Whizzard® 1600 Family

1645 Graphics Protocol

0251-0012-01

Making History out of State-of-the-Art.

MEGATEK CORPORATION
A UNITED TELECOM COMPANY

WHIZZARD® 1645 GRAPHICS PROTOCOL

251-0012-01

Change -1

April 1985

This manual has been prepared for MEGATEK personnel, licensees and customers solely for use in connection with equipment supplied by MEGATEK. Information contained herein is the property of MEGATEK. Reproduction, in whole or in part, shall not be undertaken without prior written authorization of MEGATEK Corporation.

Users are cautioned that MEGATEK reserves the right to make changes without notice in the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including, but not limited to typographical, arithmetic, or listing errors.

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Graphics Engine, Life, Megatek, Megateknology, Rasterizor, and Wand are trademarks of Megatek Corporation. Whizzard and Template are registered trademarks of Megatek Corporation.

PREFACE

MANUAL OBJECTIVE AND STRUCTURE

The Whizzard 1645 Graphics Protocol is a reference source for experienced system operators.

The manual is divided into five sections and four appendices.

Section One gives a brief overview of the system, focusing on its high-level functionality and sophisticated hardware.

Section Two discusses the Whizzard 1645's initialization diagnostics, operation modes, memory management, and command syntax conventions.

Section Three covers the Whizzard 1645's processor control commands. Included in this part are discussions of the commands that control the operation of the Whizzard 1645 system: the system and task control command and two memory access commands (Read Current Position ID command and Read Free Memory Size command). The commands that control the division and manipulation of segments and subroutines are also outlined in this section. The segment control commands encompass the Open, Append, Clear, Delete, Close, Rewrite, Rename, and Delete All commands. The subroutine control commands include the Rewrite, Open, Append, Clear, Delete, Delete All, Call, and Rename commands.

Section Four presents the Whizzard 1645's control commands that affect the images produced on the screen. The command groups included here are screen handling, set options, insert/change attributes, transform parameters, vector, fill text, markers, pixel control, and pixel manipulation.

Section Five describes the Whizzard 1645's peripheral controls. The peripheral devices that attach to the Whizzard 1645 system are a keyboard, data entry keypad, joystick, tablet/digitizer, pick module, function keys/lights, and valuators. These devices respond to the Enable, Disable, Sample, and Read commands individually described here.

Detailed information referenced in the manual is contained in three appendices at the back of the manual.

Appendix A lists all the commands discussed in Sections One through Five.

Appendix B contains data on matrix transformations.

Appendix C serves as a reference tool on the values of attributes and presents default attributes that are initially set by the Whizzard 1645 firmware and held in the current header.

Appendix D contains a glossary of the terms used in the manual.

ASSOCIATED DOCUMENTS

Additional documentation is included in the following manual:

Whizzard 1645 Operator's Guide

Basic Keyboard (LE-100)

TABLE OF CONTENTS

LIST OF TABLES

1.0 SYSTEM OVERVIEW

Megatek's Whizzard 1645 family of graphics terminals enhances the extensive capabilities of the Whizzard graphics systems by bring high-quality computer graphics to an even wider range of users. A combination of functionality and systems diagnostics in the low-cost Whizzard 1645 terminal allows the Whizzard 1645 protocol and hardware to handle intricate and complicated displays easily.

1.1 High-Level Functionality

1.1.1 Distributed Functionality

Intel 8085 and 8086 microprocessors permit off-loading of such housekeeping functions as display list management, memory, peripheral interaction, and response to high-level commands from the host computer.

1.1.2 Memory Management

The Whizzard 1645 firmware makes possible the expansion and deletion of memory space assigned to specific segments.

Maintenance of the display list is through the use of a block structured system. In this system, memory is taken to be discontiguous. Blocks of memory are attached as needed to build a picture. These blocks are of fixed length, typically 256 bytes, and are managed by a group of internal software routines. If portions of the picture are deleted, memory is reclaimed for use a block at a time.

1.1.3 Input Device Handling

Intelligent device handling of Whizzard peripherals is standard in the Whizzard 1645, providing the functionality required to sample, write to, read from, enable, and disable the following devices:

- Keyboard
- Data Entry Keypad
- Valuators
- Joystick
- Tablet
- Pick Module
- Function Keys/Lights

System users also have their own input queue. Data from the input devices can be held in the queue until the host computer requests it.

1.2 Sophisticated Hardware

1.2.1 Electronics

The Whizzard 1645 electronics are contained in a single printed circuit board. The design is based on the Intel 8066 microprocessor, which is configured with 192K bytes of EPROM memory and three programmable interrupt controllers. The standard Whizzard 1645 memory contains 128K bytes of dynamic Random /access Memory (RAM). Up to 384K additional bytes can be added to accommodate larger display list.

1.2.2 Terminal Characteristics

The Whizzard 1645 provides English language menus that allow the user to select such functions as interface characteristics, ANSI terminal characteristics, and graphics terminal characteristics (discussed in the Whizzard 1645 Operator's Guide). These characteristics can also be set by the host computer.

1.2.3 Speed

The Whizzard 1645 processes graphics and ANSI data over the RS-232C serial interface at a maximum speed of 19.2K baud. Selectable automatic XON/XOFF hand-shaking allows both the host computor and terminal to communicate at their maximum rates without losing data due to buffer overflow.

2.0 INTERFACE CHARACTERISTICS

This section treats the characteristics of the Whizzard 1645 interface to the host computer. Specifically discussed here are:

- Initialization Diagnostics
- Operation Modes
- Memory Management
- Command Syntax Conventions

For a more complete discussion of terminal and communication setup options, see the Whizzard 1645 Operator's Guide.

2.1 Initialization Diagnostics

When power is applied to the Whizzard 1645, several diagnostic tests are performed to assure proper function: the Whizzard 1645 automatically performs the power-up self-test. The keyboard function lamps light sequentially and then turn off. A two-tone beep sounds. The CUSR MODE lamp located on the keyboard is turned on.

If the keyboard does not light up and beep, the user should check the coiled cord connecting the keyboard to the electronic base and turn the power switch off and on again.

2.2 Operation Modes

The Whizzard 1645 processes characters sent from the host computer in one of two modes: terminal mode and graphics mode.

2.2.1 Terminal Mode

When the system is initialized, it functions like a standard alphanumeric terminal (terminal mode). All characters are placed into a scrolling monitor buffer in the Whizzard 1645 display list memory and displayed on the screen. (Refer to Whizzard 1645 Operator's Guide for a complete description of alphanumeric terminal mode programming for the Whizzard 1645).

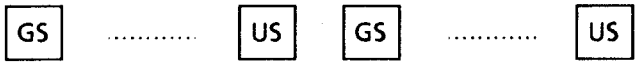
2.2.2 Graphics Mode

Graphics mode is entered when the ASCII control character GS is sent to the terminal. The GS character is followed by one or more commands which consist of a sequence of characters that invoke (under the control of the Whizzard 1645 firmware) the graphics capabilities of the terminal. Graphics mode is exited by the ASCII control character US.

Graphics Mode Commands

A graphics mode command consists of one or two command characters followed by the necessary parameters in the form of additional characters.

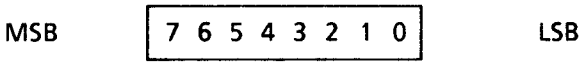
The nature of these parameters depends on the particular command. Another command may follow immediately after the last parameter required by a command, allowing for any number of graphics operations without the overhead of sending a GS character (to reenter the graphics mode) of sync due to a lost character. The US character ends the command and GS starts a new command before every graphics command. To ensure the integrity of the graphics data, the US character should be positioned after the last parameter of a command, followed by a GS, the next command, and so on, in order.



To provide resyncing of the processor, and to prevent getting the commands out of sync due to a lost character, the US character ends the command and GS starts a new command.

Least Significant/Most Significant Bits

With one exception, only the least significant six bits (0-5) of the transmitted characters are used for graphics mode commands and responses. The next most significant bit (6) is set so that the resulting bits always form a printable character. Bit 6 is set to 1, except when the six data bits are all set to 1. Then bit 6 is set to 0 to avoid transmitting the delete character. Bit 7 is the parity bit.



Several control characters are disregarded when the Whizzard 1645 is in the graphics mode preventing erroneous interpretations of a command sequence. Control characters that are disregarded are the carriage return, line feed, back space, and horizontal tab. The interpretation of the CR character is defined in setup and can cause an exit from graphics mode if desired. The US character is the only character that is never ignored.

When text is sent as graphics, characters through the *Enter Character String* command (discussed in Section Four), all seven bits of these text characters are recognized. To terminate the text string, a GS character is sent, leaving the terminal in graphics mode and ready to accept a US character or another command. The most significant bit if the transmitted character (bit 7) is the parity bit set by the system hardware. The type of parity associated with this bit can be changed interactively through setup.

Refer to the Whizzard 1645 Operator's Guide for information on remote setup.

2.3 Memory Management

Memory management at the segment level is performed at the Whizzard 1645 graphics terminal. A segment is a logical, user-defined grouping of a graphics display. Segments in the Whizzard 1645 graphics terminal correspond to the segment concept of the SIGGRAPH-proposed CORE standards. The user's program in the host computer assigns a unique access number to each segment. Any operation to be performed on a segment must refer explicitly to this number. The Whizzard 1645 assigns the physical location of segments in display list memory; therefore, modification of the display list by the host computer is made through the segment operations provided for the system - e.g., *Open Segment, Close Segment, Delete Segment*.

Information sent by the host computer to be entered into the display list memory (including vectors, matrix elements, and dash control, etc.) is assembled into valid display list commands by the Whizzard 1645 firmware. Each display list command is written into the next available location in display list memory. The next available location, however, is not necessarily the next physical location in memory.

Memory management is transparent to the host computer, to which entry of data appears, as though it were in contiguous locations. The host computer accesses elements in the display list by segment number and offset from segment start, since it cannot associate display list elements with physical addresses. Offsets are measured by the number of commands and are not directly related to command length. Three commands of two-character length produce the same offset as three commands of seven-character length. When a segment is deleted, the firmware places all memory formerly used by the segment in the free memory list.

In processing the display list to produce an image on the screen or to repaint the image after the display list has been modified, the Whizzard 1645 graphics processor proceeds through the display list memory displaying all visible segments. If incremental repaint has been enabled, the image appears as the display list is being processed. If auto-repaint is enabled, the image appears when a repaint command is received from the host computer or after the end of the display list.

2.4 Command Syntax Conventions

Throughout the rest of this manual, command names in the text are shown in italics. The actual commands appear as a pair of lines, as illustrated below for the *Reset System* command:

```
(RESET)          (MODE)
101110           0000MM
  2   E
```

The first line represents the logical elements of the command as a series of mnemonics in parentheses. The second line shows the command as a series of six-bit groups. The groups may

- Present the values of the bits — e.g., "101110", with the hex value listed below — e.g., "2E".
- Include letter symbols in place of certain bits — e.g., "0000MM," or
- Show a name for the group — e.g., "device".

If a command verb refers to 12 bits rather than the usual six bits in the line below, the mnemonic is enclosed in double parentheses — for example,

```
((CURRENT STATUS))
111110           001011
  3   E            0   B
```

These symbols are defined in the explanation of each of the processor command protocols discussed in Section Three of this manual. In some cases where the command contains more elements than will fit between the margins of this manual, continuation lines are shown as if extending continuously to the right.

A lower case "r" denotes a bit reserved for future use. When sent to the Whizzard 1650 by the host computer, reserved bits should be 0; when received as data from the WHIZZARD system, they may be either 0 or 1.

Sixteen-bit data words are transmitted as groups of three 6-bit characters, as follows:

```
rrAAAA    BBBBBB    CCCCCC
```

The 16-bit word is AAAABBBBBBCCCCCC. The remaining two bits on the left, shown here as "rr," may be unused or contain other data, depending on the particular command.

3.0 PROCESSOR CONTROL COMMANDS

The command protocols discussed in this section control the operation of the Whizzard 1645 system and the division and manipulation of segments and subroutines.

The section is divided into five parts. Each deals with a different aspect of processor control commands.

Section 3.1 briefly describes the system and task control commands, which allow system reset.

Section 3.2 discusses the two memory access commands: (1) The *Read Current Position ID* command allows the host computer to determine locations in display list memory, (2) The *Read Free Memory Size* command allows the host computer to read the open working space in the system.

Section 3.3 describes segment control commands, which are used to create, delete, and modify segments.

Section 3.4 discusses subroutine commands. These commands create, delete, modify, and insert new material into segments and other frequently used units of graphics information called subroutines.

Section 3.5 outlines three special function commands, or functions, which modify a normal *Call Subroutine* command. The *Call Subroutine* command is described at the end of Section 3.4.

3.1 System Reset Command

The system can be reset with the *Reset Terminal* command from the host computer.

(RESET)	(MODE)
101110	0000MM
2 E	

MM	*Action*
00	No action
01	No action
10	Software Reset. All segments and subroutines are deleted. The default header is reset to its initial values (see Appendix C). Segment 0 is cleared. Peripheral devices are disabled. The Whizzard 1645 display is returned to ANSI mode if the issuing task is the host. The host computer is not required to delay.
11	Invalid

3.2 Memory Access Commands

Memory access commands allow the host computer to determine locations in display list memory and to read the open working space in the system. The two memory access commands, *Read Current Position ID* and *Read Free Memory Size*, are summarized in Table 3-1 and discussed in the following pages.

Table 3-1 Memory Access Commands

Binary	Hex	Total Bytes	Command
111100	3C	2	Read Current Position Identification
111110	3E	2	Read Free Memory Size

3.2.1 Read Current Position ID

(READ CURRENT POS ID)	(MODE)
111100	rrrrrr
3 C	

The current position insert pointer indicates the location in display list memory where the next graphics command for the currently open segment or subroutine is to be written.

The mode character "r" is reserved for future use and must be set to 0.

The current position and segment or subroutine number representing the displacement of the insert pointer are returned. The returned data is in the following format:

IS0000	msseg	lsseg	00NNNN	NNNNNN	NNNNNN	term

I *Status of Read*
0 Valid; ID and seg/subr number follow
1 Invalid; the response terminator follows immediately after this character

S *Action*
0 msseg-lsseg is a segment number
1 msseg-lsseg is a subroutine number (12-bit segment or subroutine number)

NNN...NNN Forms the 16-bit ID, which represents a location in display list memory; this ID is interpreted only by the Whizzard 1645 firmware

term The response terminator selected in setup

3.2.2 Read Free Memory Size

((CURRENT STATUS))
111110 001011
3 E 0 B

The return data is in the following format:

000000 000000 000000 CCCCCC CCCCCC CCCCCC CCCCCC term

CC...CC Is the 24-bit count of number of 32 bit words remaining in vector memory

term The response terminator selected in setup

This command indicates the amount of memory, or open working space remaining within the system.

3.3 Segment Control Commands

A segment is the basic organizational unit of graphics display data supported by the Whizzard 1645 firmware. Each segment has a segment header containing information for identifying, referencing, and defining the segment. Display data physically located in the Whizzard 1645 display list memory can be entered, removed, and modified by various commands to the segment header.

Segments in the Whizzard 1645 system are subject to full 2D matrix transformation and clipping. A transformation module applies a 3 x 2 transformation matrix to the segment, allowing full rotation, scaling, and translation in two dimensions. (See Appendix B for details of the matrix transformations.)

There are two types of commands: implicit and explicit. The implicit commands include vector, character, and input attribute commands (covered in Section Four). The two types of explicit commands are segment control commands and change segment attribute commands. The discussion in this part of Section Three is concerned with the segment control commands used for creating, deleting, and preparing segments for modification.

Table 3-2 presents the segment control commands and their functions. Each command is discussed individually following the table.

Table 3-2 Segment Control Command Functions

Command	Function
Open Segment	Creates segment
Close Segment	Closes segment
Delete Segment	Deletes segment
Delete All Segments	Deletes segments
Clear	Prepares segments for modification
Append	Prepares segments for modification
Rewrite	Prepares segments for modification
Rename	Prepares segments for modification

The term "active segment" refers to the currently open segment. The "insert pointer" refers to a particular location within that segment. The following commands affect the active segment and the insert pointer:

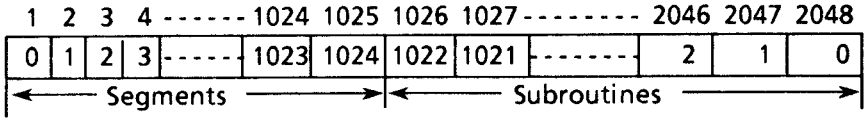
- Segment control commands
- Vector commands
- Character commands
- Insert attribute commands

These commands place graphics data, such as moves and draws, into the active segment at the location of its insert pointer.

Segment headers are created when segments are opened and defined by the host computer. The segment header appears at the top of each segment and contains the following information for identifying and referring to the particular segment:

1. Segment Number - A unique 12-bit number ranging from 0 to 2047 (a total of 2048 segments and subroutines), used for identifying and referring to each segment. The segment number is assigned when a segment is created. Because of dynamic allocation and deallocation of segments and subroutines, the higher the segment number is, the fewer subroutines you have to use.

   Example: If the highest segment number is 2047, there will be no subroutines available. If the highest segment number is 1024, the subroutines may be allocated between 0 and 1022. In this case, the total of 2048 segments and subroutines are divided into 1025 (0-1024) segment numbers and 1023 (0-1022) subroutine numbers which can be used.



2. Current Header - A special header maintained by the Whizzard 1645 firmware, which contains all the attributes, or default values, and transformation parameters required by the segment headers. When the *Open Segment* command is used to create an open retained segment, the graphics data of the default header are copied into the header of the newly opened segment. The header in the open segment is now know as a current header, and the attributes and transformation parameters contained therein are called current values.

The current header for a segment contains attribute settings for the following:

- Color
- Blink rate
- Dash enable
- Dash pattern
- Segment origin
- Visibility
- Detectability
- Matrix elements (outlined in Appendix B)
- Clipping boundaries

The attribute definitions in the default header will affect all graphics data placed in the segments (and subroutines) until a new definition is inserted in the segment. With the exception of visibility and detectability, the attributes and transformation parameters defined in the default header may be overridden by inserting new definitions in the segment. Insert attribute commands placed into any other segment besides the default segment do not affect the values in the header. A new definition

overrides the previous setting in the header and affects all graphics data subsequently added to the segment.

Appendix C presents default attributes that are initially set by the Whizzard 1645 firmware and held in the current header.

The attributes and transformation parameters held in the header of a newly opened segment may be changed at any time by using the change segment attribute commands (discussed in Section Four). These commands must specify the segment that is being changed. They do not change which segment is the active segment nor do they change the current values, However, the change segment attribute commands do permit headers of inactive segments to be altered while data is being entered into the active segment.

The remainder of Section 3.3 discusses the segment control commands outlined in Table 3-3.

Table 3-3 Segment Control Commands

Binary	Hex	Total Bytes	Command
000000	00	4	Open Segment
000001	01	3	Append Segment
000010	02	3	Clear Segment
000011	03	3	Delete Segment
000100	04	1	Close Segment
000101	05	8	Rewrite Segment
000110	06	5	Rename Segment
000111	07	1	Delete All Segments

3.3.1 Open Segment

This command creates a "retained" segment. A segment will exist as a retained unit of display data until it is removed by either the *Delete Segment* or *Delete All Segment* command.

(OPEN SEG)	(VISIB)	(SEG NO)
000000	rrrrrV	msseg-lsseg
0 0		

V *Visibility*
0 Specifies visibility OFF
1 Specifies visibility ON

msseg-lsseg Forms the 12-bit segment number.

Opening a New Segment

1. A new segment is opened and assigned a segment number (SEG NO). If SEG NO refers to an existing segment, the system ignores the command.

2. When a segment is opened, the system firmware automatically inserts a segment header in the appropriate format at the beginning of the segment. (As discussed earlier, the current header is a special header maintained by the Whizzard 1645 firmware, which contains all the attributes and transformation parameters required by the basic and transform segment headers. See Appendix C for an outline of the default attributes held in the current header.)

3. When a retained segment is opened, any other open segment or subroutine is closed. The attributes of the current header are copied into the header of the newly opened segment to serve as default values.

   Segment 0, a special non-retained default segment, always exists. When no retained segment is open, graphics data sent to the Whizzard 1645 is placed in segment 0. Segment 0 has a header which is set to the initial current header. Any of the change attribute commands may be applied to the default header to change these settings.

   When an insert attribute command corresponding to one of the current values of an active segment 0 is executed, the attribute setting specified in the command sets the current value attribute definition. Insert commands in any segment other than 0 do not affect the current header.

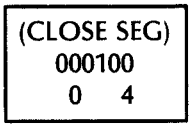
   Initially, there is no active segment. After a segment is closed, and until a new segment is opened or an existing segment reopened, there is no active segment. Whenever there is no active segment, graphics data sent to the system, such as a move or draw, is appended to the end of the non-retained segment 0. Segment 0 has a transform header which is set to the initial current header. Any of the change attribute commands may be applied to the default header to change these settings.

If the segment becomes active, the insert pointer is positioned immediately after the segment header.

4. As data is entered into the segment, the insert pointer moves up to follow the last item entered. The insert pointer automatically points to the next available location following the inserted data. (Such updating reduces communications requirements, since the segment and location are not explicitly identified every time data is entered. Only segment control commands can change which segments are active.)

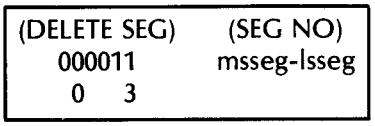
5. VISIB controls the visibility attribute of the segment and the segment visible (ON) or invisible (OFF) on the screen.

3.3.2 Close Segment



This command closes the active segment, leaving no explicitly active segment. If a user attempts to enter graphics data before another segment is opened or reopened, the data is appended to the default segment, segment 0. The *Close Segment* command makes segment 0 the active segment and positions the insert pointer at the end of segment 0. If the *Close Segment* command is executed when no segment is explicitly active, it has no effect.

3.3.3 Delete Segment

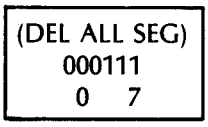


msseg-lsseg Forms the 12-bit segment number

This command deletes the segment number (SEG NO). The segment ceases to exist, and the SEG NO becomes available for reassignment to any new segment that may be opened. If the segment is visible, it will disappear from the screen. If the segment is active when the command is issued, it is closed before it is deleted. This has the same effect as if the *Close Segment* command had been issued before the *Delete Segment* command. If SEG NO refers to an inactive segment, the *Delete Segment* command does not affect the active segment or the insert pointer. All memory allocated to the deleted segment is freed for reuse.

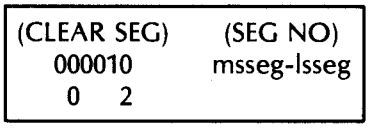
When the *Delete Segment* command is applied to a non-existent segment, the Whizzard 1645 ignores the command. If segment 0 is specified, the *Delete Segment* command clears the contents of that segment and the default attributes are restored to the default header.

3.3.4 Delete All Segments



This command closes all open segments and deletes all existing retained segments and the contents of segment 0.

3.3.5 Clear Segment

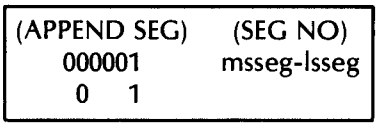


msseg-lsseg Forms the 12-bit segment number

This command reopens the segment number (SEG NO) for input, making it the active segment and positioning the insert pointer immediately after its header. The previous contents of the segment are deleted, and the memory formerly used by the contents is freed for reuse. The header is left intact. None of its attributes or transformations are altered. As data is entered into the segment, the insert pointer moves up to follow the last data element entered.

If SEG NO refers to a non-existent segment, the command has no effect. If the *Clear Segment* command is issued while a segment is active (the segment was never closed), the segment is automatically closed before the *Clear Segment* command is executed. If segment 0 is specified, it is cleared like any other segment.

3.3.6 Append Segment



msseg-lsseg Forms the 12-bit segment number

The segment number (SEG NO) is reopened for input, making it the active segment, and the insert pointer is positioned at the end of the segment. As data is entered into the segment, the insert pointer moves up to follow the last element entered.

If SEG NO refers to a non-existent segment, the system ignores the command. If the subroutine is active when this command is issued, the subroutine is first closed as by a *Close Subroutine* command. Then the *Append Subroutine* command is executed.

3.3.7 Rewrite Segment

(REWRITE)	(SEG NO)	(ID)			(OFFSET)
000101	msseg-lsseg	00IIII	IIIIII	IIIIII	msdis-lsdis
0 5					

msseg-lsseg — Forms the 12-bit segment number

III...III — Is a 16-bit identifier (ID), obtained by the *Read Current Position ID* command (discussed in Section Two), which identifies a location in display list memory

msdis-lsdis — Is the 12-bit OFFSET (in commands) from the ID to where the rewrite function is to start. OFFSET is an unsigned integer ranging from 0 (rewrite at LABEL) to 4095 (rewrite at LABEL + 4095)

This command allows any command(s) in a segment to be overwritten.

The segment number (SEG NO) is reopened for input, making it the active segment, and the insert pointer is positioned according to ID and OFFSET. ID is a 16-bit value that represents a location in display list memory. It is sent to the host computer as a result of the *Read Current Position ID* command. This ID may be interpreted only by the Whizzard 1645 firmware.

OFFSET is specified as a 12-bit number ranging from 0 to 4095. It represents the number of commands after which rewriting will begin. The insert pointer is positioned after the segment ID by the number of commands specified in OFFSET. If OFFSET is equal to N, the insert pointer is place at ID + N in the segment. For example, if OFFSET is specified as 0, the contents of the segment will be overwritten at ID.

As data is entered into the segment, the previous contents of the segment are overwritten and the insert pointer moves up to follow the last data element entered. If auto-repaint is enabled, the screen is repainted after every complete command received. Auto-repaint can be disabled and a new rewrite performed. To prevent flashing of the screen and a possible erroneous display of data, the rewrite should be the same length as the one it is replacing.

If data is entered after the last word in the segment is overwritten, the insert pointer continues to move to follow the last element entered, so that the data is appended to the end of the segment, as in the *Append Segment* command. If SEG NO refers to a non-existent segment, the *Rewrite Segment* command becomes an *Append Segment* command for segment 0. If the *Rewrite Segment* command refers to an active segment, that segment is automatically closed and reopened before the *Rewrite Segment* command is executed.

CAUTION

The *Rewrite Segment* command must be used with extreme caution. Data entered by the *Move/Draw Short Relative Vector String* and *Enter Character String* commands (covered in Section Four) should not be overwritten under any circumstances. If this data is overwritten, the result is unpredictable - possibly resulting in a "crash" of the Whizzard 1645 system, or in the loss of all graphics data in the Whizzard 1645 system.

3.3.8 Rename Segment

(RENAME SEG)	(OLD SEG NO)	(NEW SEG NO)
000110 0 6	msold-lsold	msnew-lsnew

msold-lsold Forms the old segment number

msnew-lsnew Forms the new segment number

Segment number msold-lsold is changed to msnew-lsnew. No action is taken if the old segment does not exist, if the new segment already exists, or if either segment number given is 0. Any open segment or subroutine will be closed before the *Rename Segment* command is executed.

3.4 Subroutine Control Commands

Both display list memory and transmission time can be saved by placing frequently used graphics information in subroutines. As with segments, subroutines are identified and referred to by a unique subroutine number that is assigned when the subroutine is created. Subroutine numbering is left to the user with the restriction that the sum of the highest segment number used, and the highest subroutine number used, cannot be greater than 2046.

The subroutine control commands used to create subroutines, delete subroutines, prepare subroutines for modification, and insert subroutine calls in segments or other subroutines are:

- *Open Subroutine*
- *Append Subroutine*
- *Clear Subroutine*
- *Delete Subroutine*
- *Call Subroutine*

Subroutines can be renamed using the *Rename Subroutine* command, rewritten using the rewrite commands, or moved by the *Special Subroutine Move* command.

The following commands will close any segment or subroutine that is open at the time of the command:

- *Open Subroutine*
- *Append Subroutine*
- *Clear Subroutine*

These commands make subroutines active. The insert pointer is positioned in the subroutine, at a location specified by the command, in order that all vector, character, and insert attribute commands (discussed in Section Four) can be used to place graphics data in the subroutine. The insert pointer is moved up automatically so that it always points to the next available location following the inserted data (in the same way it functions in segments).

Calls to subroutines can be placed anywhere in any segment, and subroutines can be nested. Up to 64 levels of subroutine nesting are allowed. Recursion is not allowed; that is, a call to subroutine number N cannot be placed in subroutine N. No recursion checking is performed.

Unlike segments, there are no distinguishing types of subroutines, and no header appears at the beginning of a subroutine when it is opened. Attributes or transformation parameters are inserted in subroutines by an insert attribute command or by one of the vector character commands specifying attributes. Unless one of these methods is used, the attributes and transformations applied to the contents of the subroutine can be drawn on the screen in many different ways if the subroutine is called in segments with different attributes or transformations.

The attributes and transformation parameters specified within subroutine override any others in effect when the subroutine is called. These new definitions are established and remain intact for all graphics displayed following the subroutine call; therefore, caution should be exercised when inserting such specifications in subroutines.

Starting at the insert pointer, data destined for a non-existent subroutine is entered in any currently open segment or subroutine. If no segments or subroutines are open, it is placed in the default segment. In all cases, this data is added at the insert pointer.

Deleting a subroutine does not remove any of the calls to that subroutine. Calls to a non-existent subroutine have no effect. Any new subroutine can be opened and assigned the number of a deleted subroutine. A call to the deleted subroutine will call the new subroutine. Any data entered into the new subroutine appears on the screen wherever the deleted subroutine appeared before. Calls may be made to subroutines that have not yet been defined but that will be defined at some future time.

Table 3-4 presents the subroutine control commands used in the Whizzard 1645 system. These will be individually discussed in this part of Section Three.

Table 3-4 Subroutine Control Command

Binary	Hex	Total Bytes	Command
000101	05	8	Rewrite Subroutine
100000	20	3	Open Subroutine
100001	21	3	Append Subroutine
100010	22	3	Clear Subroutine
100011	23	3	Delete Subroutine
100100	24	1	Close Subroutine
100101	25	3	Call Subroutine
100110	26	5	Rename Subroutine
100111	27	1	Delete All Subroutine

3.4.1 Open Subroutine

(OPEN SUB)	(SUB NO)
100000 2 0	mssub-lssub

mssub-lssub Forms the 12-bit subroutine number

The command opens a new subroutine and assigns a subroutine number (SUB NO). The new subroutine becomes the active subroutine, and the insert pointer is positioned at the first location in the subroutine. As data is entered into the subroutine, the insert pointer moves up to follow the last data element entered. If SUB NO refers to an already existing subroutine, or SUB NO plus the highest segment number used exceeds 2046, then the Whizzard 1645 system ignores the command.

If a segment or subroutine is active when an *Open Subroutine* command is issued, it is first closed (as if a *Close Subroutine* or *Close Segment* command were issued) before the *Open Subroutine* command is executed.

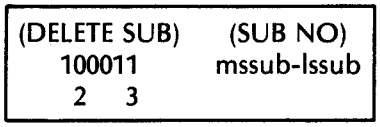
Any calls to SUB NO call the current subroutine of that number, even if those calls had originally referred to a different subroutine that was later deleted.

3.4.2 Close Subroutine

(CLOSE SUB)
100100 2 4

This command closes the active subroutine. If the *Close Subroutine* command is executed when no subroutine is active, the Whizzard 1645 system ignores it.

3.4.3 Delete Subroutine

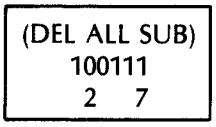


mssub-lssub Forms the 12-bit subroutine number

This command deletes the subroutine number (SUB NO). The subroutine ceases to exist, and the SUB NO becomes available for reassignment to any new subroutine that may be opened. If the subroutine is visible, it will disappear from the screen. If the subroutine is active when the command is issued, it is closed before it is deleted with the same effect as if the *Close Subroutine* command had been issued before the *Delete Subroutine* command. The *Delete Subroutine* command does not change which segment is active. All memory allocated to the deleted subroutine is freed for reuse.

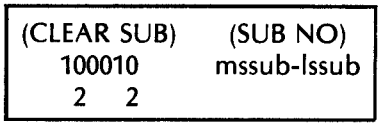
When the *Delete Subroutine* command is applied to a non-existent subroutine, the Whizzard 1645 system ignores the command.

3.4.4 Delete All Subroutines



This command deletes all subroutines and closes any open subroutines.

3.4.5 Clear Subroutine



mssub-lssub Forms the 12-bit subroutine number

This command reopens subroutine number (SUB NO) for input, making it the active subroutine and positioning the insert pointer immediately at the first location in the subroutine. The previous contents of the subroutine are deleted, and the memory formerly used by the contents is freed for reuse. As data is entered into the subroutine, the insert pointer moves up to follow the last data element entered.

If SUB NO refers to a non-existent subroutine, the command has no effect. If the *Clear Subroutine* command is issued while a subroutine is active, the subroutine is first closed as by a *Close Subroutine* command; then the *Clear Subroutine* command is executed.

3.4.6 Append Subroutine

(APPEND SUB)	(SUB NO)
100001	mssub-lssub
2 1	

mssub-lssub Forms the 12-bit subroutine number

This command reopens the subroutine number (SUB NO) for input, making it the active subroutine, and positioning the insert pointer at the end of the subroutine. As data is entered into the subroutine, the insert pointer moves up to follow the last data element entered.

If SUB NO is non-existent, the Whizzard 1645 ignores this command. If the subroutine is active when this command is issued, the subroutine is first closed as by a *Close Subroutine* command; then the *Append Subroutine* command is executed.

3.4.7 Rewrite Subroutine

(REWRITE)	(SUB NO)	(ID)	(OFFSET)
000101	msseg-lsseg	10IIII IIIIII IIIIII	msdis-lsdis
0 5			

msseg-lsseg Forms the 12-bit segment or subroutine number

III...III Is a 16-bit identifier (ID), obtained by the *Read Current Position ID* command, which identifies a particular location in display list memory

msdis-lsdis Is the 12-bit OFFSET (in commands) from the ID for the start of rewrite. OFFSET is an unsigned integer ranging from 0 (rewrite at LABEL) to 4095 (rewrite at LABEL + 4095)

This command allows any command(s) in a subroutine to be overwritten. The subroutine number (SUB NO) is reopened for input, making it the active subroutine, and the insert pointer is positioned according to ID and OFFSET. ID is a 16-bit value that represents an address in display list memory. It is sent to the host computer as a result of the *Read Current Position ID* command. This ID may be interpreted only by the Whizzard 1645 firmware.

OFFSET is a 12-bit number ranging from 0 to 4095. The insert pointer is positioned after the subroutine ID by the number of commands specified in OFFSET. If OFFSET is equal to N, the insert pointer is positioned at ID + N in the subroutine. For example, if OFFSET is specified as 0, the contents of the subroutine will be overwritten at ID.

As data is entered into the subroutine, the previous contents of the subroutine are overwritten and the insert pointer moves up to follow the last data element entered. If

data entered after the last word in the subroutine has been overwritten, the insert pointer continues to move up to follow the last data element entered so that data is added to the end of the subroutine (as in the *Append Subroutine* command).

If SUB NO refers to a non-existent subroutine, the *Rewrite Subroutine* command acts as an *Append Subroutine* command to segment 0. If the *Rewrite Subroutine* is issued while subroutine command is active, it is automatically closed before the *Rewrite Subroutine* command is executed.

CAUTION

The *Rewrite Subroutine* command must be used with extreme caution.Data entered by the *Move/Draw Short Relative Vector String* and *Enter Character String* commands should not be overwritten under any circumstances. If such data is overwritten, the result is unpredictable --possibly resulting in a "crash" of the Whizzard 1645 system or in the loss of all graphics data in the Whizzard 1645 system. Such a condition requires either a software reset or hardware reset using the 2ND FUNCTION/SOFT RESET key sequence.

3.4.8 Call Subroutine

(CALL SUB)	(SUB NO)
100101	mssub-lssub
2 5	

mssub-lssub Forms the 12-bit subroutine number

A call to a subroutine number (SUB NO) is placed at the insert pointer. One word is written into display list memory. All attributes and transformation parameters in effect at the time of the call apply to all graphics commands in the subroutine, unless other attribute and transformation definitions have been inserted in the subroutine. Any attribute or transformation parameter inserted in the subroutine applies not only to all graphics commands that follow in the subroutine, but also to graphics commands following the return from the subroutine until a new definition is encountered.

If SUB NO refers to a non-existent subroutine, the call is placed but nothing is displayed.

3.4.9 Rename Subroutine

(RENAME SUB)	(OLD SUB NO)	(NEW SUB NO)
100110 2 6	msold-lsold	msnew-lsnew

msold-lsold Forms the old subroutine number

msnew-lsnew Forms the new subroutine number

The specified subroutine, msold-lsold, is changed to msnew-lsnew. No action is taken if the old subroutine does not exist or if the new subroutine already exists. Any open segment or subroutine will be closed before the ***Rename Subroutine*** command is executed.

3.5 Special Subroutine Functions

The special subroutine functions described here allow three operations to be performed on a specific location in a display list. The location is uniquely identified by an ID derived from its address in memory. The three special functions differ from a normal *Call Subroutine* command in these ways:

1. Provide a single vector from the current position to the subroutine starting location on the screen.
2. Provide a different vector based on the same current position as before.
3. Substitute a different subroutine number at the identified instance.

Table 3-5.presents the special subroutine functions used in the Whizzard 1645 system. These are individually discussed in this section.

Table 3-5 Special Subroutine Functions

Binary	Hex	Total Bytes	Command
101000	28	9	Special 2D Subroutine Call
101000	28	11	Special 3D Subroutine Call
101000	28	12	Special 2D Subroutine Move
101000	28	14	Special 3D Subroutine Move
101000	28	9	Special Subroutine Substitution

3.5.1.1 Special 2D Subroutine Call

(SPECIAL FUNC)	(CALL)	(SUB NO)	parameters
101000	rrrr00	mssub-lssub	00DXYr xxxxxx xxxxxx
2 8			yyyyyy yyyyyy

mssub-lssub — Forms the number of the subroutine called

D	*Action*
0	Move
1	Draw

X,Y	*Vector Type for X and Y Coordinates*
0	Absolute
1	Relative

xx..xx — 12-bit X coordinates

yy..yy — 12-bit Y coordinates

The display of a subroutine number (SUB NO) is preceded by a vector move or draw as indicated. A 16-bit identifier for the specified instance of the subroutine is returned in the following format:

rrllll	llllll	llllll	term

3.5.1.2 Special 3D Subroutine Call

(SPECIAL FUNC)	(CALL)	(SUB NO)	parameters
101000	rrrr00	mssub-lssub	10DXYZ xxxxxx xxxxxx
2 8			yyyyyy yyyyyy
			zzzzzz zzzzzz

mssub-lssub — Forms the number of the subroutine called

D	*Action*
0	Move
1	Draw

X,Y,Z	*Vector Type for X, Y, and Z Coordinates*
0	Absolute
1	Relative

xx..xx — 12-bit X coordinates

yy..yy — 12-bit Y coordinates

zz..zz — 12-bit Z coordinates

A 16-bit memory address is returned from the system to indicate the location of the jump to subroutine command within the display list in the following format:

rrllll	llllll	llllll	term (0)

rrllll — Most significant 4-bits of the graphics memory address

ll...ll — Least significant 12-bits of the graphics memory address

term (0) — Represents the optional operator selected terminator string (CR, CR/LF, CR/EOT, CR/ETX)

3.5.2.1 Special 2D Subroutine Move

(SPECIAL FUNC)	(MOVE)	(SUB NO)	parameters
101000	rrrr01	mssub-lssub	00DXYr xxxxxx xxxxxx
2 8			yyyyyy yyyyyy
			Srllll llllll llllll

mssub-lssub	Forms the number of the subroutine or segment to be moved
D	*Action*
0	Move
1	Draw
X,Y	*Vector Type for X and Y Coordinates*
0	Absolute
1	Relative
xx..xx	12-bit X where SUB NO is to appear
yy..yy	12-bit Y where SUB NO is to appear
S	*Action*
1	Offset is from start of subroutine
0	Offset is from start of segment
ll..ll	Forms the 16-bit identifier of the instance

At the identified point in the display list, display of the subroutine or segment is preceded by a vector as specified in the command parameters. No data is returned.

3.5.2.2 Special 3D Subroutine Move

(SPECIAL FUNC)	(MOVE)	(SUB NO)	parameters
101000	000001	mssub-lssub	10DXYZ xxxxxx xxxxxx
2 8			yyyyyy yyyyyy
			zzzzzz zzzzzz
			Srllll llllll llllll

mssub-lssub — Forms the number of the subroutine or segment to be moved

D	*Action*
0	Move
1	Draw

X,Y,Z	*Vector Type for X, Y, and Z Coordinates*
0	Absolute
1	Relative

xx..xx — 12-bit X where SUB NO is to appear

yy..yy — 12-bit Y where SUB NO is to appear

zz..zz — 12-bit Z where SUB NO is to appear

S	*Action*
1	Offset is from start of subroutine
0	Offset is from start of segment

ll..ll — Forms the 16-bit identifier of the instance

The *Special 3D Subroutine* move command causes the subroutine positioning vector associated with a jump to subroutine command to be modified to give the appearance of moving the graphic image associated with the subroutine around the screen. The X, Y, and Z coordinates within the command are used as either an absolute location or relative vector from the current location as indicated by flags within the command.

3.5.3 Special Subroutine Substitution

(SPECIAL FUNC)	(SUBST)	(SUB NO)	parameters
101000	rrrr10	mssub-lssub	msnewsb-lsnewsb
2 8			Srllll llllll llllll

mssub-lssub — Forms the number of the segment or subroutine from which the original subroutine was called

msnewsb-lsnewsb — Forms the number of the subroutine to be used instead of SUB NO at this instance

S	*Action*
1	Offset is from start of subroutine
0	Offset is from start of segment

llllll — Forms the 16-bit identifier of the instance

A different subroutine is substituted for SUB NO at the identified instance. No data is returned.

4.0 IMAGE CONTROL

The commands discussed in this section affect the images produced on the screen. They are grouped by their functions:

- Screen Handling
- Set Options
- Insert/Change Attributes
- Transform Parameters
- Vector
- Fill
- Text and Markers
- Pick Control
- Pixel Manipulation

Screen Handling discusses the selection of alphanumeric or graphics data from the host computer. Commands for screen handling include repaint, erase/repaint, overlay, and erase-protect.

Set Options includes commands for specifying dashed lines.

Insert/Change Attributes includes commands for determining the visual attributes of the lines used to produce images. They are the commands for dash enable, dash pattern, color, visibility, and write protect mask. There are two ways attributes can take on new values. Values can be placed in a segment header by a change attribute command or placed in the executable display list by an insert attribute command.

Transform Parameter commands specify how a segment is to be scaled, translated, or rotated. They are the insert/change parameter commands covering origin, clipping, translation, and transformation.

Vector includes commands that permit specification of moves and draws in a variety of ways. They are the move/draw vector commands covering vectors (absolute, relative, and mixed) and vector strings (short relative and incremental).

Fill discusses the commands that control the shading of polygons and graphs. They are the commands for polygon fill, baseline fill, rectangle fill, and end fill.

Text and Markers discusses the commands that provide annotation for plots and other graphic outputs. They include the commands for character string, marker selection, and marker position.

Pick Control discusses the commands that control the image when picking is done. They are the commands for change detectability, insert pick label, and detectability.

Pixel Manipulation discusses the commands that allow manipulation of individually addressed pixels. They are the commands for pixel write mode, direct pixel addressing parameters, and write/read direct pixel data.

4.1 Screen Handling

The screen of the Whizzard 1645 displays the processed result of the graphics data sent by the host computer. When the selected picture is displayed, increments of the picture can be displayed as the data is being received and is processed, or the entire display list can be received and then processed and displayed.

In interactive situations, or in other conditions when a change in the data is to be displayed, the screen must be repainted to show the change. There are commands that select whether the entire screen, only a portion of the screen, or even a single segment is to be repainted.

Since the Whizzard 1645 combines both alphanumeric and graphic terminal capabilities, the display of data can be controlled by screen handling commands (Table 4-1) from the host computer. These commands are described in detail in the paragraphs following the table.

Table 4-1 Screen Handling Commands

Binary	Hex	Total Bytes	Command
111110 000001	3E 01	3	Set Erase-Protect
111110 011000	3E 18	3	Repaint Entire Screen
111110 011001	3E 19	5	Erase/Repaint (segment)
111110 011001	3E 19	11	Erase/Repaint (screen area)
111110 011111	3E 1F	3	Repaint mode
111110 100000	3E 20	3	Overlay Control

4.1.1 Set Screen Repaint Mode

((REPAINT MODE))	((MODE))
111110 011111	rrrrII
3 E 1 F	

II	*Action*
00	Turns off auto repaint and incremental display
01	Turns off incremental display only
10	Turns off auto repaint only
11	Auto repaint and incremental display is on

Incremental Display

When incremental display is ON, graphics data is processed and displayed as it is received from the host computer. Repaint cannot be initiated from the keyboard if incremental display is ON unless the terminal is in alphanumeric mode. If incremental display is OFF, the display will not reflect any additions to the display until a repaint of the display is performed. Initially the incremental display is ON.

Auto Repaint

If a segment is being rewritten when auto-repaint in ON, the screen will be repainted after every complete command is received. If the display list is altered frequently, the repaints of the screen will produce a winking effect that may not be desirable. If auto repaint is turned OFF, the commands will not force a repaint of the screen. Any changes made will not be reflected on the screen. Once auto repaint is turned OFF, changes to the display list will not cause an update of the screen. If it is necessary to clear a viewport or segment without causing a repaint of the entire screen, it may be done using a *Set Selective Erase/Repaint* command. Initially the auto-repaint is ON.

NOTE

Once incremental display or auto-repaint of the display list has been turned OFF, they should not be turned ON until the display list and all changes made to it are complete. Unpredictable results may occur.

4.1.2 Set Selective Erase/Repaint

((ERASE/REPAINT))	(MODE)	(DATA)
111110 011001	EMRrrF	DATA
3 E 1 9		

E *Erase Select*
0 No erase prior to repaint
1 Erase prior to repaint using format given by M bit

M *Erase Mode*
0 Erase area selected by F bit
1 Erase segment selected by F bit

R *Repaint Select*
0 No repaint is desired
1 Repaint area or segment selected by F bit

F *Format of Data*
0 Data is viewport boundaries
1 Data is segment number

Data Depending on what has been selected by F.

Viewport

When the data format selected by the F bit is area, the data consists of eight bytes which specify the left, bottom, right and top boundaries. Each boundary is specified by two bytes:

X-left-msb, X-left-lsb, Y-bottom-msb, Y-bottom lsb,
X-right-msb, X-right-lsb, Y-top-msb, Y-top-lsb.

Segment

When the data format selected by the F bit is a segment, the data consists of two bytes which specify the segment number:

seg-no-msb, seg-no-lsb

The lower six bits of each byte combine to specify the 12-bit segment number.

4.1.3 Set Overlay Control

((OVERLAY CONTROL))	(SELECT)
111110 100000	rraAgG
3 E 2 0	

a *Alphanumeric Change Control*
0 No change
1 Change setting to value specified by A bit

A *Alphanumeric Overlay*
0 Off
1 On

g *Graphics Change Control*
0 No change
1 Change setting to value specified by G bit

G *Graphics Overlay*
0 Off
1 On

This command provides the same function as the second function keys on the keyboard.

4.1.4 Set Erase-Protect Mask

((SET ERASE PROT))	(MASK)
111110 000001	rrrrrI
3 E 0 1	

I *Action*
0 Disable erasure
1 Enable erasure on repaint

I Is a 1-bit inhibit mask which refers to the bit plane. If I is set to 0, the data in the associated bit plane is protected against erasure. When I is set to 1, the data in the associated bit-plane may be erased when a screen repaint occurs.

4.1.5 Repaint Entire Screen

((REPAINT))
111110 011000
3 E 1 8

This command provides the ability to repaint the entire screen without any other options.

4.2 Set Options

Set Options commands (Table 4-2) control options which are not attributes:

- Dash Size
- Color Background

Table 4-2 Set Options Commands

Binary	Hex	Total Bytes	Command
111110 011010	3E 1A	3	Set Background Color
111110 011101	3E 1D	3	Set Dash Mode

4.2.1 Set Dash Mode

((SET DASH MODE))	(SIZE)
111110 011101	rrMDrr
3 E 1 D	

M	*Action*
0	Reset dash pattern on move only
1	Reset dash pattern on move or draw

D	*Action*
0	4 pixels per bit in dash pattern
1	1 pixel per bit in dash pattern

This command controls the length of units in the dash pattern, set with the *Insert Dash Pattern* command. The dash pattern length set by this command remains in effect until changed by another *Set Dash Mode* move or draw command.

4.2.2 Set Background Color

((SET BACKGROUND COLOR))	(COLOR INDEX)
111110 011010	rrrrrI
3 E 1 A	

I	*Action*
0	Black
1	Green

Sets the background color associated with a specific color index of 0 or 1. The background set by this command remains in effect until changed by another *Set Background Color* command or through Set-Up.

4.3 Insert and Change Attributes

In addition to the segment headers created when segments are opened and defined by the host computer, a "default header" is always defined. It is the header for the non-retained segment 0. When no retained segment is open, graphics data sent to the Whizzard 1645 is placed in segment 0. Insert attributes placed in segment 0 change the corresponding attributes in the current header. When a retained segment is opened, the attributes of the default header are copied into the header of the newly opened segment to serve as current values. The commands controlling transformations are grouped together in the header.

There are two classes of basic attribute commands (Table 4-3). One class of commands places an attribute definition in a segment header (Change Attribute Commands). The other class places an attribute definition in the executable display list (Insert Attribute Commands).

Table 4-3 Attribute Commands

Binary	Hex	Total Bytes	Command
001000 00XXXX	08 0X	4	Change Segment Color
001000 10100X	08 2X	4	Change Segment Dash Enable
001000 11100X	08 3X	4	Change Segment Visibility
001001 10XXXX	09 2X	6	Change Segment Dash Pattern
001011 00XXXX	0B 0X	2	Insert Color
001011 10100X	0B 2X	2	Insert Dash Enable
001100 10XXXX	0C 2X	4	Insert Dash Pattern
001100 11XXXX	0C 3X	4	Set Write-Protect Mask

An attribute defined in a segment header affects all vectors and text which follow in the segment until a redefinition of the attribute is encountered. These header attributes can be redefined at any time before the segment is painted to produce an image.

An attribute defined in the executable display list cannot be changed without rewriting the segment in which it occurs. These Insert attribute definitions also affect all vectors and all text which follow the insertion until a redefinition of the attribute is encountered. An Insert attribute in segment 0 alters the corresponding attribute in the default header.

Most header attributes (Appendix C) can be either Changed or Inserted. The Visibility and Detectability attributes apply only to an entire segment and can only be modified with a *Change* command.

4.3.1 Change Segment Color

(CHANGE3)	(SEG NO)	(COLOR)
001000	msseg-lsseg	00rrrl
0 8		

msseg-lsseg Specifies the 12-bit segment number

l	*Action*
0	Black
1	Green

The index setting in the header of segment number (SEG NO) is changed.

Everything drawn on the screen is drawn in the color selected until a new color definition is specified by the *Insert Color* command. The *Change Segment Color* command changes values at a specified color index.

4.3.2 Change Segment Dash Enable

(CHANGE3)	(SEG NO)	(DASH ENABLE)(ON/OFF)
001000	msseg-lsseg	10100E
0 8		

msseg-lsseg Specifies the 12-bit segment number

E	*Action*
0	Disables dashing
1	Enables dashing

Changes the dash enable setting in the header of segment number (SEG NO). The (ON/OFF) turns dashing on or off so that all graphics information drawn in the segment is dashed or undashed until a new dash enable setting is defined in the segment. The *Insert Dash Enable* command places such a definition in segments.

4.3.3 Change Segment Visibility

(CHANGE3)	(SEG NO)	(VISIB)(ON/OFF)
001000	msseg-lsseg	11100E
0 8		

msseg-lsseg Specifies the 12-bit segment number

E	*Action*
0	Makes segment invisible
1	Makes segment visible

Changes the visibility of segment number (SEG NO). The (ON/OFF) field makes the segment visible or not visible. This definition determines the visibility of all graphics information drawn in the segment and it cannot be overridden within the segment by any command.

The active segment is not affected.

4.3.4 Change Segment Dash Pattern

(CHANGE5)	(SEG NO)	(DASH PAT)	(PATTERN)
001001	msseg-lsseg	10PPPP	PPPPPP PPPPPP
0 9			

msseg-lsseg Specifies the 12-bit segment number

PPP...PPP Specifies the 16-bit dash pattern

(PATTERN) is a 16-bit field which is loaded into the dash pattern register in the vector generator. The 16-bit field is mapped into the dash pattern with the bits being set to 1 for beam on, and 0 for beam off.

The dash pattern defined in the segment header affects all graphics information drawn in the segment until a new dash pattern setting is loaded into the vector generator or until dash is turned off. The *Insert Dash Enable* and *Insert Dash Pattern* commands place such definitions in segments.

4.3.5 Insert Color

(INSERT1)	(COLOR)
001011	00rrrL
0 B	

L	*Action*
0	Black
1	Green

A new definition of color is inserted in the active segment at the insert pointer. Everything that follows in the segment will be drawn in this color until a new definition is loaded into this register, until another *Insert Color* command, or the processing of a new segment header.

4.3.6 Insert Dash Enable

(INSERT1)	(DASH ENABLE) (ON/OFF)
001011	10100E
0 B	

E	*Action*
0	Disables dashing
1	Enables dashing

Inserts a new dash control definition in the active segment at insert pointer. The (ON/OFF) turns dash on or off, so that all graphics which follow in the segment are drawn dashed or undashed until a new dash control definition is placed in the segment by another *Insert Dash Enable* command.

4.3.7 Insert Dash Pattern

The dashed fill pattern for filling a polygon will follow the contour of the polygon or the clip boundary in effect at the time of the fill.

(INSERT3)	(DASH)	(PATTERN)
001100	10PPPP	PPPPPP PPPPPP
0 C		

PPP...PPP Specifies the 16-bit dash pattern

(PATTERN) is a 16-bit field which is loaded into the dash pattern register in the vector generator. The 16-bit field is actually mapped into the dash pattern, with each bit set to 1 for beam on, and 0 for beam off. The length of the on and off portions of the pattern is set by the *Set Dash Mode* command.

NOTE

Any irregularity in the boundary will cause the same irregularities in the dash pattern itself.

4.3.8 Set Write-Protect Mask

(INSERT3)	(PROTECT)	(MASK)
001100	11rrrr rrrrrr	rrrrrl
0 C		

l Is a 1-bit protect mask. Write protection of the data in the bit-plane is set to 0 to disable. Write protection of the data in the bit-plane is set to 1 to enable.

This command protects the raster plane against the addition of more video display data.

4.4 Transformation Parameters

Whizzard 1645 systems can translate the position of the image, scale the image up or down, and rotate it. The commands (Table 4-4) in this section are used when such transformations are to be performed. Appendix B summarizes the matrix mathematics used to produce transformation parameters.

Table 4-4 Transformation Parameters

Binary	Hex	Total Bytes	Command
001000 11000X	08 3X	4	Change Segment Transformation Enable
001010 000000	0A 00	8	Change 2D Segment Origin
001010 000001	0A 01	10	Change 3D Segment Origin
001010 000010	0A 02	6	Change 3D Z Segment Origin
001010 000011	0A 03	10	Change Segment Translation
001010 000100	0A 04	22	Change 2D Segment Matrix
001010 000101	0A 05	28	Change 3D Segment Matrix
001010 000110	0A 06	12	Change Segment Clip Boundaries
001011 11000X	08 3X	2	Insert Transformation Enable
001101 000000	0D 00	6	Insert 2D Segment Origin
001101 000001	0D 01	8	Insert 3D Segment Origin
001101 000010	0D 02	4	Insert 3D Z Segment Origin
001101 000011	0D 03	8	Insert 2D Translation
001101 000100	0D 04	8	Insert 2D Matrix
001101 000101	0D 05	26	Insert 3D Matrix
001101 000110	0D 06	10	Insert Clip Boundaries

4.4.1 Change Segment Transformation Enable

(CHANGE3)	(SEG NO)	(XFORM ENABLE)(ON/OFF)
001000	msseg-lsseg	11000E
0 8		

msseg-lsseg Specifies the 12-bit segment number

E	*Action*
0	Transformation is disabled
1	Transformation is enabled

In the header of a transform segment, this command changes the transformation enable setting of segment (SEG NO).

Changes the transformation enable setting in the header of transform segment number (SEG NO). The (ON/OFF) field turns the transformation hardware on or off. The setting of E transforms everything in the segment according to the matrix elements defined in the segment heading. This setting applies until a new transformation enable is defined in the segment by an *Insert Transformation Enable* command.

4.4.2.1 Change 2D Segment Origin

Whizzard 1645 graphic systems are equipped with hardware translation circuitry for the *Change 2D Segment Origin* commands. This circuitry can move a series of vectors or an entire picture on the display screen with a minimum amount of I/O traffic between the host computer and the system.

(CHANGE XFORM)	(SEG NO)	(ORIGIN)	(X DATA)	(Y DATA)
001010	msseg-lsseg	000000	msx-lsx	msy-lsy
0 A				

msseg-lsseg — Specifies the 12-bit segment number

msx-lsx — The 6-bit values msx and lsx form the 12-bit X origin value in screen units

msy-lsy — The 6-bit values msy and lsy form the 12-bit Y origin value in screen units

Translates the screen origin to the specified coordinates. The origin values must lie in the range of addressable screen coordinates, -2048 to +2047, and are expressed as two's complement the 12-bit numbers. The default screen origin (0,0) is at screen center. The X and Y coordinates of any absolute move or draw are added to the specified origin by the graphics processor. All graphics that appear after this command will use the new origin until the origin is changed by another *Change 2D Segment Origin* command or an *Insert 2D Segment Origin* command.

Wrap-around occurs if the addition of absolute X and Y coordinates to the origin cause it to fall outside the range of addressable screen coordinates.

4.4.2.2 Change 3D Segment Origin

Whizzard 1645 graphic systems are equipped with hardware translation circuitry for the *Change 3D Segment Origin* commands. This circuitry can move a series of vectors or an entire picture on the display screen with a minimum amount of I/O traffic between the host computer and the system.

(CHANGE XFORM)	(SEG NO)	(ORIGIN)	(X DATA)	(Y DATA)	(Z DATA)
001010	msseg-lsseg	000001	msx-lsx	msy-lsy	msz-lsz
0 A					

msseg-lsseg — Specifies the 12-bit segment number

msx-lsx — The 6-bit values msx and lsx form the 12-bit X origin value in screen units

msy-lsy — The 6-bit values msy and lsy form the 12-bit Y origin value in screen units

msz-lsz — The 6-bit values msz and lsz form the 12-bit Z origin value in screen units

Translates the screen origin to the specified coordinates. The origin values must lie in the range of addressable screen coordinates, -2048 to +2047, and are expressed as two's complement 12-bit numbers. The defaults screen origin (0,0,0) is at screen center. The X, Y, and Z coordinates of any absolute move or draw are added to the specified origin by the graphics processor. All absolute graphics commands that appear after this command will use the new origin until the origin is changed by another *Change 3D Segment Origin* command or an *Insert 3D Segment Origin* command.

Wrap-around occurs if the addition of absolute X and Y coordinates to the origin causes it to fall outside the range of addressable screen coordinates.

4.4.2.3 Change 3D Z Segment Origin

The Whizzard 1645 *Change 3D Z Segment Origin* command uses the previously defined X and Y coordinates along with the newly defined Z coordinate specified by this command to form a new origin for the segment (Seg No) defined in the command.

(CHANGE XFORM)	(SEG NO)	(ORIGIN)	(Z DATA)
001010 0 A	msseg-lsseg	000010	msz-lsz

msseg-lsseg — Specifies the 12-bit segment number

msz-lsz — The 6-bit values msz and lsz form the 12-bit origin value in screen units

4.4.3 Change Segment Translation

(CHANGE XFORM)	(SEG NO)	(XLATE)	(ELEMENTS)
001010 0 A	msseg-lsseg	000011	M2 M5

msseg-lsseg — Specifies the 12-bit segment number

M2 and M5 — 3-character groups which encode the 16-bit values of the corresponding matrix elements

M2 is a group of three characters which encodes the 16-bit value of matrix element m(3,1) of the transformation matrix.

M5 is a group of three characters which encode the 16-bit value of matrix element m(3,2) of the transformation matrix. Refer to Appendix B for explanation of transformation matrix.

4.4.4.1 Change 2D Segment Matrix

(CHANGE XFORM)	(SEG NO)	(MATRIX)	(ELEMENTS)
001010	msseg-lsseg	000100	(M0) (M1) (M2)
0 A			(M3) (M4) (M5)

msseg-lsseg — Specifies the 12-bit segment number

M0 through M5 — Are three character groups which encode the 16-bit values of the corresponding matrix elements

Bits 1-12 of transformation matrix elements 2 and 5 are used to specify the X and Y pixel offset from the origin. Refer to Appendix B for scale factors on the X and Y.

4.4.4.2 Change 3D Segment Matrix

(CHANGE XFORM)	(SEG NO)	(MATRIX)	(ELEMENTS)
001010	msseg-lsseg	000101	(M0) (M1) (M2) (M3)
0 A			(M4) (M5) (M6) (M7)

msseg-lsseg — Specifies the 12-bit segment number

M0 through M7 — Are three character groups which encode the 16-bit values of the corresponding matrix elements

Bits 1-12 of transformation matrix elements 2 and 5 are used to specify the X and Y pixel offset from the origin. Refer to Appendix B for scale factors on the X and Y.

4.4.5 Change Segment Clip Boundaries

Whizzard 1645 systems can clip transformed images. Clipping can be used to create multiple viewports and to prevent wrap-around.

(CHANGE XFORM)	(SEG NO)	(CLIP)	(LO X) (LO Y) (HI X) (HI Y)
001010 0 A	msseg-lsseg	000110	msxl-lsxl msyl-lsyl msxh-lsxh msyh-lsyh

msseg-lsseg	Specifies the 12-bit segment number
msxl-lsxl	The 6-bit values msxl and lsxl form the 12-bit lower X clipping value
msyl-lsyl	The 6-bit values msyl and lsyl form the 12-bit lower Y clipping value
msxh-lsxh	The 6-bit values msxh and lsxh form the 12-bt upper X clipping value
msyh-lsyh	The 6-bit values msyh and lsyh form the 12-bit upper Y clipping value

The *Change Clip Boundaries* command changes the clip boundaries. (HI X) and (HI Y) are the coordinates of the upper right corner of the rectangular clip window of the segment. (LO X) and (LO Y) are the X and Y coordinates of its lower left corner. All coordinates must lie in the range from -2048 to +2047, and are expressed as two's complement 12-bit numbers.

4.4.6 Insert Transformation Enable

(INSERT1)	(XFORM ENABLE)	(ON/OFF)
001011 0 B	11000E	

E	*Action*
0	Disables transformation
1	Enables transformation

These commands insert new definitions of the transformation matrix elements, the clipping boundaries, and the origin. Each new definition placed in the segment holds for all graphics which follow in the segment, until a new definition is placed there.

A new transformation enable definition is inserted in the active segment at the insert pointer. The (ON/OFF) field turns the transformation hardware on or off. All graphics which follow in the segment are transformed or left untransformed depending on the enable setting. This definition is in effect until a new enable definition is placed in the segment by another *Insert Transformation Enable* command.

4.4.7.1 Insert 2D Segment Origin

Whizzard 1645 systems feature hardware translation circuitry as standard equipment. This circuitry can move a series of vectors or an entire picture on the display screen with a minimum amount of I/O traffic between the host computer and the Whizzard 1645.

(INSERT XFORM)	(ORIGIN)	(X DATA)	(Y DATA)
001101	000000	msx-lsx	msy-lsy
0 D			

msx-lsx The 6-bit values msx and lsx form the 12-bit X origin value in screen units

msy-lsy The 6-bit values msy and lsy form the 12-bit Y origin value in screen units

The *Insert 2D Segment Origin* command translates the screen origin to the specified coordinates. The origin values must lie in the range of addressable screen coordinates, -2048 to +2047, and must be expressed as two's complement 12-bit numbers. The default screen origin (0,0) is at screen center. The X and Y coordinates of any absolute move or draw are added to the specified origin by the graphics processor.

Wrap-around occurs if the addition of absolute X and Y coordinates to the origin causes a point to fall outside the range of addressable screen coordinates. All absolute graphics commands appearing after this command will use the new origin until the origin is reset by another *Insert 2D Segment Origin* command.

4.4.7.2 Insert 3D Segment Origin

(INSERT XFORM)	(ORIGIN)	(X DATA)	(Y DATA)	(Z DATA)
001101	000001	msx-lsx	msy-lsy	msz-lsz
0 D				

msx-lsx The 6-bit values msx and lsx form the 12-bit X origin value in screen units

msy-lsy The 6-bit values msy and lsy form the 12-bit Y origin value in screen units

msz-lsz The 6-bit values msz and lsz form the 12-bit Z origin value in screen units

The *Insert 3D Segment Origin* command translates the screen origin to the specified coordinates. The origin values must lie in the range of addressable screen coordinates, -2048 to +2047, and must be expressed as two's complement 12-bit numbers. The default screen origin (0,0,0) is at screen center. The X, Y, and Z coordinates of any absolute move or draw are added to the specified origin by the graphics processor.

Wrap-around occurs if the addition of absolute X, Y, and Z coordinates to the origin causes a point to fall outside the range of addressable screen coordinates. All absolute graphics commands appearing after this command will use the new origin until the origin is reset by another *Insert 3D Segment Origin* command.

4.4.7.3 Insert 3D Z Segment Origin

(INSERT XFORM)	(ORIGIN)	(Z DATA)
001101	000010	msz-lsz
0 D		

msz-lsz The 6-bit values msz and lsz form the 12-bit Z origin value in screen units

The *Insert 3D Z Segment Origin* command places an instruction into the current segment at the insert pointer to change the Z coordinate specification of the origin. The origin values must lie in the range of addressable screen coordinates, -2048 to +2047, and must be expressed as two's complement 12-bit numbers.

4.4.8 Insert Translation

(INSERT XFORM)	(XLATE)	(ELEMENTS)	
001101	000011	M2	M5
0 D			

M2 and M5 — Are three character groups encoding the 16-bit values of the corresponding matrix elements.

Refer to Appendix B for transformation matrix.

4.4.9.1 Insert 2D Matrix

(INSERT XFORM)	(MATRIX)	(ELEMENTS)
001101	000100	(M0)(M1)(M2)(M3)(M4)(M5)
0 D		

M0 through M5 — Are three character groups encoding 16-bit values for corresponding elements of the matrix

The *Insert 2D Matrix* command places the instruction into the current segment at the insert pointer to allow the six 2D transformation matrix elements to be modified to the values given in the command.

Refer to Appendix B for transformation matrix.

4.4.9.2 Insert 3D Matrix

(INSERT XFORM)	(MATRIX)	(ELEMENTS)
001101	000101	(M0)(M1)(M2)(M3)(M4)(M5)(M6)(M7)
0 D		

M0 through M7 — Are three character groups encoding 16-bit values for corresponding elements of the matrix.

The *Insert 3D Matrix* command places the instruction into the current segment at the insert pointer to allow the eight 3D transformation matrix elements to be modified to the values given in the command.

Refer to Appendix B for transformation matrix.

4.4.10 Insert Clip Boundaries

(INSERT XFORM)	(CLIP)	(LO X) (LO Y) (HI X) (HI Y)
001101	000110	msxl-lsxl
0 D		msyl-lsyl
		msxh-lsxh
		msyh-lsyh

msxl-lsxl The 6-bit values msxl and lsxl form the 12-bit lower X clipping value

msyl-lsyl The 6-bit values msyl and lsyl form the 12-bit lower Y clipping value

msxh-lsxh The 6-bit values msxh and lsxh form the 12-bit upper X clipping value

msyh-lsyh The 6-bit values msyh and lsyh form the 12-bit upper Y clipping value

The *Insert Clip Boundaries* command inserts clip boundary definitions in the display list. (HI X) and (HI Y) are the coordinates of the upper right corner of the rectangular clip window of the segment. (LO X) and (LO Y) are the X and Y coordinates of its lower left corner. All coordinates must lie in the range from -2048 and +2047 and are expressed as two's complement 12-bit numbers.

4.5 Vector Commands

Vector commands (Table 4-5) enter moves and draws into the active segment. They place specified graphics data into the active segment at the insert pointer and update the insert pointer to the next available location. Vector commands do not change which segment is active.

The appearance of vectors drawn on the screen is affected by the attribute settings of blink, dash, and color. For all segments, except non-retained segment 0, the attributes set in the segment header are applied to the vectors placed in the segment, until a setting is changed by an *Insert Attribute* command which overrides the header setting. In default segment 0, the attribute settings from the default header are applied to the vectors until another *Insert Attribute* command places an overriding setting in the default header Vector commands have no effect on attribute settings.

Table 4-5 Vector Commands

Binary	Hex	Total Bytes	Command
001111	OF	8	Move/Draw Mixed Vector (2D /3D)
010000	10	5	Move 2D Absolute Vector
010001	11	5	Draw 2D Absolute Vector
010010	12	5	Move 2D Relative Vector
010011	13	5	Draw 2D Relative Vector
010100	14	7	Move 3D Absolute Vector
010101	15	7	Draw 3D Absolute Vector
010110	16	7	Move 3D Relative Vector
010111	17	7	Draw 3D Relative Vector
011001	19	3 + 3N	Move/Draw 2D Short Relative Vector String
011010	1A	3 + 4N	Move/Draw 3D Short Relative Vector String
011011	1B	3 + 3N	Move/Draw Incremental Vector String

4.5.1.1 Move/Draw 2D Mixed Vector

(MIX VEC)	(COORD TYPE)	(X DATA)	(Y DATA)	(RESERVED)
001111	00DXYr	msx-lsx	msy-lsy	
0 F				

D	*Action*
0	Move
1	Draw

X, Y	*Coordinate Type*
0	Absolute
1	Relative

msx-lsx — The 6-bit values msx-lsx form the 12-bit X coordinate

msy-lsy — The 6-bit values msy-lsy form the 12-bit Y coordinate

This command allows a move or draw with the X, Y point specified in mixed relative and absolute coordinates (X and y are integers in the range -2048 to +2047). The move action D = 0 causes the current position to be changed to the new coordinates. This action does not produce a line on the screen. The draw action D = 1 draws a line from the current position to the coordinates specified.

4.5.1.2 Move/Draw 3D Mixed Vector

(MIX VEC)	(COORD TYPE)	(X DATA)	(Y DATA)	(Z DATA)
001111	10DXYZ	msx-lsx	msy-lsy	msz-lsz
0 F				

D	*Action*
0	Move
1	Draw

X, Y, Z	*Coordinate Type*
0	Absolute
1	Relative

msx-lsx — The 6-bit values msx-lsx form the 12-bit X coordinate

msy-lsy — The 6-bit values msy-lsy form the 12-bit Y coordinate

msz-lsz — The 6-bit values msz-lsz form the 12-bit Z coordinate

This command allows a move or draw with the X, Y, Z point specified in mixed relative and absolute coordinates (X, Y, and Z are integers in the range -2048 to +2047). The move action D = 0 causes the current position to be changed to the new coordinates. This action does not produce a line on the screen. The draw action D = 1 draws a line from the current position to the coordinates specified.

4.5.2.1 Move 2D Absolute Vector/ Draw 2D Absolute Vector

(MOVE ABS)	(X DATA)	(Y DATA)
010000	msx-lsx	msy-lsy
1 0		

(DRAW ABS)	(X DATA)	(Y DATA)
010001	msx-lsx	msy-lsy
1 1		

msx-lsx Form the absolute X coordinate

msy-lsy Form the absolute Y coordinate

The *Move 2D Absolute Vector* places an absolute move (no vector appears on the screen) and the *Draw 2D Absolute Vector* command places an absolute draw (the vector does appear on the screen) in the active segment at the insert pointer and updates the insert pointer to the next available location. The absolute coordinates of (X DATA) and (Y DATA) must be in the range from -2048 to +2047, and are expressed as two's complement 12-bit numbers.

4.5.2.2 Move 3D Absolute Vector/ Draw 3D Absolute Vector

(MOVE ABS)	(X DATA)	(Y DATA)	(Z DATA)
010100	msx-lsx	msy-lsy	msz-lsz
1 4			

(DRAW ABS)	(X DATA)	(Y DATA)	(Z DATA)
010101	msx-lsx	msy-lsy	msz-lsz
1 5			

msx-lsx Form the absolute X coordinate

msy-lsy Form the absolute Y coordinate

msz-lsz Form the absolute Z coordinate

The *Move 3D Absolute Vector* places an absolute move (no vector appears on the screen) and the *Draw 3D Absolute Vector* command places an absolute draw (the vector does appear on the screen) in the active segment at the insert pointer and updates the insert pointer to the next available location. The absolute coordinates of (X DATA), (Y DATA), and (Z DATA) must be in the range from -2048 to +2047, and are expressed as two's complement 12-bit numbers.

4.5.3.1 Move 2D Relative Vector/Draw 2D Relative Vector

(MOVE REL)	(X DATA)	(Y DATA)
010010	msx-lsx	msy-lsy
1 2		

(DRAW REL)	(X DATA)	(Y DATA)
010011	msx-lsx	msy-lsy
1 3		

msx-lsx Form the relative X coordinate

msy-lsy Form the relative Y coordinate

The *Move 2D Relative Vector* command and the *Draw 2D Relative Vector* command place a relative move or a relative draw in the active segment at the insert pointer. They update the insert pointer to the next available location. The displacement values in (X DATA) and (Y DATA) must by in the range from -2048 to +2047 and are expressed as two's complement 12-bit numbers.

4.5.3.2 Move 3D Relative Vector/Draw 3D Relative Vector

(MOVE REL)	(X DATA)	(Y DATA)	(Z DATA)
010110	msx-lsx	msy-lsy	msz-lsz
1 6			

(DRAW REL)	(X DATA)	(Y DATA)	(Z DATA)
010111	msx-lsx	msy-lsy	msz-lsz
1 7			

msx-lsx Form the relative X coordinate

msy-lsy Form the relative Y coordinate

msz-lsz Form the relative Z coordinate

The *Move 3D Relative Vector* command and the *Draw 3D Relative Vector* command place a relative move or a relative draw in the active segment at insert pointer. They update the insert pointer to the next available location. The displacement values in (X DATA), (Y DATA), and (Z DATA) must by in the range from -2048 to +2047 and are expressed as two's complement 12-bit numbers.

4.5.4.1 Move/Draw 2D Short Relative Vector String

(SHORT REL)	(SIZE)	(MOVE/DRAW)	(VECTOR) ..... (VECTOR) (TERM)
011001	000SSS	rrDXY0	XXXXXX YYYYYY ....
1 9		rrDXY0	XXXXXX YYYYYY GS

SSS — Is the 3-bit size descriptor. Each vector is multiplied by (SSS + 1)

D	*Action*
0	Relative move
1	Relative draw

X — Is the most significant bit of the 7-bit relative X component

Y — Is the most significant bit of the 7-bit relative Y component

XXXXXX — Are the six least significant bits of the 7-bit relative X component

YYYYYY — Are the six least significant bits of the 7-bit relative Y component

rrDXY0 - XXXXXX - YYYYYY repeats for each vector

GS — Is the terminator

This command places a short relative vector string in the active segment at the insert pointer and updates the insert pointer to the next available location. Relative offsets are 7-bit two's complement numbers in the range -64 to + 63.

The (SIZE) field is specified as a value ranging from 0 to 7, which represents a scale factor from 1 to 8 applied to the vectors, allowing eight different vector sizes.

The Whizzard 1645 will accept short relative vectors until the (TERM) terminator is received.

4.5.4.2 Move/Draw 3D Short Relative Vector String

(SHORT REL)	(SIZE)	(MOVE/DRAW)	(VECTOR) ..... (VECTOR) (TERM)
011010	000SSS	rrDXYZ	XXXXXX YYYYYY ZZZZZZ ....
1 A		rrDXYZ	XXXXXX YYYYYY ZZZZZZ GS

SSS	Is the 3-bit size descriptor. Each vector is multiplied by (SSS + 1)
D	***Action***
0	Relative move
1	Relative draw
X	Is the most significant bit of the 7-bit relative X component
Y	Is the most significant bit of the 7-bit relative Y component
Z	Is the most significant bit of the 7-bit relative Z component
XXXXXX	Are the six least significant bits of the 7-bit relative X component
YYYYYY	Are the six least significant bits of the 7-bit relative Y component
ZZZZZZ	Are the six least significant bits of the 7-bit relative Z component

rrDXYZ - XXXXXX - YYYYYY - ZZZZZZ repeats for each vector

GS	Is the terminator

This command places a short relative vector string in the active segment at the insert pointer and updates the insert pointer to the next available location. Relative offsets are 7-bit two's complement numbers in the range -64 to + 63.

The (SIZE) field is specified as a value ranging from 0 to 7, which represents a scale factor from 1 to 8 applied to the vectors, allowing eight different vector sizes.

The Whizzard 1645 will accept short relative vectors until the (TERM) terminator is received.

4.5.5 Move/Draw Incremental Vector String

(STRING)	(PLOT AS)	(DELTA X)	(M/D)	(Y COORD)		
			....(M/D)	(Y COORD)	(TERM)	
011011	rrrrrP	msx-lsx	rrrrrD	msy-lsy		
1 B			....rrrrrD	msy-lsy	GS	

P	*String plotted as*
0	Vectors
1	Points

msx-lsx 12-bit relative X displacement

D	*Action*
0	Move
1	Draw

msy-lsy 12-bit absolute Y coordinates

The sequence (rrrrD msy-lsy) repeats for each vector

GS Terminator

The X coordinate is incremented regularly (as for periods of time) while the Y coordinate varies to represent any fluctuating value (sales, temperature, etc). The points can be shown as unconnected points or as connected points.

4.6 Fill Commands

This section covers the command (Table 4-6) used for generating filled polygons (polygon fill, baseline fill). The number of allowed points in the polygon are a function of memory. In addition, there is a special command for rectangle polygons. The maximum number of points in a polygon is 600.

The options associated with polygon fill and rectangle are:

- Fill color
- Horizontal or vertical fill
- Dot and dash fill patterns
- Fill spacing
- Defining outline after the polygon has been filled.

The options associated with baseline fill are:

- Fill color above and below baseline
- Dot and dash patterns
- Fill density
- Optional output of the defining outline

The *End Fill* command is used to exit the fill function.

Table 4-6 Fill Commands

Binary	Hex	Total Bytes	Command
111110 001000	3E 08	6	Set Polygon Fill
111110 010100	3E 14	10	Set Baseline Fill
111110 001110	3E 0E	11 or 14	Set Rectangle Fill
111110 001001	3E 09	2	Exit Polygon Fill

4.6.1 Set Polygon Fill

The Set Polygon Fill command defines a polygon which is filled according to the specified parameter.

((POLY FILL))	(ATTRIBUTES)
111110 001000	rHOrDF rrcolr rrmssp-rrlssp
3 E 0 8	

H *Back-face testing*
0 Off
1 On

O *Outline*
0 No outline after fill
1 Outline after fill

D *Dash*
0 Disables dash
1 Enables dash

F *Fill direction*
0 Horizontal fill
1 Vertical fill

colr 1-bit fill color of polygon

mssp-lssp 8-bit spacing of fill lines. Must be a power of two

The direction of the fill determines the baseline used to define the object. If the direction of the fill is vertical, the baseline is horizontal. All moves and draws between the *Set Polygon Fill* command and the *Exit Polygon Fill* command define a polygon which is filled according to the specified parameters.

4.6.2 Set Baseline Fill

The *Set Baseline Fill* command defines a polygon which is filled to the established baseline according to the specified parameters.

((BASE FILL))	(COLORUP)	(COLORDN)	(ATTRIBUTES)
111110 010100	rrPCOL	rrSCOL	rrmssp-rrlssp
3 E 1 4			msbase-lsbase
			rrOrDF rrrrdr

PCOL — Primary, 1-bit fill color of area immediately above or to the left of baseline

SCOL — Secondary, 1-bit color of area immediately below or to the right of baseline

mssp-lssp — 8-bit spacing of fill lines.

msbase-lsbase — 12-bit baseline

O *Outline*
0 No outline after fill
1 Outline after fill

D *Primary Dash*
0 Disables
1 Enables

F *Primary Fill Direction*
0 Horizontal
1 Vertical

d *Secondary Dash*
0 Disables
1 Enables

All moves and draws between the *Set Baseline Fill* command and the *Exit Polygon Fill* command define a polygon which is filled according to the specified parameters.

4.6.3 Set Rectangle Fill

((RECTANGLE FILL))	(ATTRIBUTES)
111110 001110 3 E 0 E	NrOrDF mleftx-lleftx mmaxy-lmaxy mrihtx-lrihtx mminy-lminy [rrcolr rrmssp-rrlssp]

N	*Action*
0	Do not fill retangle
1	Fill retangle

O	*Outline*
0	No outline
1	Draw outline

D	*Dash*
0	No dash
1	Dash ON

F	*Fill direction*
0	Horizontal fill
1	Vertical fill

mleft-lleftx Form the absolute X coordinate mrihtx-lrihtx

mmaxy-lmaxy Form the absolute Y coordinate mminy-lminy

[rrcolr rrmssp-rrlssp] is only used when the value of N is 1 for filling the rectangle.

colr 4-bit fill color of rectangle

mssp-lssp 8-bit spacing of fill lines, must be a power of two

Allows drawing a rectangle by specifying the two pairs of X, Y coordinates. The rectangle may be filled by making N = 1. If N = 0, then the outline bit **(O)** will determine if the perimeter of the rectangle is displayed on the screen. When outline is requested, the outline is drawn with the current attributes. An *Exit Polygon* command is not required when doing a rectangle fill.

4.6.4 Exit Polygon Fill

((END FILL))	
111110	001001
3 E	0 9

Closes the polygon or baseline definition and fills the resultant area.

4.7 Text and Markers

The appearance of characters on the screen is affected by the attribute settings of blink and color. The character commands have no effect on attribute settings nor do they change which segment is active.

The character commands (Table 4-7) enter character strings into segments. A character command places the specified graphics characters into the active segment at the insert pointer and updates the insert pointer to the next available location. The actual number of words written into vector memory by each command is determined by the length of the string preceding the GS character.

Table 4-7 Text and Marker Commands

Binary	Hex	Total Bytes	Command
011000	18	4 + N	Character String
111110	3E	4	Marker Selection
011100	1C	6	Marker Position (2D/3D)

4.7.1 Character String

(CHAR STRING)	(SIZE)	(ROT)	(CHAR)...(CHAR)	(TERM)
011000 1 8	0U0SSS	0000RR	char.....char	GS

U	*Rotation*
0	Characters rotated (characters will scale, rotate, and translate)
1	Characters unrotated (characters will not scale or rotate, but will translate)
SSS	Is the 3-bit size descriptor Each character is displayed at size SSS
RR	*Orientation of String*
00	0° horizontal, built in + X direction
01	90° counter-clockwise, built in + Y direction
10	180° counter-clockwise, built upside-down in X direction
11	270° counter-clockwise, built in Y direction
char	7-bit ASCII character code
GS	7-bit ASCII GS character (011101) which terminates the character string

Characters are drawn stroke by stroke on the screen as relative displacements from the current position. Each stroke in the new character represents a series of relative moves or draws. Wrap-around may occur if the characters force the current position to overflow. If the segment is transformed, characters are scaled, translated, and clipped. If the U-bit is set to 0, the characters will be rotated.

The (SIZE) field (Table 4-8) is specified as a value ranging from 0 to 7, which represents a scale factor from 1 to 8 applied to the characters, allowing eight different character sizes. The aspect ratio (height:width) of the letters is 3:2. "White space" is included around each character to provide normal spacing in character strings.

Table 4-8 Character and Marker Sizes in Screen Units

Size	Space Width	Line Height	Characters/ Line	Lines Screen/
0	12	18	341	227
1	24	36	170	113
2	36	54	113	75
3	48	72	85	56
4	60	90	68	45
5	72	108	56	37
6	84	126	48	32
7	96	144	42	28

The (ROT) field specifies the orientation of the character string, and is specified as a value from 0 to 3. All characters are drawn relative to the current position. Following the drawing of a character string on the screen, the current position is displaced from the last character, in the direction of the string, so that the first stroke of another character would begin at that position. In general, the screen position for continued graphics should be established with a move command following the placement of text.

A character string is placed in the active segment at the insert pointer and the insert pointer is updated to the next available location.

Non-control characters are accepted by the Whizzard 1645 system in exception to the rule that only the lower six bits of each character are used when in graphics mode. Control characters except GS and US are ignored. Although the GS character is usually used to switch from terminal mode to graphics mode, in this case the GS character is used to terminate the character string command sequence. As in all other cases, the system remains in graphics mode following this termination.

4.7.2 Marker Selection

((SET MARKER))	(SIZE)	(CHAR)
111110 001010	rrrSSS	CCCCCC
3 E 0 A		

SSS 3-bit size for markers (Table 4-8)

CCCCCC 7-bit ASCII character used for marker

This command is used to select the character and its color for the marker.

4.7.3 Marker 2D Position

(MARK)	(POSI)(M/D)	(X DATA)	(Y DATA)
011110	0rDXYr	msx-lsx	msy-lsy
1 C			

D	*Action*
0	Move
1	Draw

X,Y	*Action*
0	Absolute
1	Relative

msx-lsx X position for marker

msy-lsy Y position for marker

This command sets the position in which the marker will be positioned on the screen. The marker is displayed at the current marker size.

4.7.4 Marker 3D Position

(MARK)	(POSI)(M/D)	(X DATA)	(Y DATA)	(Z DATA)
011110	10DXYZ	msx-lsx	msy-lsy	msz-lsz
1 C				

D	*Action*
0	Move
1	Draw

X,Y,Z	*Action*
0	Absolute
1	Relative

msx-lsx X position for marker

msy-lsy Y position for marker

msz-lsz Z position for marker

The *3D Marker Position* command causes a marker character to be placed at the location described by a combination of absolute and relative coordinates X, Y, and Z (as indicated by flags within the command) where X, Y, and Z are integers in the range -2048 to +2047. If visibility is enabled, a vector is drawn from the current position to the position specified in the marker command.

4.8 Pick Control

The commands (Table 4-9) in this section control the image when picking is done. Refer to Section Five for control of the pick parameters.

Table 4-9 Pick Control Commands

Binary	Hex	Total Bytes	Command
001001	09	6	Change Segment Detectability
001100	0C	4	Insert Pick Label
001110	0E		Set Detectability

4.8.1 Change Segment Detectability

(CHANGE5)	(SEG NO)	(DETECT)	(ATTRIBUTES)	(LEVEL)
001001	msseg-lsseg	010000	CDrrrP	rrrLLL
0 9				

msseg-lsseg Specifies the 12-bit segment number

C	*Action*
0	No change in detectability
1	Change detectability

D	*Action*
0	No change in screen echo level
1	Change screen echo level

P	*Action (effective only if C = 1)*
0	Segment is non-detectable for picking
1	Segment is detectable for picking

LLL	*3-bit screen echo level for SEG NO*
000	Echo the entire segment
001	Echo all vectors within the labeled subsegment and below (suppress echo above)
010	Echo all vectors within the first subroutine and below (suppress echo above)
011	Echo all vectors within the second subroutine and below (suppress echo above)
100	Echo all vectors within the third subroutine and below (suppress echo above)
101	Echo all vectors within the fourth subroutine and below (suppress echo above)

Graphics objects are ignored by the pick module unless the object is established as detectable. To identify the object picked, labels are inserted in the display list. A label identifies that portion of the display list from the occurrence of the label until the end of the segment, the end of the subroutine, or the next pick label.

4.8.2 Inset Pick Label

(INSERT3)	(PICK LABEL)	(LABEL)
001100	01LLLL	LLLLLL LLLLLL
0 C		

LLL...LLL Specifies the 16-bit label

A pick label is inserted at the current insert pointer.

To identify the object picked, labels are inserted in the display list. A label identifies that portion of the display list from the occurrence of the label until the end of the segment, the end of the subroutine, or the next pick label.

4.8.3 Set Pick Detectability

(SET PICK INFO)	(ATTRIBUTES)	(LEVEL)
001110	CDrrrP	rrrLLL
0 E		

C	*Action*
0	No change in detectability
1	Change detectability

D	*Action*
0	No change in screen echo level
1	Change screen echo level

P	*Action (effective only if C = 1)*
0	Segment is non-detectable for picking
1	Segment is detectable for picking

LLL	*3-bit screen echo level for SEG NO*
000	Echo the entire segment
001	Echo all vectors within the labeled subsegment and below (suppress echo above)
010	Echo all vectors within the first subroutine and below (suppress echo above)
011	Echo all vectors within the second subroutine and below (suppress echo above)
100	Echo all vectors within the third subroutine and below (suppress echo above)
101	Echo all vectors within the fourth subroutine and below (suppress echo above)

This command is used only to change the pick attributes in the current header. All segments opened after this command will reflect the changes made. No display list instruction is generated.

4.9 Pixel Manipulation

Individual pixels are addressable in video memory and the action of pixel writing may be modified using pixel manipulation commands (Table 4-10).

Table 4-10 Pixel Manipulation Commands

Binary	Hex	Command
111110 011110	3E 1E	Set Pixel Write Mode
111110 110000	3E 30	Set Direct Pixel Addressing Parameters
111110 110001	3E 31	Write/Read Direct Pixel Data

4.9.1 Set Pixel Write Mode

((SET PIXEL MODE))	(MODE)
111110 011110	rrrrMM
3 E 1 E	

MM	*Pixel write mode*
00	Replace pixel
01	Logical OR existing pixels with overwriting pixels
10	Logical complement existing pixels with overwriting pixels
11	Reserved

4.9.2 Set Direct Pixel Addressing Parameters

((SET DPA PAR))	(PARAMETERS)
111110 110000	rrrXYr xorgms-xorgls
3 E 3 0	yorgms-yorgls
	delxms-delxls
	delyms-delyls

xorgms-xorgls 12-bit value of X-axis component of absolute origin

yorgms-yorgls 12-bit value of Y-axis component of absolute origin

delxms-delxls 12-bit value of X-axis size

delyms-delyls 12-bit value of Y-axis size

This command sets the parameters of the space in which a direct pixel write or read will occur. It must be performed before the write or read. The values are specified in virtual units in the 4096 x 4096 virtual address space.

The values specified for xorg and yorg determine an absolute location in the virtual address space and define a corner of a rectangle in which direct pixel addressing occurs.

The magnitude of the values specified by delx and dely determines the size of the area in which direct pixel addressing occurs. All values (both positive and negative) indicate left-to-right and bottom-to-top addressing. These values may be specified independently as absolute in the virtual address space or relative to the absolute origin from which direct pixel addressing occurs.

4.9.3 Read Direct Pixel Data

((W/R DPA))	(W)(CBITS)
111110 110001	1rCCCC
3 E 3 1	

CCCC Number of bits per pixel

The CCCC field specifies the number of bits per pixel and the packing of pixel data in the *Write/Read Direct Pixel Data* command. Pixel data is read from video memory according to the parameters set by the *Set Direct Pixel Addressing Parameters* command. The data stream through this command can be used with the *Write/Read Direct Pixel Data* command, exactly as read, to replicate the screen area. Table 4-11 shows the packing for 1, 2, 3, and 4 bits per pixel. The first byte returned is a flag followed by up to 79 bytes of pixel data. The flag byte is as follows:

Flag	*Condition*
00	Full record, 79 bytes
01	Error, outside of window
02	Last record

Table 4-11 Pixel Packing

BITs/Pixel	BITs Packing							
	7	6	5	4	3	2	1	0
1	X	1	P5	P4	P3	P2	P1	P0
2	X	1	P2		P1		P0	
3	X	1	P1			P0		
4	X	1	X	X	P0			

4.9.4 Write Direct Pixel Data

((W/R DPA))	(W)(CBITS)	(DATA)
111110 110001	0rCCCC	data
3 E 3 1		

CCCC Number of bits per pixel

data A maximum dependent on window size

The CCCC field specifies the number of bits per pixel and the packing of pixel data in the *Write/Read Direct Pixel Data* command. Pixel data is written into video memory according to the parameters set by the *Set Direct Pixel Addressing Parameters* command.

5.0 PERIPHERAL CONTROL COMMANDS

The following peripheral devices are available, and their commands are discussed throughout this section:

- Keyboard
- Data entry keypad
- Joystick
- Tablet
- Pick module
- Function keys/lights
- Valuators

Extended capabilities of several of the peripherals listed above are individually explained in this section.

The commands discussed in this section are to be used with peripheral devices attached to the Whizzard 1645 system. Peripheral devices respond to four basic commands; *Enable*, *Sample*, *Read*, and *Disable*.

Enable: Permits sampling or event queuing of the device.

Sample: Transmits current data from specified device.

Read: Causes an enable command, then waits for a device interrupt; the interrupt triggers a sample command followed immediately by a disable command.

Disable: Prevents further use of the device for graphics data input until the next enable or read command is issued.

5.1 Event Handling Commands

Two event handling commands are discussed briefly on the following pages, as outlined in Table 5-1

Table 5-1 Event Handling Commands

Binary	Hex	Total Bytes	Command
101010	2A	4	Await Event
111110	3E	2	Flush Events (flush event queue)

5.1.1 Await Event

(AWAIT EVENT)	parameters
101010	Tmstimeout-lstimeout rrrrrr
2 A	

T	*Action*
0	Wait forever
1	Wait until timeout expires

mstimeout-lstimeout 11-bit timeout in seconds

Data from the peripheral events is stored in dynamic memory. Since the amount of memory required for any one event depends on the device on which the event occurred, the number of events for which data can be stored will vary. When the event data memory is full, subsequent events are lost. Table 5-2 shows typical byte requirements for events from different devices.

Table 5-2 Byte Requirement

Device	Size in Bytes	4-Bit Device Number
Keyboard	128	0000
Data entry keypad	16	0001
Joystick	32	0010
Tablet	32	0011
Pick module	128	0100
Valuator	32	0101
Function keys/lights	32	0110

The *Await Event* command examines the current event queue. If events are present, the oldest is returned and discarded from the queue. If no events are present, the Whizzard 1645 waits (until the time period specified has expired) for an event to occur.

If no event is found, the Whizzard 1645 returns:

1rrrrr term

If an event is found, the Whizzard 1645 returns data in the following format:

```
0rrrrr rrDDDD rrNNNN data term
```

DDDD 4-bit number of the device (Table 5-2)

NNNN 3-bit valuator number, or 4-bit function key number if DDDD is a valuator or function key device

data For format of this device-dependent data, see Sample command of the device

term The response number

5.1.2 Flush Event Queue

((FLUSH EVENTS))			
111110		000000	
3	E	0	0

Flush Event Queue removes data for all events from the queue.

The remainder of Section Five discusses the individual peripheral control commands.

5.2 Keyboard

In graphics mode, the keyboard may be used to enter an ASCII string. This data is stored in a buffer until it is requested by the host computer, at which time it is transmitted. The data in the buffer may be displayed in characters of size 2 along the bottom edge of the screen. Up to 240 characters may be stored in the buffer, although only 128 can be displayed without wraparound.

Table 5-3 lists the keyboard commands.

Table 5-3 Peripheral Keyboard Commands

Binary	Hex	Total Bytes	Command
110000	30	3	Enable Keyboard
110001	31	2	Disable Keyboard
110010	32	2	Sample Keyboard
110011	33	3	Read Keyboard
110111	37	13	Set Keyboard Scroll Buffer Parameters
111010	3A	3 + N	Write to Keyboard Scrolling Buffer

5.2.1 Enable Keyboard

(ENABLE)	(KEY)	(FUNCTIONS)
110000	rr0000	rrrVCE
3 0		

V *Mode*
0 Enabled for sample mode
1 Enabled for event mode

C *Action*
0 Wait until a carriage return is entered
1 Single character generation

E *Echo Response*
0 No echo
1 Echo characters on the three line scrolling buffer located at the bottom of the screen

The buffer is cleared, and operation of the keyboard in graphics mode is permitted. Echo mode is set to determine whether or not the characters will be displayed. When echo mode is on, the operator can see the accumulation unsampled character string in the keyboard buffer.

5.2.2 Disable Keyboard

(DISABLE)	(KEY)
110001	rr0000
3 1	

Further graphics input by the keyboard is prevented until the host computer issues an *Enable* or *Read* command. The disabled keyboard functions as in terminal mode, sending characters directly to the host computer.

5.2.3 Set Keyboard Scroll Buffer Parameters

(SET PARAMS)	(KEY)	parameters
110111	rr0000	msxpos-lsxpos msypos-lsypos
3 7		rrrrmscpl-lscpl rrrrmslps-lslps
		rrinten rrrrrr rrrrrr

msxpos-lsxpos X coordinate of the upper left corner of the scrolling buffer

msypos-lsypos Y coordinates of the upper left corner of the scrolling buffers

mscpl-lscpl 8-bit number specifying number of characters per line

mslps-lslps 8-bit number specifying the number of lines in the scrolling buffer

inten 4-bit intensity

The memory allocated for the scrolling buffer holds up to 1000 characters. The user can configure the buffer in any combination of characters per line per screen so long as the total does not exceed 1000 bytes. The default configuration is three lines of 80 characters placed in the lower left corner of the screen.

If this command is transmitted when the buffer contains data, that data is lost.

5.2.4 Write to Keyboard Scrolling Buffer

(WRITE PERIPH)	(KEY)	parameters
111010	rr0000	char . . . char GS
3 A		

Unlike the terminal buffer, this buffer is controlled by the user. Its parameters can be set by using the *Set Keyboard Scroll Buffer Parameters* command.

Upon receipt of the GS character, the following message is returned:

Errrrr

E	*Error*
0	No errors; write performed
1	Error; write not performed

5.2.5 Sample Keyboard

```
(SAMPLE)        (KEY)
 110010         rr0000
   3  2
```

The oldest data in the keyboard buffer, up to and including the oldest carriage return, is returned to the host. Since a carriage return may be absorbed by the host computer operating system, a bit in the host computer input indicates whether a carriage return was actually typed. The data disappears from the buffer, causing the display of the buffer contents to shift to the left.

The format of the returned data is as follows:

```
ETMVCC  CCCCCC  char1  char2  . . .  charN  term
```

E	*Enable Indicator*
0	Keyboard is enabled; data follows
1	Keyboard is not enabled; terminator follows
T	*Terminator Indicator*
0	Data sent did not contain a carriage return at the end of the buffer
1	Data sent did contain a carriage return at the end of the buffer
M	*Buffer Underflow*
0	Buffer contained data which follows
1	Buffer was empty when sampled; terminator follows
V	*Buffer Overflow*
0	No buffer overflow
1	Some data has been lost because the buffer was full and additional characters were typed
CC..CC	8-bit count of characters
char1...charN	The characters that were in the buffer, except the carriage return at the end, if it was present
term	The response terminator

5.2.6 Read Keyboard

(READ)	(KEY)	(ECHO)
110011	rr0000	rrrrrE
3 3		

E *Echo Response*
0 No echo
1 Echo characters on the three line scrolling buffer located at the bottom of the screen

This command enables the keyboard, which also clears the keyboard buffer. The keyboard buffer then accumulates keycodes until a carriage return is pressed, and the buffer contents are returned to the host computer as if a ***Sample Keyboard*** command had been sent. The keyboard is then disabled, as if a ***Disable Keyboard*** command had been sent, and the data is returned in the format of the ***Sample Keyboard*** command.

Table 5-4 lists the data entry keypad commands.

Table 5-4 Peripheral Data Entry Keypad Commands

Binary	Hex	Total Bytes	Command
110000	30	3	Enable Data Entry Keypad
110001	31	2	Disable Data Entry Keypad
110010	32	2	Sample Data Entry Keypad
110011	33	3	Read Data Entry Keypad

5.3 Data Entry Keypad

The data entry keypad, a group of 14 keys to the right of the keyboard, can return to the host computer any of 56 different codes through the combined use of the control and shift keys.

5.3.1 Enable Data Entry Keypad

(ENABLE)	(PAD)	(MODE)
110000	rr0001	rrrVrr
3 0		

V *Mode*
0 Enable for sample mode
1 Enable for event mode

A string of up to 15 data entry keypad keycodes is accumulated in the data entry keypad buffer for later transmission to the host computer.

5.3.2 Disable Data Entry Keypad

(DISABLE)	(PAD)
110001	rr0001
3 1	

Further operation of the data entry keypad is prevented until the host computer issues an *Enable Data Entry Keypad* or *Read Data Entry Keypad* command.

5.3.3 Sample Data Entry Keypad

(SAMPLE)	(PAD)
110010	rr0001
3 2	

The contents of the data entry keypad buffer are returned to the host computer. The format of the returned data is as follows:

E0CCCC	char1	char2...charN	term

E	*Enable Indicator*
0	Function pad is enabled; data follows
1	Function pad is not enabled; terminator follows
CCCC	4-bit count of characters
char1...charN	Characters in buffer
term	The response terminator

5.3.4 Read Data Entry Keypad

(READ)	(PAD)	
110011	rr0001	rrrrrr
3 3		

The data entry keypad is enabled and its buffer cleared. When any data entry keypad key is pressed, the corresponding keycode is put into the buffer and a *Sample Data Entry Keypad* command is performed, followed by a *Disable Data Entry Keypad* command. Only the one keycode is returned. The data is returned in the format of the *Sample Data Entry Keypad* command.

5.4 Joystick

Movement of the joystick positions a standard or user-definable special cursor at any point on the screen. Location of the cursor can be determined by the host computer as well as the status of the button on top of the handle. The host computer can also request that position data be sent when the button is pushed. Table 5-5 lists the joystick commands.

Table 5-5 Peripheral Joystick Commands

Binary	Hex	Total Bytes	Command
110000	30	3	Enable Joystick Keypad
110001	31	2	Disable Joystick Keypad
110010	32	2	Sample Joystick Keypad
110011	33	3	Read Joystick Keypad
110100	34	10	Set Joystick Limits
110101	35	6	Set JoystickPosition

5.4.1 Enable Joystick

(ENABLE)	(UNIT)(JOY)	parameters
111000	UU0010	CrBVrE
3 0		

UU 2-bit number of joystick unit

C *Cursor position*
0 Initialize cursor at last cursor position
1 Initialize cursor at screen center

B *Position tracking*
0 No positioning information displayed
1 "Rubber Band" position displayed between screen origin and cursor position

V *Mode*
0 Enabled for sample
1 Enabled for event

E *Cursor display*
0 Cursor not displayed
1 Cursor displayed

The position of the joystick cursor is controlled with the joystick handle, and the position data is made available to the host computer. The button position is also available.

5.4.2 Disable Joystick

(DISABLE)	(UNIT)(JOY)
110001	UU0010
3 1	

UU 2-bit number of joystick unit

Operation of the joystick is prevented until the host computer issues an ***Enable Joystick*** or ***Read Joystick*** command.

5.4.3 Sample Joystick

(SAMPLE)	(UNIT)(JOY)
110010	UU0010
3 2	

UU 2-bit number of joystick unit

The X-Y coordinates of the joystick cursor and the state of the button are transmitted to the host computer in the format:

ErrrrB msx-lsx msy-lsy 1000000 term

E *Cursor Echo*
0 No echo
1 Echoed cursor position tracking on the screen

B *Button Position*
0 Button is not pressed
1 Button is pressed

msx-lsx The most significant and least significant six bits of the 12-bit X value of the cursor screen coordinate.

msy-lsy The most significant and least significant six bits of the 12-bit Y value of the cursor screen coordinate.

term The response terminator

5.4.4 Read Joystick

(READ)	(UNIT)(JOY)	parameters
110011	UU0010	CrBrrE
3 3		

UU — 2-bit number of joystick unit

C — *Cursor Position*
0 — Initialize cursor at last cursor position
1 — Initialize cursor at screen center

B — *Position Tracking*
0 — No positioning information displayed
1 — "Rubber band" position displayed between screen origin and cursor position

E — *Cursor Display*
0 — Cursor is not displayed
1 — Cursor is displayed

The joystick is enabled and remains so until the button on top of the joystick is pushed and released. When the button is released, a *Sample Joystick* command is performed. The joystick is then disabled. The data is returned in the format of the *Sample Joystick* command.

5.4.5 Set Joystick Limits

(SET LIMITS)	(UNIT)(JOY)	parameters
110100	UU0010	msbminx-lsbminx msbminy-lsbminy
3 4		msbmaxx-lsbmaxx msbmaxy-lsbmaxy

UU	2-bit number of joystick unit
msbminx-lsmminx	Forms the 12-bit lower limit of the horizontal coordinate of the locator position
msbminy-lsbminy	Forms the 12-bit lower limit of the vertical coordinate of the locator position
msbmaxx-lsbmaxx	Forms the 12-bit upper limit of the horizontal coordinate of the locator position
msbmaxy-lsbmaxy	Forms the 12-bit upper limit of the vertical coordinate of the locator position

The joystick cursor will locate only those X-Y positions which are within the rectangle formed by the upper and lower horizontal and vertical limits.

5.4.6 Set Joystick Position

(SET POS)	(UNIT)(JOY)	Parameters
110101	UU0010	msbx-lsbx msby-lsby
3 5		

UU	2-bit number for joystick unit
msbx-lsbx	12-bit concatenation giving the X coordinate
msby-lsby	12-bit concatenation giving the Y coordinate

The cursor of the joystick is moved to the X-Y position specified in the command. If the device is not owned, the command has no effect.

5.4.7 Select Cursor

(SELECT CURSOR)	(UNIT)	(SELECT)	parameters
110110	UUDDDD	rrrrrS	mssub-lssub
3 6			

UU 2-bit number of joystick unit

DDDD *4-bit number of the device*
0010 Joystick
0011 Tablet
0100 Pick Module

S *Select cursor*
0 Cursor defined by the subroutine
1 System default cursor

mssub-lssub Specified the 12-bit subroutine number

This command selects a display list routine to draw the cursor for the joystick, tablet, or pick module. The subroutine is identified by its subroutine number. This is formed by the concatenation of msbsubnr-lsbsubnr. These bytes must always be present, even though they are ignored if the S bit is set to select the system default cursor. (S = 1).

5.4.8 Locate Position

(LOCATE POSITION)	(UNIT)
111001	UUDDDD
3 9	

UU 2-bit number of joystick unit

DDDD *4-bit number of the device*
0010 Joystick
0011 Tablet
0100 Pick Module

This command adds a move or draw instruction to the display list. It is inserted at the location of the current insert pointer, and moves the beam to the location of the cursor.

5.5 Tablet

The tablet is a flat surface with a moveable position sensing device: a pen or four-button cursor. The host computer can determine the position of the device, as well as the status of the pen or cursor buttons. A standard or user-definable graphic cursor can be displayed to track the device.

Table 5-6 lists the tablet commands.

Table 5-6 Peripheral Tablet Commands

Binary	Hex	Total Bytes	Command
110000	30	3	Enable Tablet
110001	31	2	Disable Tablet
110010	32	2	Sample Tablet
110011	33	3	Read Tablet
110100	34	10	Set Tablet Limits
110101	35	6	Set Tablet Position
110111	37	7	Set Tablet Parameters

5.5.1 Enable Tablet

(ENABLE)	(TABL)	parameters
111000	rr0011	CrBVrE
3 0		

C *Cursor position*
0 Initialize cursor at last cursor position
1 Initialize cursor at screen center

B *Position tracking*
0 No positioning information displayed
1 "Rubber Band" position displayed between screen origin and cursor position

V *Mode*
0 Enabled for sample
1 Enabled for event

E *Cursor display*
0 Cursor not displayed
1 Cursor displayed

The tablet position data is made available to the host computer. Button data is also available for up to four buttons. A graphic cursor may optionally track the tablet pen or cursor on the screen.

5.5.2 Disable Tablet

(DISABLE)	(TABL)
110001	rr0011
3 1	

The tablet is prevented from operating until the host computer issues an *Enable Tablet* or *Read Tablet* command.

5.5.3 Sample Tablet

(SAMPLE)	(TABL)
110010	rr0011
3 2	

The X-Y coordinates of the tracking device are returned to the host computer, as well as the status of the pen or cursor's buttons, in the format:

EDBBBB msx—xbits—lsx msy—ybits—lsy term

E	*Echo Response*
0	Valid data follows
1	Tablet not enabled or allocated to other user; terminator follows
D	*Tablets*
0	Single
1	Dual
BBBB	The status of the four buttons, numbered 0 through 3 from left to right; For each B: 0 = button released, 1 = button depressed
msx-xbits-lsx	16-bit X-coordinate data
msy-ybits-lsy	16-bit Y-coordinate data
term	The response terminator

Coordinate data returned in the 16-bit words is dependent on the resolution of the tablet in use. The actual data will be left justified within the 16-bit word. For example, a 13-bit tablet will show the most significant 4 bits of Y in msy, the next 6 bits in ybits, and the 3 least significant bits of Y in the 3 most significant bits of lsy. The remaining 3 bits of lsy will be set to zero.

5.5.4 Read Tablet

(READ)	(TABL)	parameters
110011	rr0011	CrBrrE
3 3		

C *Cursor Position*
0 Initialize cursor at last cursor position
1 Initialize cursor at screen center

B *Position Tracking*
0 No positioning information displayed
1 "Rubber band" position displayed between screen origin and cursor position

E *Cursor Display*
0 Cursor is not displayed
1 Cursor is displayed

The tablet is enabled and remains so until the operator pushes and releases any of the buttons on the tracking device. Button action causes a *Sample Tablet* and then a *Disable Tablet* command to be executed. The data returned is in the same format as that of the *Sample Tablet* command.

5.5.5 Set Tablet Parameters

(SET PARAMS)	(TABL)	parameters
110111	rr0011	rrDSSS msxoff-lsxoff msyoff-lsyoff
3 7		

D	*Action*
0	Tablet is used as a locator
1	Tablet is used as a digitizer in the positive range only

SSS	*Nominal Scale Factor*	*Number of Tablet Bits*	*Hexadecimal Offset for 1-to-1 Mapping*
001	Double	11	FC0
000	Unity	12	F 80
011	Half	13	F 00
010	Quarter	14	E 00
101	Eighth	15	C00
100	Sixteenth	16	8 00
110	Reserved		
111	Reserved		

msxoff-lsxoff Forms the 12-bit X offset value

msyoff-lsyoff Forms the 12-bit Y offset value

The maximum tablet resolution supported is 16 bits. The offset is the distance in tablet units between the desired origin of coordinates and the "effective" center of the tablet. The effective center of the tablet is the point at which the most significant bit returned by the table changes state.

If D=1, then the origin is the lower left corner of the screen (this is how the digitizer is used). The locator has the origin being the center of the screen. The 1 to 1 offset is added to the new raw data coming from the tablet and then shifted the appropriate number of times to make the 12 bit X, Y coordinates.

5.5.6 Set Tablet Limits

(SET LIMITS)	(TABL)	(LIMITS)
110100	rr0011	msbminx-lsbminx msbminy-lsbminy
3 4		msbmaxx-lsbmaxx msbmaxy-lsbmaxy

msbminx-lsmminx — Forms the 12-bit lower limit of the horizontal coordinate of the locator position

msbminy-lsbminy — Forms the 12-bit lower limit of the vertical coordinate of the locator position

msbmaxx-lsbmaxx — Forms the 12-bit upper limit of the horizontal coordinate of the locator position

msbmaxy-lsbmaxy — Forms the 12-bit upper limit of the vertical coordinate of the locator position

The tablet cursor will stay within the rectangle formed by the upper and lower horizontal and vertical limits.

5.5.7 Set Tablet Position

(SET POS)	(TABL)	parameters	
110101	rr0011	msbx-lsbx	msby-lsby
3 5			

msbx-lsbx 12-bit concatenation giving the X coordinate

msby-lsby 12-bit concatenation giving the Y coordinate

The cursor of the joystick is moved to the X-Y position specified in the command. The offset between the screen origin and the tablet origin is changed to accommodate the new cursor position. If the device is not owned, the command has no effect.

5.6 Pick Module

This device allows the selection of any specific portion of the display, or one of the locator devices (joystick or tablet), with the digital comparator. The host computer may request a block of data describing the last picture element selected.

Table 5-7 Pick Module Commands

Binary	Hex	Total Bytes	Command
110000	30	3	Enable Pick Module
110001	31	2	Disable Pick Module
110010	32	2	Sample Pick Module
110011	33	3	Read Pick Module
110111	37	7	Set Locator and Pick Window
111110	3E		Set Additional Parameters

5.6.1 Enable Pick Module

(ENABLE)	(PICK)	parameters
110000	rr0100	CTBVRE
3 0		

C *Cursor Position*
0 Initialize cursor at last position
1 Initialize cursor at screen center

T *Digital Comparator Select*
0 Use comparator
1 Use light pen

B *Position Tracking*
0 No positioning information displayed
1 "Rubber Band" position displayed between screen origin and cursor position

V *Mode*
0 Enable for sample
1 Enable for event mode

R *Auto-Raster Enable*
0 Do not enable light pen auto raster
1 Enable light pen auto raster; necessary for locating the light pen if it is not over a detectable vector

E *Cursor Display*
0 Cursor is not displayed
1 Cursor is displayed

The pick module is turned on, allowing the selection of vectors or larger picture units with the light pen, joystick, or tablet. The *Select Locator* and *Pick Window* command selects the joystick or tablet.

5.6.2 Disable Pick Module

(DISABLE)	(PICK)
110001	rr0100
3 1	

The pick module cannot be operated until the host computer issues an *Enable Pick Module* or *Read Pick Module* command.

5.6.3 Sample Pick Module

(SAMPLE)	(UNIT)(PICK)	reserved
110010	UU0100	rrrrrr
3 2		

UU 2-bit number of pick unit

The contents of the pick module block are returned in the following format

(group 1)	(group 2)	...(group 10)	check-sum	term

Each data group consists of six characters in the format

rrAAAA	BBBBBB	CCCCCC	rrDDDD	EEEEEE	FFFFFF

and constitutes one 32-bit word of graphics memory in the format:

AAAABBBBBBCCCCCCDDDDEEEEEEFFFFFF

Contents of the returned data are explained in the *Read Pick Module* command. The check-sum is the modulo 63 sum of the 60 preceding characters.

If a locator is being used in conjunction with the pick module, pick module, pick data has the format:

ErLLLL	LLLLLL	LLLLLL	XXXXXX	XXXXXX	YYYYYY	YYYYYY	term

LL..LL Segment label

XX..XX 12-bit X cross hair

YY..YY 12-bit Y cross hair

If pick is not enabled or allocated, the system returns:

100000 term

Contents of the pick module control block are followed by a check-sum and the terminator. The contents of the returned data are described in the *Read Pick Module* command (see next page).

5.6.4 Read Pick Module

(READ)	(PICK)	parameters
110011	rr0100	CTBrRE rrrrFF
3 3		

C *Cursor Position*
0 Initialize cursor at last cursor position
1 Initialize cursor at screen center

T *Digital Comparator Select*
0 Use comparator
1 Use light pen

B *Position Tracking*
0 No positioning information displayed
1 "Rubber band" position displayed between screen origin and cursor position

R *Auto-Raster Enable*
0 Do not enable light pen auto raster
1 Do enable light pen auto raster; necessary for locating the light pen if it is not over a pickable vector

E *Cursor Display*
0 Cursor is not displayed
1 Cursor is displayed

FF *Format for Data Returned to Host*
00 10 data groups of 32-bit graphics memory words (see below)
01 Segment label followed by 12-bit cross hair position

The pick module is enabled and remains so until its status is changed by the release or press of the locator button or of the sensor of the light pen. This causes a sample and then a disable of the pick module.

The data returned by the pick module is in the format:

data1	data2	data3	data4	data5
data6	data7	data8	data9	data10
checksum	term			

The ten data groups consist of the contents of the pick module control block.

The first data group contains status and vector count:

Bit 0 (N)	If 1, this bit indicates that the host computer has not responded to a previous pick module interrupt; when the host computer responds to the interrupt, it enables further picking
Bit 1 (U)	If 1, this bit indicates that an update of label information is to be performed; the labels for which addresses exist are loaded with the addresses
Bit 2 (R)	If set to 1 by the pick module, a forced auto-raster is produced (if pick module was enabled with auto-raster generation)
Bit 3 (P)	If 1, the light pen is on (button down or conducting); if (0), it is off (button up or non-conducting)
Bits 4-15	Reserved
Bits 16-31	Contain the number of vectors (after clipping) since last label was encountered; if clipping occurs, the vector count may not match the number of display list vectors

The second data group holds the segment label address and segment label containing the sensed or picked vector

Bits 0-15	Contain the segment label
Bits 16-31	Contain the segment label address

The third group contains the subsegment label address and subsegment label containing the sensed or picked vector

Bits 0-15	Contain the subsegment label
Bits 16-31	Contain the subsegment label address

Data groups 4 through 7 contain the subroutine addresses and labels of the sensed or picked vector. These groups contain zeroes if they exist at a level lower than the subroutine level of the sensed or picked vector or if the vector is not in a subroutine.

Fourth Group

Bits 0-15	Contain the first subroutine level label
Bits 16-31	Contain the first subroutine level label address

Fifth Group

Bits 0-15	Contain the second subroutine level label
Bits 16-31	Contain the second subroutine level label address

Sixth Group

Bits 0-15	Contain the third subroutine level label
Bits 16-31	Contain the third subroutine level label address

Seventh Group

Bits 0-15	Contain the fourth subroutine level label
Bits 16-31	Contain the fourth subroutine level label address

The eighth data group is reserved.

The ninth and tenth data groups contain the cursor position and the pick position in screen coordinates.

Ninth Group

Bits 0-3	Reserved
Bits 4-15	Cursor position X coordinate
Bits 16-19	Reserved
Bits 20-31	Cursor position Y coordinate

Tenth Group

Bits 0-3	Reserved
Bits 4-15	Pick position X coordinate
Bits 16-19	Reserved
Bits 20-31	Pick position Y coordinate

Each of the ten data groups returned consists of six characters:

```
rrAAAA BBBBBB CCCCCC rrDDDD EEEEEE FFFFFF
```

Each data group corresponds to one 32-bit vector memory word:

```
AAAABBBBBBCCCCCCDDDDEEEEEEFFFFFF
```

The checksum is the modulo-63 sum of the 60 preceding characters. If the pick module is not enabled when sampled, only the following is returned:

```
100000 term
```

term Is the response terminator

5.6.5 Set Locator and Pick Window

(SET PARAMS)	(PICK)	(LOCATOR)	Pick parameters
110111 3 7	rr0100	rrDDDD	rrrrMM XXXXXX rrrrNN YYYYYY

DDDD *Indicates the Source of Pick Input*
0010 Joystick input
0011 Tablet input

MM-XXXXXX Is the 8-bit width used by the pick module to determine if a pickable vector is close enough to the pick cursor to be picked

NN-YYYYYY Is the 8-bit height used by the pick module to determine if a pickable vector is close enough to the pick cursor to be picked

When the joystick or tablet is the picking device for the digital comparator and the pick module, the two specified values determine the picking range. They define the rectangular "window" centered on the pick cursor, within which a vector can be detected.

5.6.6 Set Peripheral Additional Parameters

((SET PERIPH))	(DATA)	parameters
111110 000101 3 E 0 5	rr0100	rrrMMM

MMM
000 Pick, no return if no hit
001 Pick is used as a locator
100 Return pick information, even if no hit
101 Pick as locator, return even if no hit

All others use pick as pick only.

5.7 Function Keys and Valuators

The function keys/lights consist of 16 labeled red keys, each with an associated lamp. Three valuators are located on the left side of the keyboard. These devices are disabled as a whole, but each valuator and function key/light is otherwise treated as an individual unit.

A special internal structure allows each valuator to turn continuously without encountering stops or "dead zones" normally found in potentiometers. Each turn contains a discrete resolution of 348 ± 6 elements that are used to calculate a current 16-bit value.

Each function key has a position and value. The position indicates whether the key is up or down. The value is a logical variable (1 or 0) associated with the position of the key. A function key can be programmed to be momentary or toggled. In momentary mode, the value tracks the position, either in-phase or out-of-phase, depending on initialization. In toggle mode, each press of the key causes the value to change to the opposite state.

The light associated with each of the 16 function keys can be turned on or off, or made to blink at 1Hz or 8Hz. A light can be coupled to a function key so that changing value of the function key, either programmatically or by pressing it, changes the state of the light. If the light is initially off when the coupling is enabled, the light alternates between off and on as the function key value changes. If it is blinking, it alternates between blinking and off.

A programmatic change to the state of a light coupled to a function key does not affect the value of the function key.

Table 5-8 Valuator and Function Key/Light Commands

Binary	Hex	Total Bytes	Command
110000 XX0101	30 X5	4	Enable Valuator
110000 XX0110	30 X6	3	Enable Function Keys/Lights
110001 XX0101	31 X5	3	Disable Valuator
110001 XX0110	31 X6	2	Disable Function Keys/Lights
110010 XX0101	32 X5	3	Sample Valuator
110010 XX0110	32 X6	2	Sample Function Keys/Lights
110011 XX0101	33 X5	3	Read Valuator
110011 XX0110	33 X6	3	Read Function Keys/Lights
110111 XX0101	37 X5	15	Set Valuator Parameters
110111 XX0110	37 X6	7	Set Function Keys/Lights Parameters
110111 XX0111	37 X7	7	Set Valuator Threshold

5.7.1 Enable Valuators

(ENABLE)	(VALUATORS)	parameters
110000	rr0101	rrrVrr rrrrNN
3 0		

V	*Mode*
0	For sample
1	For events

NN 2-bit number of a valuator (0-3)

The specified valuator is enabled.

5.7.2 Disable Valuator

(DISABLE)	(VALUATORS)	(VALUNUM)
110001	rr0101	rrrrNN
3 1		

NN 2-bit number of valuator (0-3)

Operation of the valuators is prevented.

5.7.3 Enable Function Keys/Lights

(ENABLE)	(LIGHTS)	parameters
110000	rr0101	rrrVrr
3 0		

V	*Mode*
0	Enable for sample
1	Enable for event mode

The function keys/lights are enabled.

5.7.4 Disable Function Keys/Lights

(DISABLE)	(LIGHTS)
110001	rr0101
3 1	

Further operation of the function keys/lights is prevented.

5.7.5 Sample Valuator

(SAMPLE)	(VALUATORS)	(DAILNUM)
110010	rr0101	rrrrNN
3 2		

NN Number of the valuator to sample (0-3)

The data is returned in the format:

ErAAAA BBBBBB CCCCCC term

E *Status*
0 Enabled to receive valid data
1 Not enabled: terminator follows

AAA....CCC Forms the value of the valuator

term Is the response terminator

If E = 0, data is returned in the 16-bit format:

AAAABBBBBBCCCCCC

5.7.6 Sample Function Keys/Lights

(SAMPLE)	(LIGHTS)
110010	rr0110
3 2	

The position, value, and lamp state are returned for all the function keys/lights in the form:

Er0000	rrPPPP	PPPPPP	PPPPPP	
	rrVVVV	VVVVVV	VVVVVV	
	rrLLLL	LLLLLL	LLLLLL	term

E *Status*
0 Enabled to receive valid data
1 Not enabled; terminator follows

P *Action*
0 Function key not pressed
1 Function key pressed

V *Action*
0 Function key is off
1 Function key is on

L *Action*
0 Light is off
1 Light is on

term Is the response terminator

5.7.7 Read Valuator

(READ)	(VALUATORS)	(VALNUM)
110011	rr0101	rrrrNN
3 3		

NN Number of the control valuator (0-3)

The command enables the specified valuator. An event is triggered when the valuator exceeds its event threshold. The threshold is set by the Set Valuator Parameters command. Data is sampled and the valuator is disabled.

5.7.8 Read Function Keys/Lights

```
(READ)          (LIGHTS)
110011          rr0110          rrrrrr
 3   3
```

The system waits for any key to be pressed. Then the position, value, and light status for all 16 keys and lights are returned as 10 characters:

```
ErSSSS      rrPPPP      PPPPPP      PPPPPP
            rrVVVV      VVVVVV      VVVVVV
            rrLLLL      LLLLLL      LLLLLL      term
```

E	*Status*
0	Valid data follows
1	Not enabled; terminator follows
SSSS	Number of the key causing the event
P	
0	Key not pressed
1	Key pressed
V	
0	Key is off
1	Key is on
L	
0	Light is off
1	Light is on
term	Is the terminator

5.7.9 Set Valuator Parameters

(SET PARAMS)	(VALUATORS)	parameters
110111 3 7	rr0101	rrrrNN rrMMMM MMMMMM MMMMMM rrXXXX XXXXXX XXXXXX rrGGGG GGGGGG GGGGGG rrVVVV VVVVVV VVVVVV

NN — Selects the valuator (0-3)

MMM...MMM — Sets the minimum value (greater than or equal to -32768) that the valuator will generate; when the minimum value has been reached, continued decrease of the valuator has no effect

XXX...XXX — Sets the maximum value (less than or equal to 32767) which the valuator will generate; when the maximum value has been reached, continued increase of the valuator has no effect

GGG...GGG — Is an integer (0 to 65,535) set to interpret turns of the control dial in terms of current valuator value; new control dial value is calculated from the current value plus (80 x turns x this value)

VVV...VVV — Sets the current valuator value as a signed integer (-32768 to 32767); control dial clockwise rotation will increase the value; counter-clockwise rotation will decrease the value

5.7.10 Set Function Key/Light Parameters

(SET PARAMS)	(LIGHTS)	parameters
110111	rr0110	rrKKKK MMMMMM NNNNNN CCTTVV 000LLL
3 7		

KKK...NNN — Represents the keys 15 through 0 (most significant bit for key 15; least significant bit for key 0) selected for association of parameters; if the bit corresponding to a key is 1, the key parameters are changed

CC *Couple Mode*
00 No change
10 Uncouple
11 Couple

TT *Toggle Mode*
00 No change
10 Momentary — Function key changes value only while held down
11 Toggle — Function key changes value each time it is pressed

VV *Function Key Value*
00 No change
10 Function key value is 0 when not pressed
11 Function key value is 1 when not pressed

LLL *Light Value*
000 No change
100 Light off
101 Light on
110 Light blinking at 1 Hz
111 Light blinking at 8 Hz

5.7.11 Set Valuator Threshold

(SET PARAMS)	(VALUATORS)	parameters
110111	rr0111	rrrrrr rrrNNN rrmsthrsd thrsd lsthrsd
3 7		

NNN	Selects control valuator (0 to 7)
msthrsd	4 most significant bits of threshold
thrsd	6 next bits of threshold
lsthrsd	8 least significant bits of threshold

The event threshold is set for the specified valuator. The threshold value (0 to 32767) determines the amount of change in valuator resolution needed to generate a "new-value-interrupt."

5.8 Generate Hard Copy

(HARD COPY)	(DATA)
101011	BArrrr rFSMMM
2 B	

B *Feed Form Before Copy*
0 No
1 Yes

A *Feed Form After Copy*
0 No
1 Yes

F *Color or Black/White*
0 Color
1 Black/White

S *Black/White Reverse*
0 No
1 Yes

MMM *Is the 3-bit menu value from one to six*
001 Graphics
010 Alpha Overlay
011 Graphics and Alpha
100 Line Printer
101 Debug Output (line analyzer)
110 Exit

This command returns the following data upon completion of the print job:

rrrrrr terminator

APPENDIX A

1645 COMMAND SET SUMMARY - in binary order

Binary	Hex	Command	Page
000000	00	Open Segment	3-8
000001	01	Append Segment	3-10
000010	02	Clear Segment	3-10
000011	03	Delete Segment	3-9
000100	04	Close Segment	3-9
000101	05	Rewrite	
		- Segment	3-11
		- Subroutine	3-16
000110	06	Rename Segment	3-12
000111	07	Delete All Segments	3-10
001000	08	Change Attributes (1 character commands)	
		- Segment Color	4-9
		- Segment Dash Enable	4-9
		- Segment Transformation Enable	4-13a
		- Segment Visibility	4-10
001001	09	Change Segment Attributes (2 character commands)	
		- Segment Detectability	4-32
		- Segment Dash Pattern	4-10
001010	0A	Change Segment Transformation Parameters	
		- 2D Segment Origin	4-14
		- 2D Segment Matrix	4-15a
		- 3D Segment Origin	4-14a
		- 3D Z Segment Origin	4-15
		- 3D Segment Matrix	4-15a
		- Clip Boundaries	4-16
		- Translation	4-15
001011	0B	Insert Segment Line Type Control (1 character commands)	
		- Color	4-11
		- Dash Enable	4-11
		- Transformation Enable	4-16
001100	0C	Insert Segment Line Type Control (3 character commands)	
		- Pick Label	4-33
		- Dash Pattern	4-12
		- Set Write-Protect Mask	4-12
001101	0D	Insert Transformation Parameters	
		- 2D Segment Origin	4-16a
		- 2D Matrix	4-17a
		- 3D Segment Origin	4-17
		- 3D Z Segment Origin	4-17
		- 3D Matrix	4-17a
		- Clip Boundaries	4-18
		- Translation	4-17a
001110	0E	Set Pick Detectability	4-33
001111	0F	Move/Draw Mixed Vector (2D/3D)	4-19
010000	10	Move 2D Absolute Vector	4-19a
010001	11	Draw 2D Absolute Vector	4-19a
010010	12	Move 2D Relative Vector	4-20

010011	13	Draw 2D Relative Vector	4-20
010100	14	Move 3D Absolute Vector	4-19a
010101	15	Draw 3D Absolute Vector	4-19a
010110	16	Move 3D Relative Vector	4-20
010111	17	Draw 3D Relative Vector	4-20
011000	18	Character String	4-28
011001	19	Move/Draw 2D Short Relative Vector String	4-20a
011010	1A	Move/Draw 3D Short Relative Vector String	4-21
011011	1B	Move/Draw Incremental Vector String	4-22
011100	1C	Marker Position (2D/3D)	4-30
100000	20	Open Subroutine	3-14
100001	21	Append Subroutine	3-16
100010	22	Clear Subroutine	3-15
100011	23	Delete Subroutine	3-15
100100	24	Close Subroutine	3-14
100101	25	Call Subroutine	3-17
100110	26	Rename Subroutine	3-18
100111	27	Delete All Subroutines	3-15
101000	28	Subroutine Special Functions	
		- Special 2D Subroutine Call	3-19
		- Special 2D Subroutine Move	3-20
		- Special 3D Subroutine Call	3-19a
		- Special 3D Subroutine Move	3-20a
		- Special Subroutine Substitution	3-21
101010	2A	Await Event	5-2
101011	2B	Generate Hard Copy	5-36
101110	2E	Reset Terminal	3-2
110000	30	Enable Command	
		- Keyboard	5-5
		- Data Entry Keypad	5-8
		- Joystick	5-10
		- Tablet	5-16
		- Pick Module	5-22
		- Valuator	5-29
		- Function Keys/Lights	5-29
110001	31	Disable Command	
		- Keyboard	5-5
		- Data Entry Keypad	5-9
		- Joystick	5-11
		- Tablet	5-17
		- Pick Module	5-22
		- Valuator	5-29
		- Function Keys/Lights	5-29
110010	32	Sample Device	
		- Keyboard	5-7
		- Data Entry Keypad	5-9
		- Joystick	5-12
		- Tablet	5-17
		- Pick Module	5-23
		- Valuator	5-30
		- Function Keys/Lights	5-31

110011	33	Read Device	
		- Keyboard	5-8
		- Data Entry Keypad	5-9
		- Joystick	5-13
		- Tablet	5-18
		- Pick Module	5-24
		-Valuator	5-31
		-Function Keys/Lights	5-32
110100	34	Set Limits For Device	
		-Joystick	5-14
		-Tablet	5-20
110101	35	Set Device Position	
		-Joystick	5-14
		-Tablet	5-21
110110	36	Select Cursor	5-15
110111	37	Set Parameters For Device	
		-Keyboard Scroll Buffer	5-6
		-Tablet	5-19
		-Locator And Pick Window	5-27
		-Valuator	5-33
		-Function Key/Light	5-34
		-Valuator Threshold	5-35
111001	39	Locate Position	5-15
111010	3A	Write To Keyboard Scrolling Buffer	5-6
111100	3C	Read Current Position ID	3-3
111110	3E	Escape To Additional Commands	

Binary	Hex		
000000	00	Flush Event Queue	5-4
000001	01	Set Erase-Protect Mask	4-5
000101	05	Set Peripheral Additional Parameters	5-27
001000	08	Set Polygon Fill (enter polygon mode)	4-24
001001	09	Exit Polygon Fill (exit polygon mode)	4-27
001010	0A	Marker Selection	4-29
001011	0B	Read Free Memory Size	3-4
001110	0E	Set Rectangle Fill	4-26
010100	14	Set Baseline Fill	4-25
011000	18	Repaint Entire Screen	4-6
011001	19	Set Selective Erase Repaint	4-4
011010	1A	Set Background Color	4-7
011101	1D	Set Dash Mode	4-7
011110	1E	Set Pixel Write Mode	4-34
011111	1F	Set Screen Repaint Mode	4-3
100000	20	Set Overlay Control	4-5
110000	30	Set Direct Pixel Addressing Parameters	4-35
110001	31	Write/Read Direct Pixel Data	4-36

111111	3F	Escape to Additional Modes	

APPENDIX B

MATRIX TRANSFORMATIONS

With the Whizzard 1645, any segment with a "transform" type header can be translated, rotated, or scaled by supplying the appropriate matrix elements as attributes.

These transformation capabilities relieve the host computer of the burden of transforming each individual vector of a segment. For a segment of 1000 vectors, many thousands of arithmetic operations are performed. The transformation parameters are simply passed to the Whizzard 1645 system.

2D Transformations

Any 2D vector can be represented by the row matrix:

$$X\ Y$$

Any line transformation on the above 2D coordinate can be performed with a real 2x2 matrix. The resulting vector is the matrix multiplication of the original vector and the transformation matrix. The above vector representation can be used for scaling and rotation but is limited to linear transformations of the form:

$$x' = ax + by$$

In order to translate to a point other than the screen origin, a third value must be used with the 2D vector to create the three-element row matrix:

$$V = [X\ Y\ 1]$$

This augmented vector is now a 2D representation in what is know as the homogeneous coordinate system which overcomes the limitations of the linear transformations. Transformations can now be performed on equations of the general form:

$$x' = ax + by + c$$

3x3 matrices can now be created which will allow 2D transformations such as translation and non-center-of-screen scaling and rotation.

Many simple individual transformations may be concatenated into a complex resultant matrix of the same dimensions. The final matrix can then be passed to the hardware to perform an elaborate transformation on all vectors of a given segment. A general 2D transformation can be made up from the three simple transformations for translation, scaling, and rotation.

Translation is performed with the matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ TX & TY & 1 \end{bmatrix}$$

Each vector is then post-multiplied by this matrix to give a new vector

$$V' = VT$$

Following through the matrix multiplication we get the three new equations, where the tautology "1 = 1" is shown here to keep the proper 3 x 3 format

$$\begin{aligned} x' &= x + 0y + TX \\ y' &= 0x + y + TY \\ 1 &= 1 \end{aligned}$$

Scaling is achieved in a similar way with the matrix

$$S = \begin{bmatrix} SX & 0 & 0 \\ 0 & SY & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix permits scaling about the center of the screen only. The post-multiplication of this matrix will be the product of vector components and the corresponding scale factor.

$$\begin{aligned} x' &= SX\,x + 0y + 0 \\ y' &= 0x + SYy + 0 \end{aligned}$$

Standard notation for the elements of a matrix indicate row by the first subscript and column by the second subscript.

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Thus we can refer to the translation elements of T as t_{31} and t_{32} and similarly, the scaling elements of matrix S as s_{11} and s_{22}.

If a segment needs to be both scaled and translated, a single matrix can be found by concatenating the scale and translate matrices.

$$V' = V\,S\,T$$

$$= V \begin{bmatrix} SX & 0 & 0 \\ 0 & SY & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ TX & TY & 1 \end{bmatrix}$$

$$= V \begin{bmatrix} SX & 0 & 0 \\ 0 & SY & 0 \\ TX & TY & 1 \end{bmatrix}$$

Since matrix concatenation is non-commutative, a different matrix would result if the matrices had been concatenated in the reverse order: if translation had occurred before scaling.

Scaling, as represented by the matrix S, is always referenced to the center of the screen. When it is necessary to scale about another point, the problem is solved by translating the point to the screen origin (T_{-c}). The image is then scaled and translated back to the desired point (T_c). If the scale factors are SX and SY, and the scale center is (CX, CY), then the combination of matrices to accomplish off-center scaling (S) is:

$$T_{-c}\ S\ T_c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -CX & -CY & 1 \end{bmatrix} \begin{bmatrix} SX & 0 & 0 \\ 0 & SY & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ CX & CY & 1 \end{bmatrix}$$

$$= \begin{bmatrix} SX & 0 & 0 \\ 0 & SY & 0 \\ CX(1-SX) & CY(1-SY) & 1 \end{bmatrix}$$

Rotation about the screen origin through the angle (a) measured clockwise, requires the following matrix:

$$R = \begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

As with scaling, when an image is to be rotated about an off-center point (CX, CY), the vectors must first be translated to the screen origin (T_{-c}), rotated (R), then returned (T_c).

$$T_{-c}\ S\ T_c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -CX & -CY & 1 \end{bmatrix} \begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ CX & CY & 1 \end{bmatrix}$$

For scaling (S), rotation (R), and translation to other than screen origin (T) (in that order), the final matrix (M) is used:

where $V' = V\ M$

$M = (T_{-c}\ S\ Tc\ T_{-c}\ R\ T_c\ T)$

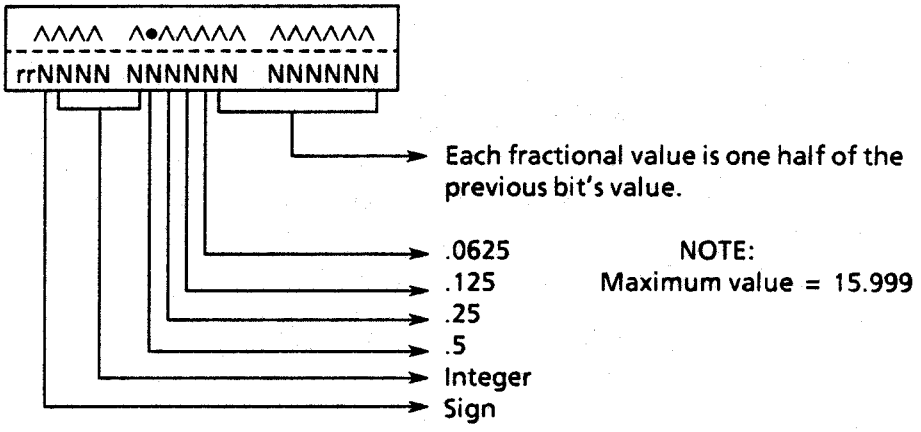
$= T_{-c}\ S\ R\ T_c\ T$

Translation and matrix commands for 2D Insert and Change Segment are as follows:

$$\begin{bmatrix} M0 & M3 & 0 \\ M1 & M4 & 0 \\ M2 & M5 & 1 \end{bmatrix}$$

M0, M1, M3, and M4 have the following format:

One sign bit, four integer bits, an implied binary point, and 11 bits for values to the right of the binary point.



M2 and M5 are 16-bit integer values (one sign bit, 15 bits for integer values). The maximum value is 32767.

NOTE:

Negative numbers are in 2's complement form.

3D Transformations

Three-dimensional transformations are similar to two-dimensional ones. As in 2D, vectors in 3D are augmented by another element in order to allow for translation.

$$V = [X\ Y\ Z\ 1]$$

Like the 2D matrices, the 3D matrices can be concatenated in any order to obtain the final resultant matrix. Matrices for 3D transformations, however, have four rows and four columns.

To translate in 3D, the following matrix is used:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ TX & TY & TZ & 1 \end{bmatrix}$$

Following through the matrix multiplication, we obtain new values for x, y, and z that are offset from their original positions.

$$\begin{aligned} V' &= V\,T \\ x' &= x + 0y + 0z + TX \\ y' &= 0x + y + 0z + TY \\ z' &= 0x + 0y + z + TZ \end{aligned}$$

Scaling about an arbitrary center (CX, CY, CZ) is accomplished by applying the following matrices:

$$T_{-c}ST_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -CX & -CY & -CZ & 1 \end{bmatrix} \begin{bmatrix} SX & 0 & 0 & 0 \\ 0 & SY & 0 & 0 \\ 0 & 0 & SZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ CX & CY & CZ & 1 \end{bmatrix}$$

where (T_{-c}) = translation to screen origin; S = scale; (T_c) = translation to selected center.

Rotation in 3D is somewhat more complicated than in 2D because there are three axes of rotation. For ease in visualization, the axes chosen are the major coordinate axes. Rotation about other axes is possible, but the matrices become complicated.

To rotate about an arbitrary center (CX, CY, CZ), the following set of matrices is used

$$T_c RT_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -CX & -CY & -CZ & 1 \end{bmatrix} R \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ CX & CY & CZ & 1 \end{bmatrix}$$

where R is one of the three rotation matrices R_x, R_y, or R_z.

The Whizzard 1645 hardware uses a left-handed coordinate system. The axes are positive X to the right, positive Y up, and positive Z into the screen. If the rotation angle (a) is measured clockwise looking back towards the origin from the positive axis, the matrices R_x, R_y, and R_z. are defined as follows:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) & 0 \\ 0 & \sin(a) & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos(a) & 0 & \sin(a) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(a) & 0 & \cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos(a) & \sin(a) & 0 & 0 \\ -\sin(a) & \cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These may be combined in any order, but the display changes when the order of combinations is altered. That is, rotating about the X-axis, then about the Z-axis, will result in a picture different from the one produced when the Z-axis rotation is performed first.

If we represent a general 3D transformation for scaling, rotation about X, Y, then Z, and translation (in that order) as

$$V' = V\,M$$

then the matrix M requires six matrix multiplications.

$$M = T_{-c}\ S\ R_x\ R_y\ R_z\ T_c\ T$$

The Whizzard 1645 system is a digital system, and imposes an upper limit on the resultant vector (V). The translation elements m_{41}, m_{42}, m_{43} may not exceed 32767 in magnitude. Other elements may not exceed a magnitude of 15.999.

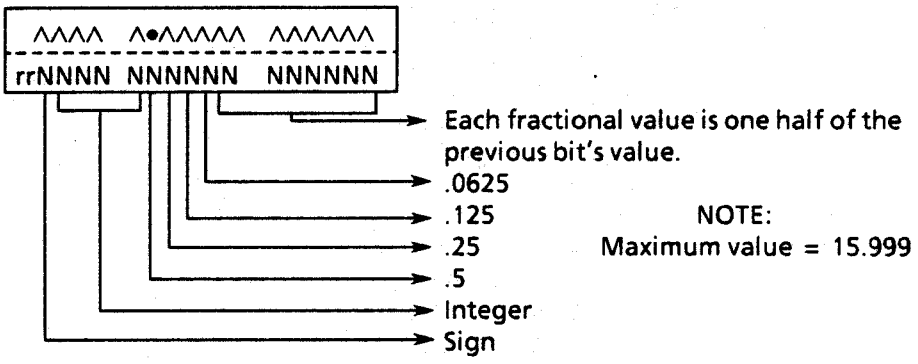
Because the Whizzard 1645 system performs orthographic projection, different values of Z with the same X and Y coordinates will appear on the screen at the same point. Therefore, a resultant Z component of V is not needed. This simplifies the computation in the hardware, since only the first two columns of the resultant matrix (M) need be used. The Whizzard 1645, however, expects the full 3x3 or 4x4 matrix so the invariant fourth column 0,0,0,1 must be supplied to complete the matrices for processing. The viewable picture is then composed of the resultant vectors that lie in the 4Kx4K space centered on the origin.

Translation and matrix commands for 3D Insert and Change Segment are as follows:

$$\begin{bmatrix} M0 & M3 & 0 & 0 \\ M1 & M4 & 0 & 0 \\ M6 & M7 & 1 & 0 \\ M2 & M5 & 0 & 1 \end{bmatrix}$$

M0, M1, M3, M4, M6, and M7 have the following format:

One sign bit, four integer bits, an implied binary point, and 11 bits for values to the right of the binary point.



M2 and M5 are 16-bit integer values (one sign bit, 15 bits for integer values). The maximum value is 32767.

NOTE:

Negative numbers are in 2's complement form.

Appendix C

VALUES OF ATTRIBUTES

Attribute/Mode	Value/Meaning	Default
Character Angle	0- 0° 1- 90° 2- 180° 3- 270°	0
Character Size	See table below	3
Character Transformation	0- Transformation Off 1- Transformation On	1
Detectability	0-Not Detectable 1-Detectable	1
Echo Level	0-5	0
Header Type	0-Basic 1-Transformable	1
Highlighting	0-Highlighting Off 1-Highlighting On	0
Linestyle	0-Solid 1-Dashed	0
Marker Size	0 to 7 (*see character size for values)*	
Size of Short Relative Vector	0 to 7	0
Visibility	0-Not visible 1-Visible	1

Size	Space Width	Line Height	Chars/ Line	Lines Screen
0	12	18	341	277
1	24	36	170	113
2	36	54	113	75
3	48	72	85	56
4	60	90	68	45
5	72	108	56	37
6	84	126	48	32
7	96	144	42	28

DEFAULT ATTRIBUTES

The default segment header, also referred to as the current header, is the header of the non-retained segment 0 at initial power-up. This header gets copied into any retained segments as soon as they are opened by the user.

The values for the default are listed below.

Contents	Default
Type	Transforms allowed; visibility on
Pick attribute	Pickable
Transformations	Enabled
Color	0
Dashing	Disabled
Origin coordinates	0
Clipping boundaries	Maximum (edge of screen)
Transformation matrix elements	Unity matrix (no rotation; no translation; scale factor of 1)

Other default attributes that apply when a reset occurs are as follows:

Attribute	Default	Values
Marker Size	3	0-7

APPENDIX D
GLOSSARY

A

Absolute coordinates - Location of points in terms of X, Y or Z distance from the established origin.
Absolute vector - Line segment with endpoint expressed in X, Y, and Z coordinates.
Access time - Time delay between request to computer for data and its availability.
Address - One or more characters identifying a data source, destination or storage site.
Alphanumeric display — Display providing strings of letters and numbers, rather than graphics.
Analog - Discrete value representing degrees of a physical quantity, not a digital representation.
Analog vector generator - Device that converts coordinate data into deflection signals for the CRT.
ANSI - American National Standards Institute.
Anti-aliasing - Software adjustment of the effects of raster pixel addressing to make diagonal or curved lines appear smooth and continuous.
Area fill processor - Device or board that shades or colors a delineated surface.
ASCII - American Standard Code for Information Interchange; standardized eight-bit data character code for computer applications.
Attribute - Any characteristic of a display item (color, line style, character font) or segment (visibility, detectability).
AVG - See analog vector generator.

B

Background - Portion of a display not changed during an application.
Background display list - Display list not requiring rapid refresh, different from foreground display list.
Baseline fill - Coloring area of graph between data line and the specified baseline.
Baud - A transmission rate equal to the number of signals sent per second; one bit per second in a binary system.
Beam position - Location of the beam following a move or draw command.
Bit map - Digital representation of an image in which bits are mapped to pixels; in color graphics, a different bit map is used for each red, green, and blue value.
Bit plane - Hardware used as a storage medium for a bit map.
Blackness - Color characteristic that scales brightness from dark to light.
Blanked area - Display space where display elements are not visible.
Blanked vector - Vector instruction that produces no visible output but changes the beam position.
Blanking - Making individual display elements or areas invisible.
Blink - Highlighting display or display segments by turning them off and on.
Buffer - Circuit that temporarily holds and subsequently transmits data to or from the host computer or another peripheral device.
Button device - Programmable input device used to transform graphics displays.

C

Clip boundary - Display space boundary, beyond which graphics display is blanked.
Clipping - Process of setting graphics display boundaries.
Color table - Table providing a limited color selection from a large theoretical color range. Also called color map.
Control dial - Graphics input device that produces a continuous range of display values. Also called valuator.
Coordinates - Number of X, Y, or Z units that give location of a point.
Current position - Beam position on the display surface prior to a move or draw instruction; also referred to a pen position for hard copy device.
Cursor - Recognizable display entity that can be moved about the display surface by a graphic input device.

D

Data Entry Keypad - Button on a keyboard device that transmits control characters or information to the program.
Data tablet - Flat graphic device used with a stylus or pick to input X and Y coordinate values.
Decluttering - Selective erasure of display items when display details are too dense.
Defaults - Preprogrammed values or settings that apply until replaced by programmed values or settings.
Detectability - Attribute that permits display items to be recognized by the display device.
Diagnostics - Software program that tests hardware operation and firmware functions, automatically reporting any malfunction.
Digital vector generator - Module used in raster displays to generate straight line of pixels between specified endpoints.
Digitize - Register visual image or real object in format that can be processed by the computer.
Digitizer - Data table that generates coordinate data from visual data read into the systems with pick or stylus.
Direct address - Operator designation of memory location for data retrieval or storage.
Direct memory access - Process by which data is transferred between the graphics display controller and host memory without passing through host CPU.
Disable - Command that ends operation of a graphics input or output device.
Display - Collection of graphics elements visible on the monitor.
Display background - Static background against which dynamic displays are projected.
Display command - Processor generated instruction to the display device.
Display device - Cathode ray tube or plotter that produces visual images from computer-generated graphics data.
Display elements - Points, line segments, characters, or markers that comprise graphics displays.
Display list - Sequence of display instructions that create, change, and refresh graphics display.
Display menu - List displayed on the monitor indicating program options available to the operator.

K

Key - Button on a keyboard device that transmits control characters or information to the program.
Keyboard - Graphics input device that allows operators to enter characters or values through typewriter keyboard augmented with special function keys.
Key-joy - Alphanumeric typewriter keyboard with attached joystick.

L

Least significant bit - Lowest order bit in a byte; bit identifying the lowest power of 2 in a byte.
LED - Light-emitting diode usually indicating device status.
Light button - Detectable display item that functions as a button device.
Light pen - Graphics input device used as a pick to identify a detectable display element.
Lightness - Beam intensity associated with a raster display color value.
Line segment - Portion of a line between two points.
Line style or type - Primitive attribute that defines lines as solid or dashed, gives dash pattern and terminators.
Locator device - Graphic input device, such as a joystick or data tablet, used to determine coordinate information.

M

Marker - Symbol that can be repeated on the display surface.
Mask - In interactive computer graphics, a program function that protects data from erasure or alteration.
Matrix - In graphics programming, the array of X, Y, and Z coefficients for calculating a geometric transformation.
Memory management - Allocation and de-allocation of memory storage locations to display list or data segments.
Menu - List of program options displayed on monitor to prompt operator.
Most significant bit - Highest order bit in a byte; value varies from system to system depending on the number of bits in a byte.
Move - Change the current beam position without producing a visible vector.

N

Node - Intersection of two or more interconnections.
Non-retained segment - Segment open when all retained segments are closed; data is lost when the non-retained segment is closed.

O

Operating system - Software regulating the compilation and execution of programs, storage assignment, and data flow between computer and peripheral.
Origin - Zero intersection of X, Y, and Z axis, from which all points are calculated.
Output primitive - Graphics entity or basic display component: point, line segment, character, marker, or text string.
Overlay - Plane of a graphics display that can be superimposed on another display.

P

Painting - Raster design technique based on illuminating specified red, green, and blue phosphors.

Parity check - Bit set to verify that a computer word has an even number or odd number of bits; used to check that a bit has not been lost in transmission.

Passive graphics - Display allowing no dynamic interaction.

Passive mode - Console setting that precludes operator interaction with the display.

Pick - Event triggered by an electronic device that generates a report identifying a detected display item and the segment containing it.

Pick device - An event device, such as a light pen or a locator device with a comparator, used to identify detectable display items.

Pick identifier or label - Name associated with a detectable display item.

Pixel - Picture element; minimum raster display element, represented as a point with a specified color or intensity level.

Plotter - Computer-controlled pen device that produces hard copy of a display on paper or electrostatic surface.

Polygon fill - Coloring or cross-hatching of a closed, multi-sided, program-defined surface.

Primitive - Basic display element; point, segment, alphanumeric character, or marker.

Primitive attribute - Visual characteristics of an output primitive, such as character size, line style or blink rate.

Protocol - Data exchange conventions governing communication between units of a computer system.

R

Raster - Rectangular pixel matrix permitting dynamic color displays.

Raster count - Number of scan lines in a raster display.

Raster scan - Sweep of the electron beam across raster pixel matrix to excite phosphors and create the visual display.

Raster unit - Distance between the midpoints of two adjacent pixels.

Read - Retrieve data from memory or graphics input device; process halts graphics operations.

Read-only memory - Memory that can furnish data but not receive or alter any; generally stores a system's operating instruction set.

Refresh - Process of renewing the display on the surface of a CRT.

Refresh cycle - Complete scan of display surface by electron beam to maintain display illumination.

Refresh rate - Time needed to complete one refresh cycle, measured in Hz.

Relative draw - Electron beam displacement in terms of X, Y, and Z distances to draw a visible vector on the display surface.

Relative move - Electron beam displacement in X, Y, and Z distances without leaving a visible trace on the display surface.

Relative vector - Vector with endpoint specified in terms of distance from current position, rather than in terms of absolute X, Y and Z coordinates.

Repaint - Refresh a display surface with an updated display.

RGB color - Color described in terms of its red, green, and blue intensity levels.

Rotate - Transform a display or display item by revolving it around a specific axis or center point.
Rubber-banding - Attaching a fixed point on the display to a cursor with a line that appears to stretch and contract like a rubber band as the cursor moves.

S

Sample - Query a graphic device for coordinate data or operating status.
Segment - Portion of the display list that defines a display item, has a header to set or change attributes and a recallable name.
Segment attribute - Segment display characteristic such as detectability or visibility.
Segmentation - Division of a display into parts that can be recalled or transformed individually.
Selective erase - Deletion of specified portions of a display without affecting other portions.
Short relative vector - SRV; vector of a limited length with end point identified in terms of X, Y, and Z distances from the current beam position.
Subroutine - Series of computer instructions that can be used repeatedly in one or more program applications.
Symbol - Marker displayed as line segment terminators, as in line graph.

T

Tablet - Data tablet or digitizer; graphics input device that generated coordinate data from visual data input through a puck or stylus.
Text string - Series of alphanumeric characters.
Transformation - Geometric alteration of a graphics display, such as scaling, translation or rotation.
Transformation matrix - Matrix of multiplication factors for scaling or rotation operations.
Translate - Shift a display item across the display surface to a new location.

U

USART - Universal synchronous-asynchronous receive-transmit chip that controls serial data transmission.

V

Valuator device - Graphic input device, such as a control dial, that inputs graduated values within a user-defined range.
Viewpoint - Subset of device space.
Visibility - Dynamic segment attribute that determines whether or not a segment appears on the display surface.

W

Whiteness - Color characteristic registered as percentage of additive light intensity.
Write - Make permanent or temporary data record in memory or storage medium.
Write protect - Software feature that prevents erasure during refresh cycle.

INDEX

T

V

W

Additional copies of this and other Megatek manuals are available from:

Megatek Corporation
9605 Scranton Road
San Diego, CA 92121

Attention: Manager, Office Services
Telephone: (619) 455-5590

For more information on the 1645, you may wish to order the following additional manuals:

0251-0012 Graphics Protocol Manual
0252-0021 Wand Reference Manual
0252-0024 Wand User's Guide