORG), or (xa,ya), in absolute "inches"
= (xa*256, ya*256), or (xr,yr), in raster position
= (xr-2048, yr-2048) in screen position

Figure I.2.0.1-1 Methods of Specifying Screen Location

I.2.0.2 Error Checks

The GRAPHELP routine structure provides no mechanism for the subroutines to return error indicators to the calling routine. As a result, when errors are detected in these routines, default values for erroneous parameters are assumed where possible, and a message is written to an error log file on the MODCOMP. This file can be dumped periodically by a separate program. If an error is irrecoverable, a request is made that the system abort the calling program.

In order to enhance execution speed, not all possible error checks have been implemented in the plotting routines. In the routine descriptions, conditions which are not checked but which could lead to bad plots are marked. None of the unchecked error conditions will abort the plot package.

I.2.0.3 Conventions

Several conventions have been followed in the following descriptions of the routines. Among these are the standard FORTRAN variable naming conventions (variable names beginning with the letters A-H and O-Z represent floating point numbers and variable names beginning with I-N represent fixed point 16-bit integers), and the convention that all floating point numbers are single precision (2-word MODCOMP format) unless otherwise stated.

The graphics routines may affect the current beam position on the graphics terminal. If the beam position is updated by a routine, the change is specified under a "Side Effects" paragraph in the routine description.

I.2.0.4 Defaults

Several of the graphics package routines, such as BIAS, XAXIS, YAXIS, etc., result in the display of characters on the screen. In the calls as they are defined below, there is no mechanism to select the size of the characters. The characters used by the plot package are size 1 (about 0.1

inch wide and about 6 lines per inch). A character of size 0, as generated by the MEGATEK, resides within an imaginary box which is 8 rasters by 12 rasters. The space occupied by a character (including the white space between characters and between lines) is, of course larger and is 12 by 18 rasters for size 0 characters. These sizes are linearly scaled upward for other character sizes, so that the raster dimensions of characters of size N can be computed by multiplying these raster counts by N+1.

I.2.1 ABSVEC (XAB,YAB,LINETP)

Purpose: To draw a line from the present beam position to absolute location (XAB,YAB).

Parameters:

XAB and YAB are floating point "inches" measured from the lower left-hand corner of the screen.

LINETP must be an integer.
LINETP = 0 results in an invisible line
= 1 causes a solid line
= any other value results in a dashed line

Error Conditions:

XAB must be $\leq$ INMAX
YAB must be $\leq$ INMAX

XAB and YAB are forced to be less than INMAX by calculating the coordinates XAB1 = MOD(XAB,INMAX) and YAB1 = MOD(YAB,INMAX). The resulting absolute vector is drawn from the present beam position to the point (XAB1,YAB1), which may provide unexpected results if XAB and/or YAB are $\geq$ INMAX.

Side Effects: The beam position is updated to (XAB1,YAB1). The setting of LINETP to 1 or another nonzero value sets the line type to be solid or dashed, respectively, for calls to any relative vector routine

(RELVEC, RDRAW, and RDRAWC) before the next call to any other routine which uses the parameter LINETP.

1.2.2 AUTOFR (VAL,IFORM,NFORM,IPREC)

Purpose: To convert a given floating point number into FORTRAN F-type display format.

NOTE: This function is different from the GRAPHELP implementation, which returns a format to be used in conversion, rather than the converted number.

Parameters:

VAL is the floating point number to be converted.

IFORM is the array which will receive the display code representation; the first nonblank ASCII character is placed in the left half of the first word of IFORM (left justification). This array should be long enough to hold the returned value (maximum of 6 words).

NFORM is an integer variable in which the width of the display representation is returned.

IPREC is the number of figures to be displayed to the right of the decimal point ($0 \leq IPREC \leq 6$).

Algorithm: The FORTRAN ENCODE capability is used to convert the number into ASCII display code. The resulting field is up to 12 characters wide and is packed into the output array at 2 characters per word. If the number cannot be fitted into twelve characters with the given precision, or if the precision is invalid, the output array is filled with asterisks and NFORM is set to 12.

I.2.3 AXPREC (IPREC)

Purpose: To specify the precision of axis labels that are output when the routines XAXIS and YAXIS are called.

Parameter:

IPREC = number of digits to the right of the decimal point in axis labels.

Error Conditions:

IPREC must be ≥ 0. If it is not, it is set to the default value.

Default Value:

If this routine is not called to set axis label precision, the precision is zero.

I.2.4 BIAS (XIN,YIN,SEC)

Purpose: To specify the location and magnitude of the bias time which is associated with curves which are skewable (see the descriptions of CURVE1 and CURVE2 in paragraphs I.2.6 and I.2.7). Calling this routine causes the bias time to be displayed on the graphics screen at the specified location.

Parameters:

XIN and YIN specify the lower left corner of the first character of the bias as it is displayed on the graphics screen ($\pm$hhh:mm:ss). XIN and YIN are given in floating point "inches".

SEC specifies the seconds of bias time (SEC may be positive, negative, or zero). The bias time will be initially displayed as built from this figure and will be updated as the terminal operator skews displayed curves using function keys.

Error Conditions:

XIN and YIN should be chosen so that the characters of the displayed bias time will not extend past the edge of the screen (a space about 5 mm by 25 mm is required), or wraparound will result. No checks are made in this routine for wraparound.

Default Value:

If this routine is not called to set the bias value, no bias value will be displayed on the screen and no updating of a time bias will be performed if the user skews the displayed curves.

Side Effect: The current beam position is updated to the end of the bias time character string. One should note that if XIN+XORG > INMAX or if YIN+YORG > INMAX, wraparound occurs (before the display of the character string) and the new beam position is (MOD (XIN+XORG,INMAX), MOD(YIN+YORG,INMAX)).

1.2.5 BLINKQ

Purpose: To toggle the blink function on and off. All graphics vectors and characters written after an odd-numbered call to BLINKQ but before the next call to BLINKQ will blink. Blink is initially off.

Programming Note:

Blinking does not take effect until the next absolute vector is created. Presently, BLINKQ does not take effect for calls to NUMBRQ, RELVEC, RDRAW, RDRAWC, and SYMBOQ unless a call to ABSVEC, DATAQ, DRAW, DRAWC, GRID, PLOT, XAXIS, or YAXIS intervenes.

1.2.6 CURVE1(YARRAY,NPOINT,XINCR,XINIT,LINETP,LABEL,ICRVTP)

Purpose: To scale, plot, and connect a list of floating point user values, with each point's vertical position being determined by the value of an element in the list, and that point's horizontal position being

determined by a constant offset from its predecessor. The plot is clipped according to a window set explicitly in a call to SETWIN or implicitly in a call to QCALE or SCALE.

Parameters:

YARRAY is an array of 32-bit floating point user values, with at least NPOINT entries. If the array contains any value in the first NPOINT positions which is not a valid floating point number, results are unpredictable. (See CURVE2 for a routine which examines data for missing point flags.)

NPOINT is the number of points to be plotted. If NPOINT ≤ 0, the subroutine takes no action. If NPOINT exceeds the dimension of YARRAY, results are unpredictable. NPOINT must be ≤ 2000; if this condition is violated, only the first 2000 points are plotted.

XINCR is the floating point displacement (in user units) in the horizontal (X) direction between successive points. Ordinarily, XINCR is positive, though negative values are allowed; care should be taken, however, to be sure that XINIT is adjusted so that all points lie in the plot window, or they will be clipped.

XINIT is the floating point value (in user units) of the X-coordinate of the first data point.

LINETP is an integer denoting the type of line with which the data points are to be connected. Permissible values for LINETP are shown in paragraph I.2.1.

LABEL is a 2-byte integer variable containing in its right-hand byte the ASCII character code for the character to be used in annotating this curve. This character is displayed periodically along the curve (approximately every two "inches"). The plot package attempts to position the annotation for different curves so that the annotations do not overlap.

ICRVTP is an integer describing whether or not this curve can be skewed by use of the SKEW function keys. Generally, only curves representing past or predicted data will have this attribute in TMS.

ICRVTP = 0 means that the curve is not skewable
= any other value means that the curve is skewable

Prerequisites: The current scaling and origin information is required by CURVE1, so INIT should have been called previously to specify the origin, and each dimension (X and Y) should have had its scaling information set by a call to YOURSC, SCALE, or QCALE. If the default clipping window set in QCALE or SCALE is not desired, SETWIN should also have been called to specify a window for clipping. If these routines have not been called prior to a call to CURVE1, the defaults stated in the routine descriptions will apply.

Programming Note:

If any part of the clipping window (set in SETWIN or by default in QCALE or SCALE) does not lie on the screen, clipping is adjusted so that any points which lie off the visible screen are clipped. Subsection I.2.28 contains more details about the adjustment of the clipping window in such a case.

Side Effect: The beam position after this routine is called is updated to be the plotted location (possibly clipped) of the NPOINTth point plotted.

I.2.7 CURVE2 (YARRAY,NPOINT,XINCR,XINIT,LINETP,LABEL, ICRVTP)

Purpose: To scale, plot and connect a list of floating point values with a constant horizontal displacement between successors. The curve is clipped according to a window set explicitly in SETWIN or implicitly in QCALE or SCALE. Also, whenever a value encountered in the input array YARRAY is the absent-point flag (hexadecimal 7FC0 0000 on the MODCOMP) the point is not plotted or connected to the valid points. Instead, the horizontal position is incremented one step for each data value examined, and the plot is continued with the next valid value found (if any).

Additional Information:

Parameters, prerequisites, and side effects are identical to those stated for CURVE1.

I.2.8 DATAQ(XARRAY,YARRAY,NPOINT,INC,LINETP)

Purpose: To scale, plot, and connect a series of points, each of which is specified by its floating point user value. Clipping is not performed in this subroutine.

Parameters:

XARRAY is an array of floating point X-values, with at least NPOINT entries. All entries in the first NPOINT positions should be valid floating point numbers, or results are unpredictable.

YARRAY is an array of floating point Y-values, with at least NPOINT entries. All entries in the first NPOINT positions should be valid floating point numbers, or results are unpredictable. The ith point to be plotted is specified by (XARRAY(i), YARRAY(i)).

NPOINT is an integer value giving the number of points to be plotted. If NPOINT ≤ 0, no plotting occurs. If NPOINT exceeds the dimensions of either XARRAY or YARRAY, results are unpredictable.

INC is an integer variable specifying how points are to be chosen from XARRAY and YARRAY. If INC is i, every ith value is plotted; normally, INC is set to 1 so that each array element is plotted.

LINETP determines the type of line which is used to connect the points. Permissible values for LINETP are given in paragraph I.2.1.

Side Effect: The current beam position is updated to the position of the last point plotted.

I.2.9 DRAW(XS,YS,LINETP,ITYPE)

Purpose: To draw a line from the present beam position to the location (XS,YS), where XS and YS are user values which are scaled before plotting.

Parameters:

XS and YS are floating point user values which are scaled according to scaling information set by SCALE, QCALE, or YOURSC.

LINETP describes the type of line to be drawn. Values for this integer parameter are given in paragraph I.2.6 (description of CURVE1).

ITYPE is a parameter included for compatibility with the GRAPHELP definition of this routine; in GRAPHELP, ITYPE specifies the type of plot (linear, log-linear, log-log, etc.). Since this plot package supports only linear plots (ITYPE=4) at present, this parameter is ignored by the subroutine.

Programming Note:

This routine does not clip vectors (see DRAWC for a version of the routine which performs clipping), so it is possible to draw a vector which "wraps around" the screen. No checks are made for wraparound.

Side Effect: The beam position is updated to the location of the plotted point (XS,YS). If wraparound occurs, the new beam location reflects the wraparound.

1.2.10 DRAWC(XS,YS,LINETP,ITYPE)

Purpose: To draw a line from the present beam position to the location (XS,YS), where XS and YS are user values which are scaled before plotting. The resulting line is clipped according to the window resulting from a previous call to SETWIN (or implicitly in a call to QCALE or SCALE).

Additional Information:

This routine differs from DRAW (paragraph 1.2.9) only in that DRAWC performs clipping.

1.2.11 ENTGRA

Purpose: To enter the graphics mode of adding elements to the display being built. All graphics subroutine calls are valid only after a call to ENTGRA and before the next call to EXITGR.

Programming Note:

When ENTGRA is called, the terminal operator's capabilities to interrogate the screen (to obtain values corresponding to a given point on the plot) and to skew curves are suspended. These capabilities are restored when EXITGR is called.

I.2.12 ERASEQ

Purpose: To erase the screen when in graphics mode. This call blanks the screen and erases previous graphical elements; text elements displayed on the screen by ordinary FORTRAN writes are retained in the terminal, though they are not displayed on the screen unless called back by the user's depressing the SHOW TEXT terminal function key. As described in the TMS Level C Requirements Document, only the last 20 lines of text (total of both visible and invisible lines) are retained at any time.

Blanking when the calling program is not in graphics mode is accomplished through a call to SCREEN (paragraph I.2.26).

I.2.13 ERSALL

Purpose: To erase the screen when the calling program is in graphics mode. The effect of this routine in the TMS graphics package is the same as the effect of ERASEQ (paragraph I.2.12).

I.2.14 EXITGR

Purpose: To exit the graphics mode of adding elements to the picture being built. This routine should be called before termination of the program building the display.

Programming Note:

When EXITGR is called, the terminal operator's capabilities to interrogate the screen and to skew curves are restored. These capabilities are suspended when ENTGRA (paragraph I.2.11) is called.

I.2.15 GETSCA(SF,VLO,IWHO,ITIME)

Purpose: To return the current values of the scale factor and least plottable value for either the X-axis or the Y-axis. SF and VLO are set by calls to QCALE, SCALE, or YOURSC.

Parameters:

SF is the scaling factor (in user units/"inch") by which values in user units are divided to convert them to "inches" for plotting. SF is set by this routine, and applies to the axis specified by IWHO.

VLO is the least plottable value in user units for the axis specified by IWHO. This value is the starting point of the axis in the displayed plot.

IWHO specifies for which axis the values SF and VLO are returned.

IWHO = 0 means the X-axis
= any other value means the Y-axis

ITIME specifies whether the axis specified by IWHO represents time or not. Possible values for ITIME are given in paragraph I.2.21.

I.2.16 GRID(XGRD,YGRD,XD,YD,LINETP,IREL)

Purpose: To draw a regular grid (the size of the X intervals may differ from the size of the Y intervals) on the graphics screen. The lower left-hand corner of the grid is taken to be the plot origin established by a call to INIT.

Parameters:

XGRD and YGRD are the coordinates in "inches" of the upper right-hand corner of the grid on the screen.

XD and YD are the distances between the Y-direction and X-direction grid lines, respectively. XD and YD are expressed in "inches".

LINETP determines the type of lines used in the grid (solid, invisible, or dashed). The values for LINETP are the same as those given for ABSVEC (paragraph I.2.1).

IREL is a parameter included for GRAPHELP compatibility; in GRAPHELP IREL can be used to specify that the grid must be drawn using only relative vectors, but the value of IREL is ignored in the TMS graphics package.

Error Conditions:

The following conditions must be satisfied. If any of the conditions do not hold, no grid is produced.

$0 \leq$ XGRD $\leq$ INMAX
$0 \leq$ YGRD $\leq$ INMAX
$0 <$ XD $<$ INMAX
$0 <$ YD $<$ INMAX

Programming Note:

Ordinarily, XGRD-XORG should be a multiple of XD and YGRD-YORG should be a multiple of YD, where (XORG,YORG) is plotting origin which should have been previously established by a call to INIT. If these differences are not whole multiples of the X-increment and Y-increment, the last X-direction line will be at YORG + k x YD, where YORG + k x YD $\leq$ YGRD and YORG + (k+1) x YD > YGRD. An analogous relationship holds for Y-direction lines.

Side Effect: The beam position after a call to GRID is set to the plot origin (XORG,YORG), as defined either by default or by a call to INIT.

I.2.17 INIT(XORG,YORG)

Purpose: To set the coordinates of the absolute plot origin in "inches". These coordinates determine the lower left-hand corner of plots; this point on the screen can be associated with arbitrary user units by the routines QCALE, SCALE, and YOURSC.

Parameters:

XORG and YORG are the coordinates of the absolute plot origin in "inches". XORG and YORG are always specified as offsets from the lower left-hand corner of the screen, which has the position (0,0) for purposes of this definition.

Error Conditions:

The following conditions must hold:

$0 \leq \text{XORG} \leq \text{INMAX}$

$0 \leq \text{YORG} \leq \text{INMAX}$

The default value, as defined below, is applied if these conditions do not hold.

Programming Note:

This routine should be called before any routine which references the plot origin, since the plot origin in effect at the time of a call to a routine using the origin is the one which will be used.

Default Values:

If INIT is not called the absolute plot origin is taken to be (1.0,1.0) [in "inches"]. If either XORG or YORG is out of range, as mentioned above, that coordinate is assigned a value of 1.0.

I.2.18 INTENS(IBRITE)

Purpose: To choose the beam intensity for plotting.

Parameter:

IBRITE is an integer in the range [1,16], where higher values result in higher intensity. If IBRITE = 1, a zero intensity (blank trace) is used. If the value of IBRITE is not in the allowable range, the default value is used.

Default Value:

If INTENS is not called or if an invalid value is specified for IBRITE, an intensity of 8 is used.

Programming Note:

Setting beam intensity may not take effect immediately. See the programming note for BLINKQ (paragraph I.2.5).

I.2.19 NUMBRQ(VAL,IPREC,NSIZE)

Purpose: To convert a floating point number to display characters and then to draw the display characters on the screen at the current beam position.

Parameters:

VAL is the floating point number to be displayed.

IPREC is an integer giving the number of digits to be displayed to the right of the decimal point; if IPREC is ≤ 0, no decimal point will be displayed and no fractional digits will be displayed. The displayed value of VAL is truncated to IPREC precision.

NSIZE is an integer defining the height of the displayed characters. Permissible values for NSIZE are given in paragraph I.2.29 (description of SYMBOQ).

Programming Note:

The programmer must be careful to position the beam to a point where the displayed number will not extend beyond the edge of the

screen, or wraparound will occur. No checks are made in this routine for wraparound.

Side Effect: The current beam position after this routine is updated as described for SYMBOQ (paragraph I.2.29).

I.2.20 PLOT(XIN,YIN,LINETP)

Purpose: To draw a line from the present beam position to a location relative to the absolute plot origin set in a call to INIT.

Parameters:

XIN and YIN are floating point "inches" relative to the absolute plot origin set by INIT.

LINETP is an integer which describes the type of line (solid, invisible, or dashed) that is to be drawn. Permissible values for LINETP are given in the description of CURVE1 (paragraph I.2.6).

Error Conditions:

ABS(XIN) + XORG must be $\leq$ INMAX
ABS(YIN) + YORG must be $\leq$ INMAX
where XORG and YORG are set by INIT.

These conditions are not checked, and if they are violated, the resulting line will wrap around the screen.

Side Effect: The beam position is updated to (XIN,YIN) [these coordinates are in "inches" from the absolute plot origin]. Note that if wraparound occurs, the new beam position is (MOD(XIN,INMAX), MAX(YIN,INMAX)).

1.2.21 QCALE(AMIN,AMAX,AXLEN,SF,VLO,IWHO,ITIME)

Purpose: To set the scale factor for user units and the least plottable value from given minimum and maximum user unit values.

Parameters:

AMIN and AMAX are floating point user unit values for the minimum and maximum values to be scaled, respectively.

AXLEN is the length of the axis (identity of the axis is specified by IWHO) in "inches".

SF is the scaling factor by which user unit values are divided to convert them to "inches". This value is calculated by the subroutine.

VLO is the least plottable value. This value is set by the routine and in the TMS graphics package is always equal to AMIN.

IWHO specifies the axis for which scaling information is being given.

IWHO = 0 means the X-axis
= any other value means the Y-axis

ITIME specifies whether the user units for the specified axis deal with time or with some other unit of measure.

ITIME = 0 means the units are not time
= 1 means the units are seconds
= 2 means the units are minutes
= 3 means the units are hours
= any other value means the units are not time.

Algorithm: The scaling factor SF has dimensions of user units per "inch" and is calculated in the following straightforward way:

SF = (AMAX - AMIN) / AXLEN

Error Conditions:

The relationship $0 < AXLEN \leq INMAX$ must hold. If this condition is violated, default values are set for SF, VLO, and for the clipping window for the specified coordinate.

Default Value:

If this routine is not called (and SF, VLO, and the clipping window are not set by calls to other routines such as SCALE, YOURSC, and SETWIN), or if an error is detected in the AXLEN specification when this routine is called, SF, VLO, and ITIME are set to zero; the default value for the clipping window is specified in paragraph I.2.28 (SETWIN).

Side Effects: As indicated, the clipping window for the axis identified by the IWHO parameter in the call to QCALE is set in this routine. The minimum value (user units) of the window is set to AMIN and the maximum value to AMAX. Other changes to the clipping window can be accomplished by calling SETWIN.

The values for SF and VLO are used, together with the coordinates of the plot origin ((XIN,YIN) -- set by default or by a call to INIT), to calculate the value in user units for a particular point on the screen when the terminal operator interrogates the screen using the joystick. If SF and VLO are set more than once for a particular axis, the last values set are the ones used; if SF and VLO are never set, they are taken to have the default values stated above.

I.2.22 RDRAW (XS,YS,LT,ITYPE)

Purpose: To draw in user units relative to the current beam position, but without clipping.

Parameters:

XS and YS are relative displacements (in user units) from the current beam position. A line is drawn from the present position to the point defined by XS and YS.

LT determines whether the line that is drawn is invisible (LT=0) or is drawn with the current intensity and line type (solid or dashed). Intensity is set through the routine INTENS and the line type is determined by the line type used in the last call generating absolute vectors (any call except RELVEC, RDRAW, and RDRAWC).

ITYPE is a parameter which is included for compatibility with GRAPHELP, but which is ignored by this routine. A brief description of the GRAPHELP usage of ITYPE is given in paragraph I.2.9.

Programming Note:

This routine does not clip vectors (see RDRAWC for a version of the routine which performs clipping), so it is possible to draw a vector which wraps around the screen. No checks are made for wrap around.

Side Effect: The beam position is updated to the location of the plotted point:

(MOD((curr X loc)+XS,INMAX),MOD((curr Y loc)+YS,INMAX))

I.2.23 RDRAWC (XS,YS,LT,ITYPE)

Purpose: To draw in user units relative to the current beam position, while clipping at window boundaries. Window boundaries must have been set by previous calls to QCALE, SCALE, or SETWIN.

Additional Information:

Since clipping of relative vectors was not implemented for TMS, this routine is identical to RDRAW (paragraph I.2.22).

I.2.24 RELVEC(X,Y,LT)

Purpose: To draw a line from the present beam position to a point specified in "inches" away from the present beam position.

Parameters:

X and Y are displacements in "inches" from the present beam position.

LT specifies the visibility of the line which is drawn (as discussed in the description of RDRAW [paragraph I.2.22]).

Programming Note:

X and Y must satisfy the relations

X + old X position ≤ INMAX
Y + old Y position ≤ INMAX

or wraparound will occur. No check is made for wraparound in this routine.

Side Effect: The beam position is updated to the position of the end of the relative vector. One should note that if wraparound occurs, the new beam position will be

(MOD(X+old X pos,INMAX),MOD(Y+old Y pos,INMAX))

I.2.25 SCALE(ARRAY,NPOINT,AXLEN,INC.IWHO,ITIME)

Purpose: To set the scaling factor for converting user units to "inches". The scale factor is set based on data values in ARRAY.

Parameters:

ARRAY is a group of floating point values upon which the scaling factor should be based. The maximum and minimum values are selected from the array and are used in the same way that AMAX and AMIN are used in calls to QCALE.

NPOINT defines how many data values are to be considered from ARRAY.

AXLEN gives the length of the axis specified by IWHO; the axis length is expressed in "inches".

INC specifies which values of ARRAY will be considered; NPOINT values will be examined, one from every INCth cell of ARRAY. Ordinarily, INC is 1, so that successive cells of ARRAY are examined.

IWHO specifies for which axis the scale factor is being defined. As for QCALE,

IWHO = 0 means X-axis
= any other value means Y-axis

ITIME specifies whether the user units for the specified axis are to be interpreted as time. Possible values for ITIME are given in the discussion of QCALE (paragraph I.2.21).

Error Conditions:

NPOINT must be an integer greater than zero; if it is not, it is assumed to be zero and no action is taken by the subroutine.

INC must be an integer greater than zero; if it is not, the routine returns without taking any action.

AXLEN should be such that the axis length, when combined with the plot origin set in a call to INIT, does not extend past the edge of the screen (wraparound). No check is made for this condition, however.

The length of ARRAY must be at least NPOINT * INC elements, or results are unpredictable. No check is made for this condition, however.

Programming Note:

Except for the means of specifying the minimum and maximum data values for computing the scale factor, this routine behaves in the same way as QCALE; additional information about algorithms and default values can be found in paragraph 1.2.21.

Side Effects: As with QCALE, the clipping window limits for the axis specified by IWHO are implicitly set by a successful call to this routine; paragraph 1.2.28 (SETWIN) discusses error conditions which may arise in connection with the clipping window. Similarly, this routine affects the values used for screen interrogation, as discussed in the "Side Effects" portion of paragraph 1.2.21.

1.2.26 SCREEN

Purpose: To erase the display screen when the calling program is not in graphics mode (graphics mode is entered through a call to ENTGRA).

Programming Note:

Except for the mode of the calling program, this routine is identical to ERASEQ (paragraph 1.2.12).

1.2.27 SETPDQ

Purpose: To initialize the plotting package by resetting the plot origin, the clipping window, etc. This routine should be called whenever a reinitialization of the display package is desired; it must be called before the first plot is made after a calling program is initiated. When SETPDQ is called the screen is erased.

1.2.28 SETWIN (XMN,YMN,XMX,YMX)

Purpose: To set the clipping window used in routines CURVE1, CURVE2, DRAWC, and RDRAWC.

Parameters:

XMN and YMN are the coordinates of the lower left-hand corner of the clipping window, while XMX and YMX are the coordinates of the upper right-hand corner of the window. All four values are expressed in user units.

Error Conditions:

At the time a clipped plot is produced, each of the four values specifying the window must lie on the screen; that is

$$0 \leq XMN/SF + XORG \leq INMAX$$
$$0 \leq XMX/SF + XORG \leq INMAX$$
$$0 \leq YMN/SF + YORG \leq INMAX$$
$$0 \leq YMX/SF + YORG \leq INMAX$$

where XORG and YORG are the coordinates of the plot origin as established by default or in a call to INIT. (Note that SF and (XORG,YORG) can be set in other plot package calls after the call to SETWIN.) If any of these conditions is false at the time of a clipped plot, clipping for the axis involved (both maximum and minimum values) is at the edges of the screen.

Programming Note:

The clipping window for an axis is implicitly set in calls to QCALE and SCALE (but not in a call to YOURSC).

Default Values:

If the clipping window for an axis is not set in one of the ways mentioned above, when clipped plotting is performed lines are clipped only when they extend past the edge of the screen.

I.2.29 SYMBOQ(NCHAR,ITEXT,NSIZE)

Purpose: To display a string of ASCII characters on the graphics screen.

Parameters:

NCHAR is an integer specifying the number of characters to be displayed.

ITEXT is an integer array of ASCII characters (packed two to a word) which are to be displayed. If ITEXT does not contain at least NCHAR characters the display results are unpredictable.

NSIZE is an integer defining the size of the characters. NSIZE can range from 0 to 7, where higher numbers correspond to larger character sizes. The smallest character (size 0) is approximately 1 mm tall, while the largest character (size 7) is about 8 mm tall.

Error Conditions:

NCHAR must be ≥ 0; if it is not, it is assumed to zero, and no characters are displayed.

If NSIZE does not fall in the range 0-7, a value of 1 (corresponding to a character height of about 2 mm) is used.

Programming Notes:

The displayed character string begins at the current beam position (which is taken to be the lower left-hand corner of the first character) and continues to the right. No checks are made for wraparound (continuing past the edge of the screen) and no scrolling occurs. No ASCII control characters should appear in the character string except carriage return (CR) and line feed (LF); if other control characters do appear, the results are unpredictable.

The CR LF sequence is considered to mean "advance one line"; the left-hand margin of the succeeding line is taken to be the X-position of the beginning of the character string. Wraparound is

possible in the vertical direction if space is not allowed by the calling program for lines to appear beneath a current line when the CR LF sequence is output.

Side Effect: The beam position is updated after a call to SYMBOQ to be at the end of the character string which is displayed.

I.2.30 WILDCR (NWILD)

Purpose: To support a variety of special functions offered by the GRAPHELP package. This routine is a null routine at present in the TMS graphics package; calls to WILDCR are allowed, but perform no work.

I.2.31 XAXIS(LABEL,NCHAR,AXLEN,BIGTIC,SMTIC,LABTIC)

Purpose: To draw a labeled X-axis, with periodic tic marks, beginning at the plot origin defined in a call to INIT (or by default, if INIT has not been called).

Parameters:

LABEL is an integer array containing the ASCII characters for the axis label, packed two to a word. This label is displayed centered below the axis. Only alphabetic and numeric characters may appear in the label, together with the spacial characters +-.,#$&*()%=/?

NCHAR is an integer value giving the number of characters contained in the label. If NCHAR = 0, no label is displayed.

AXLEN specifies the length of the axis in "inches".

BIGTIC is a floating point value in "inches" giving the separation between long tic marks (extending about 4 mm from the axis) along the axis. The sign of BIGTIC has the following interpretation:

BIGTIC > 0 means the tic marks are drawn downward from the axis
= 0 means no tic marks are drawn
< 0 means the tic marks are drawn upward from the axis.

SMTIC is a floating point value in user "inches" giving the separation between short tic marks (extending about 3 mm from the axis) along the axis. The sign of SMTIC has the same interpretation as the sign of BIGTIC.

LABTIC is an integer value specifying whether each long tic mark is to be labeled with the value (in user units) which the mark represents. If LABTIC = 0, no labeling will occur; if LABTIC is any other value, long tic marks will be annotated. If LABTIC is > 0 annotations will be displayed below the axis, and if LABTIC is< 0 annotations will be shown above the axis. The annotation value is calculated as

VLO + SF x ((X-coord of tic mark) - XORG),

where XORG is the X-coordinate of the plot origin (set in a call to INIT or by default), VLO is the value in user units of the X-coordinate of the plot origin (VLO is set in a call to QCALE, SCALE or YOURSC), and SF is the scale factor in user units per "inch" (also set in a call to QCALE, SCALE, or YOURSC).

If the user values for an axis are time in minutes (as specified with the ITIME parameter in a call to QCALE, SCALE, or YOURSC), annotations of tic marks are in the form ±hh:mm. If the user values are time in seconds, annotation of tic marks are of the form ±hh:mm:ss.

Error Conditions:

If the array LABEL contains control characters in any of its first NCHAR characters, the results are unpredictable.

NCHAR must be ≥ 0. If it is not, it is assumed to be zero and no label is displayed for the axis.

AXLEN must be such that XIN + AXLEN $\leq$ INMAX, or wraparound will occur. No check is made for wraparound in this routine, however,

Programming Notes:

Small tic marks are placed on the axis every SMTIC "inches". Larger tic marks are drawn every BIGTIC "inches".

The precision of the value displayed beneath tic marks is controlled by a call to AXPREC (paragraph 1.2.3). If this routine has not been called, the default precision stated in paragraph 1.2.3 applies.

The calling program must insure that the plot orogin is set far enough from the edge of the screen that the axis annotation will fit between the axis and the edge of the screen; about 0.6 "inch" should be allowed.

Side Effect: The beam position is moved to the plot origin (XORG, YORG), which has been defined either by default or by a previous call to INIT.

1.2.32 YAXIS(LABEL,NCHAR,AXLEN,BIGTIC,SMTIC,LABTIC)

Purpose: To draw a labeled Y-axis, with periodic tic marks, beginning at the plot origin defined in a call to INIT (or by default, if INIT has not been called).

Parameters:

LABEL is an integer array containing the ASCII characters for the axis label, packed two to a word. This label is displayed centered to the left of the axis; the characters in the label are in

normal orientation but appear in a veritical column. Only alphabetic and numeric characters may appear in the label, together with the special characters +-.,#$&()%=/? .

NCHAR is an integer value giving the number of characters contained in the label. If NCHAR = 0, no label is displayed.

AXLEN specifies the length of the axis in "inches".

BIGTIC is a floating point value in "inches" giving the separation between long tic marks axis. The sign of BIGTIC has the following interpretation:

BIGTIC > 0 means the tic marks are drawn to the left from the axis
= 0 means no tic marks are drawn
< 0 means the tic marks are drawn to the right from the axis

SMTIC is a floating point value in user "inches" giving the separation between short tic marks (extending about 3 mm from the axis) along the axis. The sign of SMTIC has the same interpretation as the sign of BIGTIC.

LABTIC is an integer value specifying whether each long tic mark is to be labeled with the value (in user units) which the mark represents. If LABTIC = 0, no labeling will occur; if LABTIC is any other value, long tic marks will be annotated. If LABTIC is < 0, annotations will be made on the right side of the axis and if LABTIC is > 0 annotations will be made on the left side of the axis. The annotation value is calculated as

VLO + SF x ((Y-coord of tic mark) - YORG),

where YORG is the Y-coordinate of the plot origin (set in a call to INIT or by default), VLO is the value in user units of the Y-coordinate of the plot origin (VLO is set in a call to QCALE, SCALE, or YOURSC), and SF is the scale factor in user units per "inch" (also set in a call to QCALE, SCALE, or YOURSC).

If the user values for an axis are time in minutes (as specified with the ITIME parameter in a call to QCALE, SCALE, or YOURSC), annotations of tic marks are in the form ±hh:mm. If the user values are time in seconds, annotations of the tic marks are of the form ±hhh:mm:ss.

Error Conditions:

If the array LABEL contains control characters in any of its first NCHAR characters, the results are unpredictable.

NCHAR must be ≥ 0. If it is not, it is assumed to be zero and no label is displayed for the axis.

AXLEN must be such that YIN + AXLEN ≤ INMAX, or wraparound will occur. No check is made for wraparound in this routine, however.

Programming Notes:

Small tic marks are placed on the axis every SMTIC "inches". Larger tic marks are drawn every BIGTIC "inches".

The precision of the values displayed beneath tic marks is controlled by a call to AXPREC (paragraph I.2.3). If this routine has not been called, the default precision stated in paragraph I.2.3 applies.

The calling program must insure that the plot origin is set far enough from the edge of the screen that the axis annotation will fit between the axis and the edge of the screen; about 0.6 "inch" should be allowed.

Side Effect: The beam position is moved to the plot origin (XORG, YORG), which has been defined either by default or by a previous call to INIT.

I.2.33 YOURSC(SF,VLO,IWHO,ITIME)

Purpose: To allow the calling program to set an arbitrary scale factor and least plottable value for a specified axis.

Parameters:

SF is a floating point scale factor, which must be ≥ 0. Further information about the use of SF is given in paragraph I.2.21.

VLO is a floating point value which defines the least value (in user units) plottable; VLO is the value in user units associated with the plot origin (XIN,YIN) set either in a call to INIT or by default.

IWHO specifies the axis for which scale factors are being set. The following values apply, as in other routines:

IWHO = 0 means X-axis
= any other value means Y-axis

ITIME specifies whether the user units for this axis are to be interpreted as time. Possible values for ITIME are discussed in paragraph I.2.21.

Error Conditions:

SF must be ≥ 0. If SF is negative, the old values for SF and VLO are left unchanged.

Programming Note:

It should be noted that a call to YOURSC does not set the plot clipping window. The clipping window is set implicitly in calls to QCALE and SCALE and can be set explicitly in calls to SETWIN.

Side Effect: The setting of SF and VLO also affects the values used for screen interrogation, as discussed in the description of QCALE.

APPENDIX II

DETAILS OF TERMINAL PROGRAM GRAPHICS PICTURES

II.0 INTRODUCTION

This appendix contains details about the structure of each of the terminal program permanent graphics pictures discussed in paragraph 4.3 of the main body of the paper. A paragraph is devoted to each of pictures 1 through 8 (with the exception of picture 7, which is not allocated or used); the paragraph describes the layout of the picture and the actual display list commands used. In this appendix, where the context does not make the interpretation clear, octal numbers are distinguished by the suffix K.

II.1 Picture 1

This is a 30-word picture which was reserved for display of error messages, but which is not used in the current version of the terminal program. Error messages are included in the scrolled list of ASCII messages (picture 8) instead. This picture consists only of a standard header and trailer (as described in paragraph 4.3.2).

II.2 Picture 2

This 30-word picture displays the bias time window on the screen. The bias time is initially generated by a TMS application program in the MODCOMP and sent to the NOVA in a graphics packet (see paragraph 3.1 and paragraph I.2.4). The bias time is updated by the CS task each time curve sliding is performed by the amount of sliding (each time one of the function keys to skew curves is depressed the bias time is updated by a fixed amount).

The picture begins with a standard header, which is followed by a two-word absolute vector display list command which positions the beam to the location specified by the application programmer for the time window. Ten words of display processor character commands then follow to display a

time in the format ±hhh:mm:ss in size 2 characters. The picture is concluded with a standard trailer.

II.3 Picture 3

This 30-word picture is similar in function to picture 2, in that picture 3 displays a window to show the coordinates of the joystick cursor. The joystick coordinates are given as degrees Fahrenheit (Y-position) and time (X-position).

The picture is composed of a standard header, followed by a two-word absolute vector command to position the beam to a fixed point for display of the window. The location of the window is currently set to be (13.2,15.4) in "inches" (see I.2.0.1), which is equivalent to (3379,3942) in rasters. Sixteen words defining the position in the format ±dddF ±hhh:mm:ss then follow. The picture is concluded with a standard trailer.

As with the window in picture 2, the window in picture 3 is updated in place without stopping or excluding the display processor (see paragraph 4.3.4).

II.4 Picture 4

Picture 4 is also a 30-word picture, but its function is more complex. The picture contains display list commands to generate a graphics cursor (a plus sign) and, when needed, a text cursor (underline) which is attached to the graphics cursor. Figure II.4-1 shows the format of the picture.

Picture 4 then (words 8-14) draws a plus sign with arms 12 rasters long and centered at the joystick location. Depending on the setting of word 16 (controlled by the JS task), the display processor will either skip around the instructions to generate the underline attached to the plus sign (if no text cursor is to be displayed), or will draw a 12-raster-wide underline just below the plus sign.

WORD	OCTAL CONTENTS	COMMAND
1-5		Standard header
6 7	134000 address in module JGTON	JSR (jump to subroutine) to location which contains an absolute vector command to position beam to joystick location
8	172000	SPECIAL FUNCTION (shift into extended format)
9	003000	Short relative vector to move right 12 rasters invisibly
10	072000	Short relative vector to move left 24 rasters visibly
11	003014	Short relative vector to move right 12 rasters and up 12 rasters invisibly
12	040150	Short relative vector to move down 24 rasters visibly
13	000014	Short relative vector to move up 12 rasters invisibly
14	170000	SPECIAL FUNCTION (exit extended format mode)
15	130000	JUMP
16	*+1 or *+5	Address to which jump should go (*+1 if text cursor is to be displayed and *+5 otherwise)
17	176000	SPECIAL FUNCTION (shift into extended format and expanded resolution mode)
18	036567	Short relative vector to move down 9 rasters and left 6 rasters invisibly
19	043000	Short relative vector to move right 12 rasters visibly
20	170000	SPECIAL FUNCTION (exit extended format and expanded resolution modes)
21-24		Standard trailer

The initial JSR in the picture transfers briefly to several locations in the MEGATEK-supplied joystick handler in module JGTON. In JGTON is an absolute vector command to move the beam invisibly to the current location of the joystick, and then a return-from-subroutine instruction.

Figure II.4-1. Format of Picture 4

JS performs an additional manipulation of picture 4 when the user depresses the button on the end of the joystick. When JS senses that the button is depressed (this information is made available by JGTON), word 8 of the picture is changed to 176000K, which sets expanded resolution mode for the cursor drawing (see [14]) and causes the plus sign size to be quadrupled.

All of the updates to this picture (words 8 and 16) are made without stopping or excluding the display processor.

11.5 Picture 5

Picture 5 is a 100-word picture controlled by the KB task for display of characters in the current command line (the line on which the user composes commands to be sent to the MODCOMP). The picture begins with a standard header and a two-word absolute vector command to position the beam to the current cursor location. Initially, the cursor is at (0.2,0.6) in "inches" ((48,143) in rasters); the Y-position never changes, of course, but the X-position is updated as the cursor moves. A two-word long relative vector (120044K and 100000K) then follows to draw a 36-raster-wide underline (for a cursor). A two-word absolute vector (014060K and 174224K) then positions the beam to the first of the line again, and 72 words of size 2 character commands follow for the characters in the command line. The picture is concluded with a standard trailer.

The character commands are set to display blanks when no characters have been typed and are overstored by KB as characters are received from the keyboard. KB also modifies word 9 of the picture (the second word of the long relative vector) when the text cursor is to be displayed as attached to the graphics cursor (see II.4). When this situation obtains, KB changes word 9 to b3 0, which causes the relative vector's underline to be invisible (see [14] for a detailed discussion of the significance of the bit positions of each display list command).

The modifications to picture 5 are accomplished without stopping or excluding the display processor.

11.6 Picture 6

Picture 6 is a 400-word picture controlled by KB for displaying the plot annotation typed by a terminal user. The picture begins with a standard header and a two-word absolute vector to position the text cursor. A two-word long relative vector (120044K and 0, normally) then follows to draw a 36-raster-wide underline invisibly as the text cursor. This cursor is made visible (by replacing the 0 with a 100000K) only when annotation of the screen is being performed.

The contents of the rest of the picture depend on what annotation has been entered. For each tab position, a two-word absolute vector is required to position the beam. This command is then followed by words containing character commands (size 2 characters) for the characters typed. The entire picture is terminated with a standard trailer.

A tab position is defined each time the terminal user "grabs the cursor" (by depressing the GRAB CRSR function key) and then types displayable characters. "Grabbing the cursor" causes the command line cursor (picture 5) to become invisible and the text cursor attached to the joystick cursor (picture 4) to become visible. The user can then position the text cursor to a desired point and type a comment on the screen. When he begins typing, the text cursor in picture 4 is made invisible, and a tab position is defined by insertion of an absolute vector in picture 6. The picture 6 text cursor is set to be placed at the location of the joystick cursor and is made visible. The character commands for the keys typed are then added until the annotation at the present position is ended by a carriage return or a TAB, BACK TAB, or END NOTE keyin.

Because of the complexity of updating this picture, the display processor is excluded from picture 6 by the first method described in paragraph 4.3.2.

11.7 Picture 8

Picture 8, at 1,500 words, is the largest of the permanent pictures. Picture 8 is controlled by KB through the subroutine SCROL and is used to display the last 20 lines of ASCII commands or messages from the MODCOMP. The picture begins with a standard header, which is then followed by 20 "line units", and a standard trailer. Each "line unit" is composed of a two-word absolute vector to position the beam to the beginning of the line, and 72 words of character commands for the characters in each line. The character words in each line units are always filled (with commands to generate either blanks or displayable characters).

SCROL makes lines visible or invisible by turning the intensity bits for each absolute vector (the first 4 bits of the second word) on or off, as necessary. When a new line is to be added to the display and the last line scrolled off, SCROL first passes through the line units and adds 72 rasters to the Y-position (second word of the absolute vector) for each line. The absolute vector for the scrolled-off line is then initialized to the position of the lowest line [(0.2,1.1) in "inches", or (48,280) in rasters] and the new characters are inserted in that line unit.

The updates to picture 8 are accomplished without stopping or excluding the display processor.

REFERENCES

[1] Hopkins, G. T. A Bus Communications System, The MITRE Corporation, MTR-3515, November 1977.

[2] Roman, G. S. The Johnson Space Center Broadband Communications System, The MITRE Corporation, MTR-3621 (JSC #14495), 1 July 1978.

[3] Brown, J. S. and Weinrich, S. S. Trend Monitoring System (TMS) Communications Hardware - Volume I - Computer Interfaces, The MITRE Corporation, MTR-4721 (JSC #14682), February 1979.

[4] Brown, J. S. and Hopkins, G. T. Trend Monitoring System (TMS) Communications Hardware - Volume II - Bus Interface Units, The MITRE Corporation, MTR-4721 (JSC #14723), March 1979.

[5] Brown, J. S. and Lenker, M. D. Trend Monitoring System (TMS) Communications Software - Volume I - Computer Interfaces, The MITRE Corporation, MTR-4723 (JSC #14792), April 1979.

[6] Gregor, Paul J. Trend Monitoring System (TMS) Communications Software - Volume II - Bus Interface Unit (BIU) Software, The MITRE Corporation, MTR-4723 (JSC #14793), April 1979.

[7] Brown, J. S. and Lenker, M. D. Diagnostic Procedures for Trend Monitoring System (TMS) Communications, The MITRE Corporation, MTR-4724 (JSC #14794), April 1979.

[8] Brown, J. S. Procedures for Building Trend Monitoring System (TMS) MODCOMP Graphics Library and MEGATEK Terminal Program, The MITRE Corporation, WP-6214 (JSC #14826), March 1979.

[9] Lenker, Mike. Trend Monitoring System Preliminary Design Study, The MITRE Corporation, MTR-4705, September 1977.

[10] GRAPHELP User's Guide (Version 2), ID 422431-1506, IMLAC Corporation, February 1977.

[11] Functional Design Specification - Trend Monitoring System, NASA Johnson Space Center, JSC #13900, February 1978.

[12] Trend Monitoring System Level "C" Requirements Document, 78-FD-005, NASA Johnson Space Center (JSC #13992), November 1978.

[13] Herndon, E. S. Letter to C. G. Krpec, NASA/JSC (FD7), Subject: "Program Documentation for MITRE Portions of the Trend Monitoring System (TMS) Software", The MITRE Corporation, D72-L-426/HO, July 2, 1979.

[14] MEGATEK MG-552 Graphics Display Frocessor – MEGRAPHIC 5000 Series Product Description, MEGATEK Corporation, 1977.

[15] Real Time Operating System Reference Manual, Data General Corporation, Publication 093-000056-06, February 1975.

[16] Herndon, E. S. Letter to C. G. Krpec, NASA/JSC (FD7) regarding changes to MEGATEK keyboards, The MITRE Corporation, D72-L-356/HO, October 25, 1978.

[17] MEGATEK 5000 Series Software Manual - FORTRAN, Revision 1.3, MEGATEK Corporation, December 6, 1977.

End Date
Filmed
4-4-80