H-79-0348 [REV A]

Graphic7™

COMPUTER GRAPHICS DISPLAY SYSTEM

GRAPHIC CONTROL PROGRAM ENHANCED [GCP+] PROGRAMMER'S REFERENCE MANUAL

Information Products Division
Federal Systems Group
SANDERS
DANIEL WEBSTER HIGHWAY, SOUTH—NASHUA, NEW HAMPSHIRE 03061

Sanders Associates, Inc., reserves the right to make corrections or alterations to this manual at any time without notice.

First Edition - February 1980

Reprint - April 1980

Reprint - August 1980

Reprint - September 1980

Reprint - October 1980

Reprint - January 1981

Reprint - March 1981

Reprint - June 1981

Reprint - November 1981

TABLE OF CONTENTS

Section 1

GRAPHIC 7 System Description

TABLE OF CONTENTS (Cont)

Section 2

Operating Modes

Section 3

GRAPHIC 7 Instructions

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1
GRAPHIC 7 SYSTEM DESCRIPTION

1.1 INTRODUCTION

Sanders GRAPHIC 7 is an intelligent interactive graphic display terminal that comes ready to use. Its standard serial and parallel interfaces make it immediately compatible with most computers. The display image is created by using CRT stroke-writing techniques to ensure the maximum in brightness, resolution, and response time.

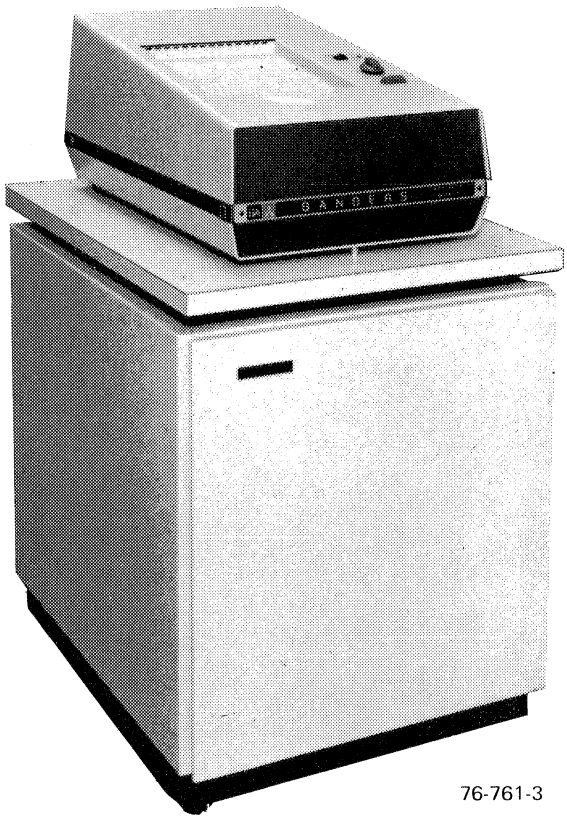
Two high-performance microprocessors in the GRAPHIC 7 handle the separate tasks of interaction and display imaging. A Graphic Control Program (GCP+) provided in read-only memory handles communications with the host computer, controls data entry devices, manages the display image refresh, and performs other tasks normally done by a host computer. This program plus built-in test routines minimize the amount of software required for the host computer. An optional FORTRAN Graphic Support Software Program (FSP) is also available which further reduces software requirements for the host computer.

The adaptability, flexibility, and high performance of the GRAPHIC 7 make it ideally suited for many uses. Typical applications include:

- Computer-aided design and manufacturing
- Simulation and training
- Resource and information management
- Command and control
- Air traffic control
- Data reduction

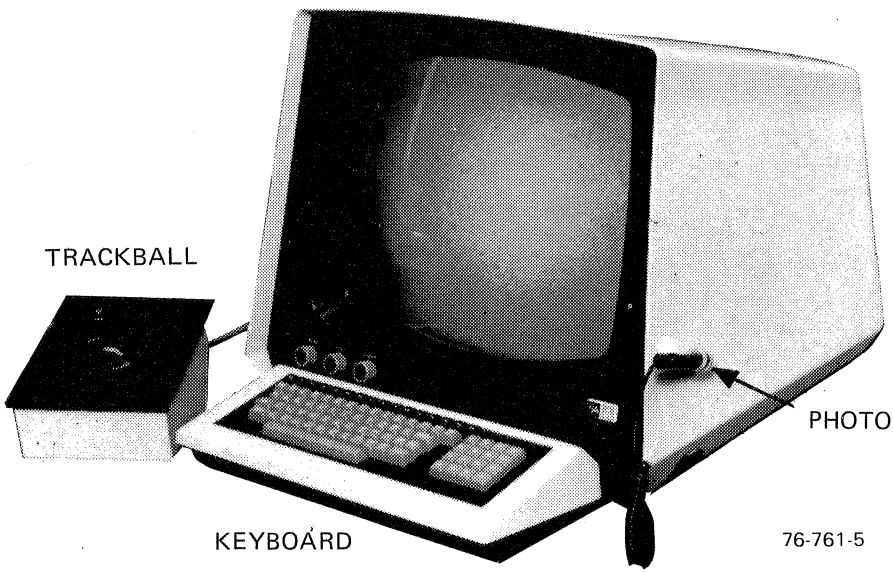
76-734-4

TERMINAL CONTROLLER
(WITH OPTIONAL CABINET)



76-761-3

HARD COPY UNIT



76-761-5

GA-77-419-01

Figure 1-1 GRAPHIC 7 System Components.

1.2 EQUIPMENT DESCRIPTION

A basic GRAPHIC 7 comprises a terminal controller and a 21-inch CRT display indicator. Optional input devices include keyboards, a PHOTOPEN,® a trackball, a forcestick, and a data tablet. A hardcopy unit is also available as a optional output device. Figure 1-1 shows different components of the GRAPHIC 7. Representative configurations of these components are shown in figure 1-2.

1.2.1 TERMINAL CONTROLLER

The terminal controller is a standard rack-mountable card cage containing 17 card slots and one power supply assembly. As shown in figure 1-3, cards installed in the slots are interconnected by means of a processor bus and a graphic bus. The cards that are interconnected via that processor bus handle communications with the host computer and direct operations performed by cards on the graphic bus. The cards on the graphic bus control the drawing of images on associated display indicators. The following paragraphs describe the function of each type of card that may be installed in the card slots of the terminal controller.

1.2.1.1 Read/Write Memory

The basic configuration of a GRAPHIC 7 terminal controller includes one card of random access read/write memory capable of storing 8192 16-bit words. Two additional cards may be added to provide a total of 24,576 words of read/write memory. Locations in the read/write memory are assigned addresses 000000_8 through 137777_8 and are accessed by means of a 16-bit address on the processor bus. The 16-bit address can be used to access the location of a word (16 bits) or of an individial byte (8 bits) as required. Refer to figure 1-4 for a GRAPHIC 7 system memory map.

A large read/write memory is available as an optional replacement for the 8K memory. Each large memory card is capable of storing $65,536_{10}$ (64K) sixteen bit words or 128K separately addressable 8-bit bytes. A maximum of two large memory cards can be installed in a GRAPHIC-7 system for a total of 128K 16-bit words of memory. (The large read/write memory card is also available in 16K and 32K word sizes.)

PHOTOPEN is a trademark of Sanders Associates, Inc.

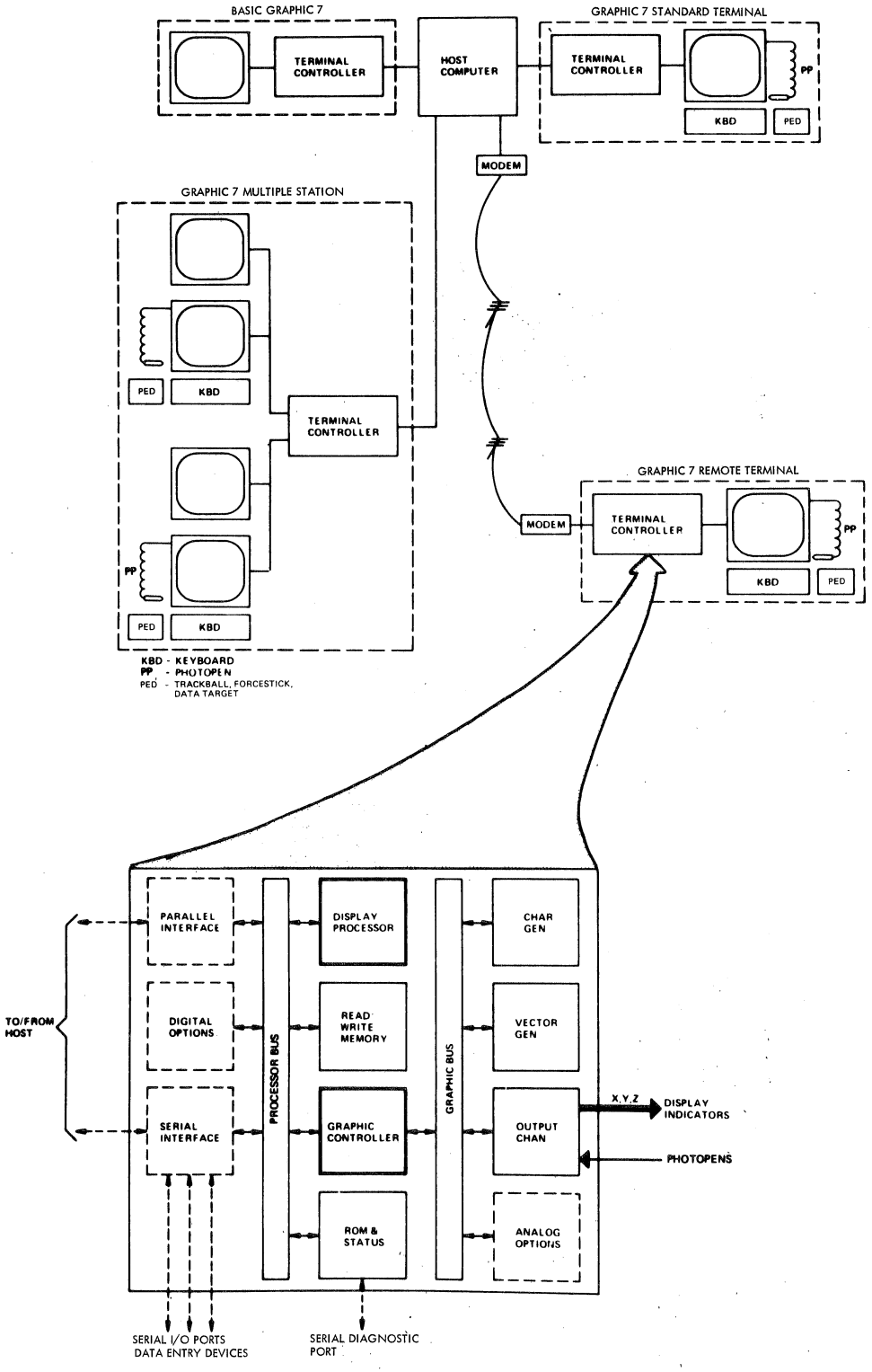


Figure 1-2 Representative GRAPHIC 7 System Configurations.

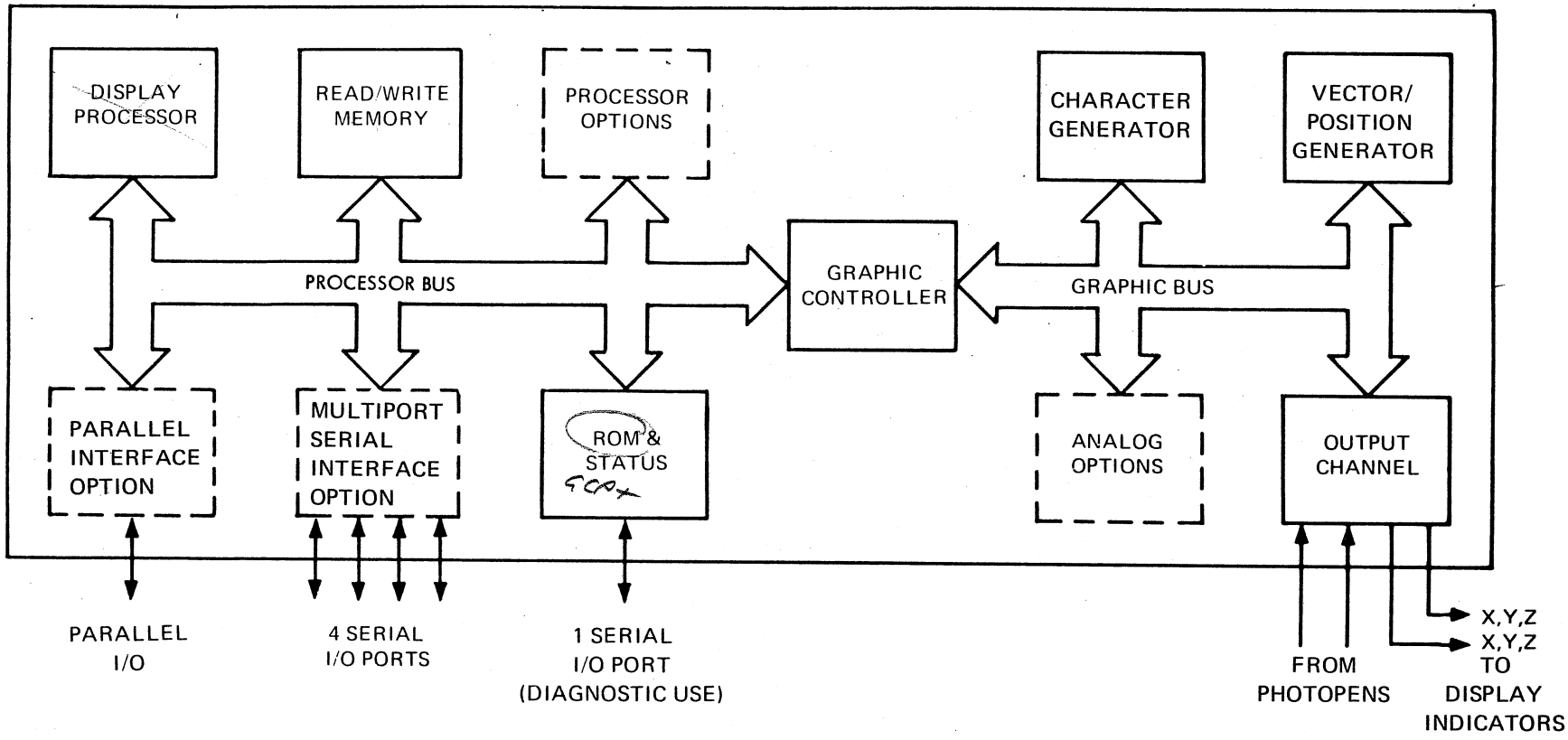
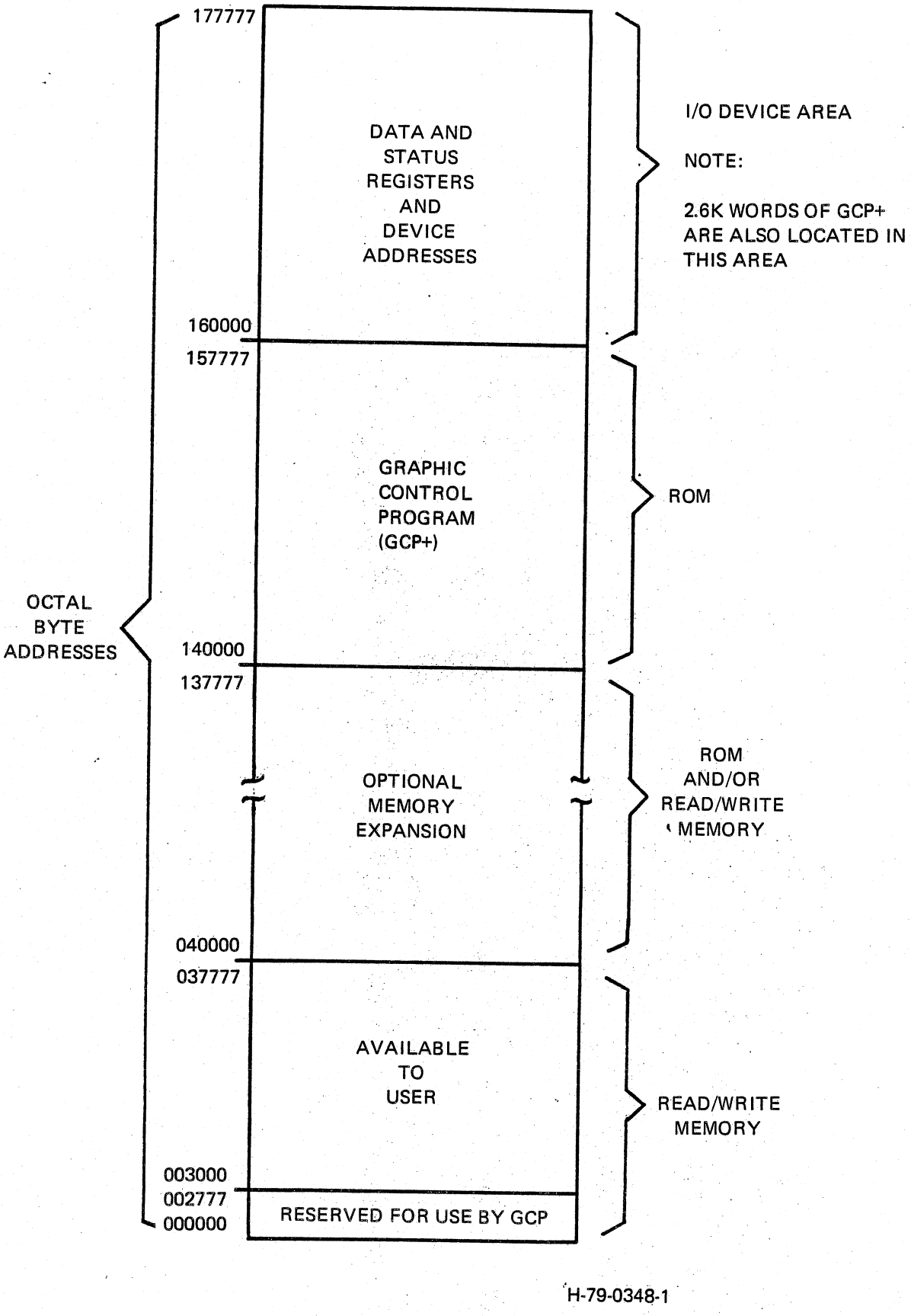


Figure 1-3 GRAPHIC 7 Terminal Controller Functional Block Diagram.



H-79-0348-1

Figure 1-4 GRAPHIC 7 System Memory Map.

NOTE

User refresh programs will not execute in RAM memory in the 24K to 32K area (140000-177777). This area is reserved for Sanders' display processor option software. The option software is loaded from the expansion module or is down-loaded from the host.

1.2.1.2 Parallel Interface

The parallel interface card is an option intended for installations where the GRAPHIC 7 is located in proximity to the host computer. It allows high-speed host/GRAPHIC 7 communications with handshaking and can be operated in a DMA mode. If a parallel interface card is installed in the terminal controller, GCP+ assumes that it is connected to the host computer. Therefore, if serial communication with the host computer is desired, a parallel interface card cannot be connected to the processor bus.

NOTE

Normally, if a parallel interface port is used, a single parallel interface card (for the host computer) is installed in the terminal controller. For special applications, however, up to four parallel interface cards may be installed.

1.2.1.3 Multiport Serial Interface

The multiport serial interface card contains four serial interface ports that operate in a serial asynchronous mode using RS-232C or TTL voltage levels with standard transmission rates up to 9600 baud. Additionally, the first port can be operated as a full RS-232C asynchronous interface at transmission rates greater than 9600 baud. For GCP+ applications, the maximum transmission rate supported is 9600 baud. Normally, the host computer is connected to the first port, which is compatible with the standard communication and terminal interfaces supplied by most computer manufacturers. The remaining three ports on the card are used for peripheral devices.

Two multiport serial interface cards may be installed in a terminal controller to handle additional peripheral devices if required. Normal device assignments for each port are listed in Appendix A.

1.2.1.4 Display Processor

The display processor card is a general purpose digital computer that runs the GCP+ and acts as master control for all devices connected to the processor bus. It contains multiple high-speed general-purpose registers that can be used as accumulators, pointers, index registers, or auto-indexing pointers in auto-increment and auto-decrement modes. Functions performed by the display processor card include system initialization, interface handling, local data editing, and local generation of simple display images.

Instructions used for the display processor emulate the instruction set for the PDP-11® series of minicomputers manufactured by Digital Equipment Corporation (DEC™). They are fetched either from the GCP+ in read-only memory on the ROM and status logic card (paragraph 1.2.1.5) or from the read/write memory (paragraph 1.2.1.1).

PDP and DEC are registered trademarks of Digital Equipment Corporation.

1.2.1.5 ROM and Status Logic

The ROM and status logic card contains the read-only memory in which the GCP+ used to control the display processor is stored (refer to figure 1-4). Also contained on the card are display status and interrupt logic circuits plus a serial interface port to which a teletypewriter may be connected for diagnostic purposes.

The standard read-only memory provided on the ROM and status logic card contains the GCP+ program. The GCP+ is approximately 6.6K words (16 bits). Like read/write memory, read-only memory may be accessed to retrieve either 16-bit words or individual 8-bit bytes.

1.2.1.6 Graphic Controller

The graphic controller card is a microcontroller that controls generation of the image on the display indicator. Instructions used by the graphic controller are fetched via the processor bus from either the read/write or the read-only memory. These instructions, referred to as graphic controller instructions, are described in detail in Section 3. The complete series of sequential instructions that defines any particular display image is referred to as a refresh file.

The graphic controller may be considered as a device on the processor bus of the terminal controller. It contains its own set of registers that maintain intruction addresses, control fetch opeations, and perform any branching that may be specified by non-graphic instructions. It also calculates relative data when required, loads data into appropriate registers, and initiates execution of refresh file instructions.

Status bits of the graphic controller are maintained by circuits on the ROM and status logic card (paragraph 1.2.1.5). These bits plus the graphic controller registers are accessible to the display processor (paragraph 1.2.1.4) which maintains control over the entire terminal controller.

1.2.1.7 Vector/Position Generator

The vector/position generator comprises three separate circuit cards on the graphic bus: a ramp generator and two digital-to-analog (D/A) converter cards. These cards operate together to produce CRT beam-positioning voltages as defined by digital X- and Y-coordinate instructions. They also generate unblanking signals to enable vectors to be drawn on the associated display indicators.

One D/A converter is used to address X coordinates on the face of the display indicator CRT while the second is used to address Y coordinates. Each D/A converter can address 2048 coordinates of which 1024 fall within the displayable area. The CRT beam is automatically blanked whenever it is moved to a coordinate that lies outside the displayable area. Refer to figure 1-5.

Beam positioning and vector drawing instructions may specify either absolute or relative data. Absolute data specifies the locations of particular coordinates whereas relative data specifies locations in terms of the distance moved from the previous location.

1.2.1.8 Character Generator

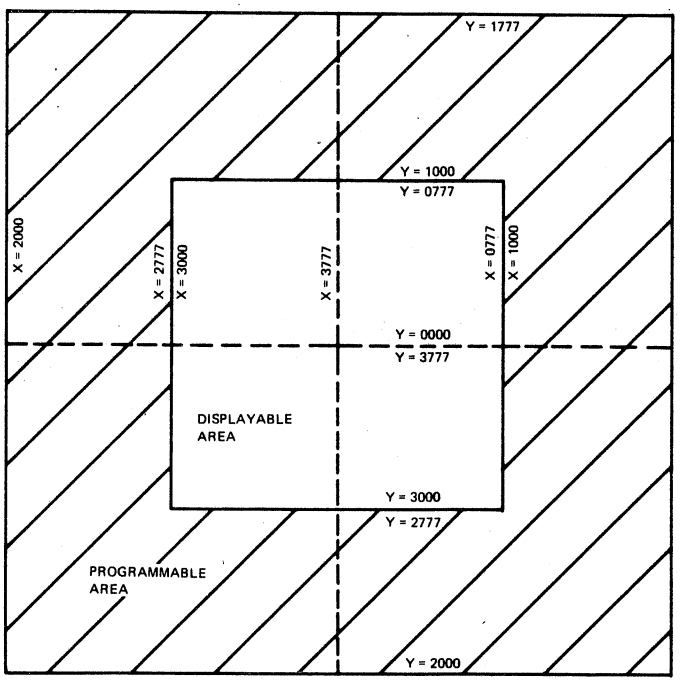
The character generator card contains read-only memories that store information for drawing characters in accordance with instructions received on the graphic bus from the graphic controller. The basic set of characters supplied with the character generator read-only memories is a standard set of 96 ASCII characters. When the ASCII code corresponding to the desired character is applied to the read-only memories, the character is drawn at the position determined by the vector/position generator (paragraph 1.2.1.7).

As determined by instructions from the graphic controller, characters of four different sizes can be generated or characters can be made to blink. Characters may also be rotated 90 degrees counterclockwise to accommodate vertical writing requirements.

Space is provided on the character generator card for additional read-only memories so that characters in addition to the basic set of 96 can be generated. Read-only memories for six groups of 16 characters can be added to provide a total of up to 192 standard and special characters that can be produced by the character generator.

1.2.1.9 Ramp/Conic Generator

The ramp/conic generator is an optional card that may be connected to the graphic bus. It generates X-, Y-, and Z-axis waveforms that enable ellipses, or 90-degree segments thereof to be drawn on the display indicator. Ellipses may be centered at any addressable location on or off the CRT screen and are oriented so that their major and minor axes lie in parallel with the X and Y areas of the display indicator. The lengths of the axes are determined by instructions that specify



GA-77-419-05

NOTE

Coordinate designations are in octal format.

Figure 1-5 CRT Programmable vs. Displayable Areas.

semimajor and semiminor axis lengths for each ellipse. Semimajor and semiminor axis lengths are independently programmable from zero to half-screen.

Program control also permits any combination of 90-degree segments of an ellipse to be displayed. These segments are defined by the ellipse axes.

1.2.1.10 Output Channel

The output channel card contains four Z-axis output channels. Thus, up to four display indicators and/or hard copy units can be driven by one output channel card. Program control permits the four Z-axis outputs to be blanked and unblanked selectively so that the same or different images can be sent to each of the ouput devices as required.

Intensity control circuits and a blink oscillator are also located on the output channel card. Three bits under program control permit eight different intensity levels to be selected for the Z-axis outputs. A single bit enables or disables the blink oscillator.

Additional functions performed by the output channel card include providing timing signals to the graphic controller and processing signals from PHOTOPENs. In normal system installations, each of the output devices connected to an output channel card can be located up to 50 cable-feet away with no degradation in performance.

1.2.1.11 2-D Coordinate Converter

A 2-D coordinate converter card is available as an option for the GRAPHIC 7. This option permits components of a displayed image to be rotated and/or translated on the CRT screen as determined by software instructions. Details concerning applications and programming of the 2-D coordinate converter are contained in Sanders publication H-78-0061.

1.2.1.12 EPROM Expansion Module

As options are added to the GRAPHIC 7, the additional software required to handle the options will be stored on the model 7750 expansion module (EM).

The expansion module may contain up to 32K 16-bit words of non-volatile read-only memory (EPROMS). The data may be loaded from the EM automatically by pressing the SYSTEM button or when so instructed by the host, depending on the options stored.

1.2.1.13 2-D/3-D Coordinate Converter

The Model 5753 2-D/3-D coordinate converter converts a Sanders graphic display into a three dimensional display capable of independent dynamic manipulation of objects in apparent space. Among the functions provided by the Model 5753 are translation, scaling, rotation, windowing, independent display coordinate mapping, perspective, and zooming with perspective.

The perspective feature is especially useful for realistic viewing of an object. Utilizing perspective, the location of the viewer is defined relative to the image space, and all lines and objects within the image space are then viewed at the proper perspective for that location. The view may be completely orthographic if the viewer does not wish to use the perspective feature.

Objects can be defined within a 64K (X), 64K (Y), by 32K (Z) image space and presented on a 1K by 1K screen or any portion thereof. Translations can be made within the limits of the image space and scaling range is 64 to 1. Rotation can be provided about any axis.

3-D windowing, in conjunction with independent screen coordinate mapping, allows the presentation of any data within a software definable X, Y, Z image space to be presented on the full screen or any portion of the screen. Zooming is accommodated by scaling and changing the user's apparent perspective viewpoint.

Alphanumeric data can be moved about the screen with vector defined data without scaling and rotation.

The 5753 provides for both homogeneous and non-homogeneous matrix operation. Also, transformations of 2-D images can be accomplished including translation, rotation, scaling, and windowing.

1.2.1.14 Floating-Point Converter

The model 5744 floating-point converter option transforms incoming floating point binary numbers into displayable numbers. The displayable numbers may be in any of sixteen formats selected by the host. The bi-directional converter also converts the displayed numbers into floating-point binary for transmission back to the host.

The floating-point converter saves host computer time and storage resources by performing these conversions within the graphic terminal. It allows data to be

transmitted to and from the host in its most compact form and frees the host programmer from the conversion programming task.

The floating-point converter can perform more than 500 conversions per second, which allows it to be used in high data-rate applications resulting in significant off-loading of the host computer.

1.2.2 DISPLAY INDICATOR

Up to four display indicators (monochrome or 4-color) can be connected to one GRAPHIC-7 terminal controller. The standard display indicator is a self-contained, large screen, high speed, X-Y-Z CRT indicator. The CRT is rectangular with a diagonal measurement of 21 inches. With the exception of the CRT, the unit consists entirely of solid-state circuits. It is available in an enclosure as a desk-top unit or with slide mountings as a 24-inch rack-mountable unit. A 23-inch circular desk-top unit is also available as an option.

The CRT of the standard display indicator provides a normal display area of 12 by 12 inches which can be modified to 12 by 16 inches for special applications. The display area of the optional unit is a circle 20 inches in diameter. Notable features of the display indicators are low power consumption, fast writing speeds, long life, and high quality viewing characteristics.

Locations on the CRT screen are specified in terms of a matrix containing 2048 coordinates in the X dimension and 2048 coordinates in the Y dimension. Two's complement notation is used to designate the coordinates with location 0,0 being defined as the center of the CRT screen. Of the 2048 by 2048 addressable locations, the displayable area comprises the field of 1024 by 1024 coordinates centered about the middle of the CRT screen. Refer to figure 1-5.

1.2.3 INPUT DEVICES

Optional data input devices for the GRAPHIC 7 give the operator two-way interaction with the display and processing system. Input devices available include two types of keyboards, a PHOTOPEN, a trackball, a forcestick, and a data tablet. The GCP+ in firmware can support up to two keyboards, two PHOTOPENS, and two position entry devices (trackball, forcestick, or data tablet). In addition to the foregoing, a teletypewriter or paper tape reader can be connected to the GRAPHIC 7 for the input of maintenance data.

1.2.3.1 Keyboards

Standard keyboards available for the GRAPHIC 7 are the Model 5783 and Model 5784 keyboards. The keyboards contain a main block of alphaumeric keys plus a matrix and a row of function keys.

The Model 5783 keyboard offers an alphanumeric block of 58 keys. These keys generate standard seven-bit ASCII codes with an eighth (MSB) bit always set to 1. The alphabetic keys generate both upper and lower case codes. A four-by-four matrix of function keys is located to the right of the alphanumeric block and a row of 16 function keys is located immediately above the alphanumeric block. Each function key generates a single eight-bit octal code from 000 to 037.

An added feature of the Model 5784 keyboard is that each function key contains a LED that can be lighted or turned off as required under program control. The Model 5784 also has provisions for additional keys to the basic board. These keys are for future expansion and are located on both sides of the space bar.

The keyboards operate at a rate of 9600 baud and interface to the terminal controller via ports on the multiport serial interface card. Refer to Appendix A for layouts of the two keyboards and the specific codes generated by each key.

1.2.3.2 PHOTOPEN

The PHOTOPÉN is a small hand-held device that detects light from data displayed on the CRT of a display indicator. Detected light is converted into an electrical impulse to identify the specific data at which the PHOTOPEN is pointed. The excellent resolving capability of the PHOTOPEN enables individual characters and even displayed points of light to be distinguished.

A switch in the PHOTOPEN is actuated when the PHOTOPEN is pressed against the CRT screen. Actuation of this switch causes the data sensed by the PHOTOPEN to be processed as determined by program control. GCP+ provided with the GRAPHIC 7 can support up to two PHOTOPENs.

1.2.3.3 Trackball, Forcestick, and Data Tablet

The trackball, forcestick, and data tablet are referred to as PEDs (position entry devices). These devices are used as determined by program control to move a cursor and/or data displayed on the CRT screen. Movement initiated by the trackball is proportional to the speed and direction in which the trackball is rolled. Movement initiated by the forcestick is proportional to the direction and force with

which the forcestick is deflected. Movement initiated by a data tablet is proportional to the speed and direction in which the data tablet pen is moved along the data tablet surface. PEDs are connected to the system via ports on the multiport serial interface card(s) in the terminal controller.

1.2.3.4 Maintenance Data Input Devices

A teletypewriter and/or a paper tape reader can be connected to the GRAPHIC 7 to input data for maintenance purposes. The teletypewriter is normally connected to a port on the ROM and status card in the terminal controller while the paper tape reader is connected to one of the ports on a multiport serial interface card. The teletypewriter serves basically as a troubleshooting aid. The paper tape reader is used to load special user or diagnostic programs into the GRAPHIC 7.

1.2.4 OUTPUT DEVICES

The standard output device for the GRAPHIC 7 is the CRT display indicator described in paragraph 1.2.2. A hard copy unit is available as an optional output device. Using the same signals that go to a standard display indicator, the hard copy unit can produce a duplicate on paper of any static image displayed on the CRT of the display indicator. Operation of the hard copy unit is controlled manually or by program control.

A hard copy multiplex switch is available as an optional device. The multiplex switch is capable of interfacing up to four GRAPHIC 7 units to a single hard copy unit. The multiplex switch generates copies on a first come, first served basis and requires no additional interfacing.

SECTION 2
OPERATING MODES

2.1 GENERAL

The GRAPHIC 7 system can be operated in either the local or the system mode. In the local mode, the GRAPHIC 7 operates as a stand-alone system; in the system mode, the GRAPHIC 7 operates on-line to the host computer. Initialization in either mode causes built-in diagnostic routines to be performed automatically.

2.2 LOCAL MODE

After primary power has been applied to the GRAPHIC 7, the system may be initialized in the local mode by pressing the LOCAL pushbutton switch on the front of the terminal controller. Pressing this switch causes a verification test pattern to be displayed on each of the associated display indicators, causes the built-in diagnostic routines to be performed, and enables local mode commands to be executed. The following paragraphs discuss these operations as they relate to software.

NOTE

When the LOCAL switch is pressed, the built-in memory diagnostic exercises the complete memory system. For systems containing more than 32K of memory, it may take several seconds before the terminal verification pattern is displayed. As part of the memory diagnostic, the memory configuration installed in the terminal controller is saved and can be examined if desired. Address 736 contains the RAM configuration word and address 750 contains the ROM configuration word.

The RAM and ROM configuration words are defined as follows:

RAM CONFIGURATION WORD

15 14 13 12 11 10 9 8	7	6	5 4 3 2 1 0
LARGE MEMORY SYSTEM	0	0	SMALL MEMORY SYST

Large Memory System (64K cards)

B_{15}	B_{14}	B_{13}	B_{12}	B_{11}	B_{10}	B_9	B_8	(16K word 1/2 banks)
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØK
Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	16K
Ø	Ø	Ø	Ø	Ø	Ø	1	1	32K
Ø	Ø	Ø	Ø	Ø	1	1	1	48K
Ø	Ø	Ø	Ø	1	1	1	1	64K
Ø	Ø	Ø	1	1	1	1	1	80K
Ø	Ø	1	1	1	1	1	1	96K
Ø	1	1	1	1	1	1	1	112K
1	1	1	1	1	1	1	1	128K
Banks		3		2		1		0

Small Memory System (8K cards)

B_5	B_4	B_3	B_2	B_1	B_0	(4K word pages)
Ø	Ø	Ø	Ø	Ø	Ø	0K
Ø	Ø	Ø	Ø	1	1	8K
Ø	Ø	1	1	1	1	16K
Ø	1	1	1	1	1	20K
1	1	1	1	1	1	24K
Pages	5	4	3	2	1	0

ROM CONFIGURATION WORD

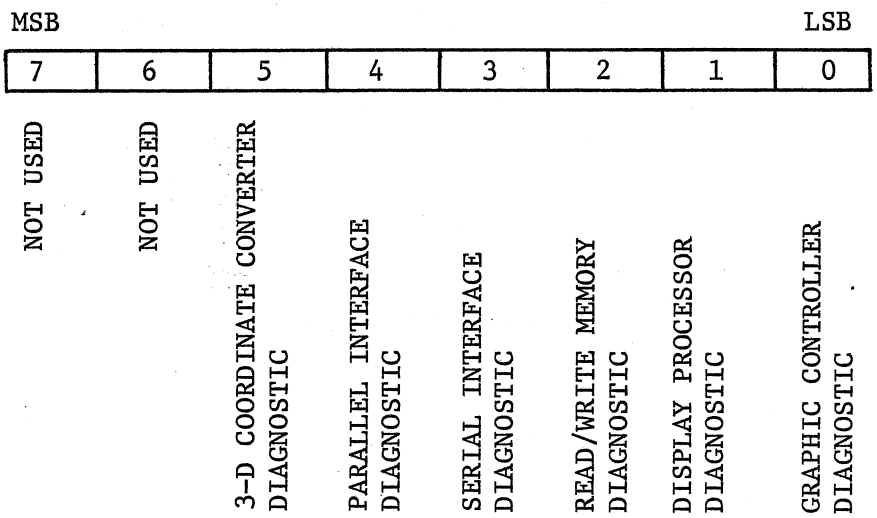
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ROM		Ø	Ø	Ø	Ø

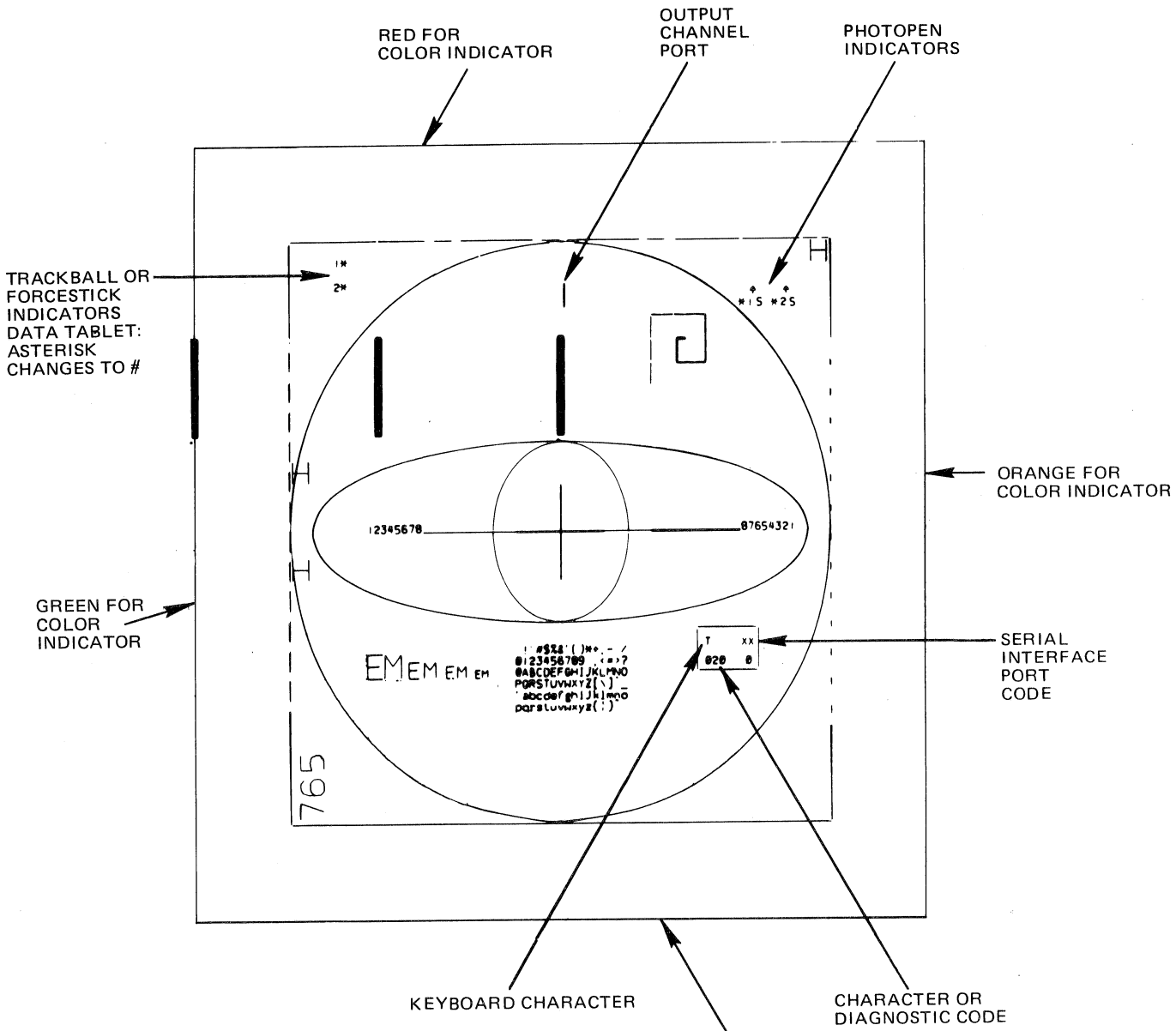
B_5	B_4	
Ø	Ø	No optional ROM
1	Ø	4K optional ROM
1	1	8K optional ROM

2.2.1 VERIFICATION TEST PATTERN AND DIAGNOSTICS

Figure 2-1 is the verification test pattern that is displayed on each display indicator when the GRAPHIC 7 is initialized in the local mode. This pattern remains displayed until terminated by the proper command (paragraph 2.2.2) or until a period of 45 minutes has elapsed since an operation affecting the pattern was last performed.

Components of the verification test pattern that are primarily associated with software and the operation of peripheral devices are identified in figure 2-1. When the system is first initialized in the local mode, "XX" appears in the small box in the lower right portion of the pattern. The "XX" indicates that the code appearing in the same box contains the results of the built-in diagnostic routines that were automatically performed. The diagnostic code is a three-digit octal representation of an eight-bit binary code that indicates the results of each diagnostic routine. Bits in the binary code are assigned as follows:





NOTE:

This figure illustrates the verification test pattern that is generated when the ramp/conic generator option is installed in the terminal controller. If a ramp/conic generator card is not installed, each circle and ellipse will be displayed as four straight lines. If a 2-D coordinate converter card is installed in the terminal controller, all information contained within the ellipse will be rotated, translated and displayed at the top left of the display.

GA-77-419-04

Figure 2-1 Verification Test Pattern.

When a diagnostic routine detects a malfunction, the corresponding bit in the error code is set to a 1; if no malfunction is detected, the bit is set to a 0. The octal code displayed in the verification test pattern then indicates the results of all the diagnostic tests. For example, 000 indicates all tests passed, 002 indicates the display processor diagnostic test failed, 030 indicates the serial and the parallel interface diagnostic tests failed, and 077 indicates that all diagnostic tests failed.

An additional routine performed whenever the GRAPHIC 7 is initialized in the local mode is a checksum calculation based on all GCP+ stored in read only memory. The result, which is deposited in memory 5ØØ (octal), can be examined as described in paragraph 2.2.2.1 and compared with the correct value contained in the GCP+ program listing.

As soon as any input is received by the terminal controller via a serial interface port, the "XX" in the small box is replaced by a code that indicates the port to which the input device is connected. Codes associated with each serial interface port are shown in table 2-1.

When the serial interface port designation is displayed in the small box, the three digit octal code in the box indicates the code last transmitted to the terminal controller. Additionally, if the code represents a displayable character, the character appears in the upper left corner of the box. If the code does not represent a displayable character, the upper left corner of the box is blank. In systems using SI (shift in) and SO (shift out) codes to identify characters in an extended set, the SI character is displayed over the left hand digit of the code and the SO character is displayed over the right hand digit.

The numeral in the upper center of the verification test pattern indicates the port on the output channel card to which the Z axis of the display indicator is connected. Connectors J5 through J8 on the output channel card correspond to indicators 1 through 4, respectively.

TABLE 2-1

SERIAL INTERFACE PORT CODES

Code	Serial Interface Port	Device	Associated Connector
F1	3	Keyboard (with function keys)	J5 on multiport serial interface card no. 1
F2	7	Keyboard (with function keys)	J5 on multiport serial interface card no. 2
TT	TTY	Teletypewriter	J2 on ROM and status card
S1, S5	1 or 5	Any	J2 or J3 on multiport serial interface card no. 1 or no. 2
HC	5	Hardcopy	J3 on multiport serial interface card no. 2

NOTE

No indicator code is provided for ports 4 or 8. These ports are normally used for PEDs which have separate indicators on the test pattern.

Trackball (or forcestick) indicators appear in the upper left corner of the verification test pattern. The "1*" indicator is associated with the device normally connected to serial interface port 4 (J6 on multiport serial interface card no. 1) while the "2*" is associated with the device normally connected to serial interface port 8 (J6 on multiport serial interface card no. 2). These indicators are always displayed on the test pattern regardless of whether or not a trackball or forcestick is connected to the system. If a trackball or forcestick is connected to port 4 or 8, it can be manipulated to move its associated indicator about the screen of the CRT as desired. (See paragraph 2.2.1.2 for data tablet.)

PHOTOPEN indicators are displayed in the upper right corner of the verification test pattern. The "*1S" with an arrow is associated with a PHOTOPEN connected to the PPN1 connector on the front of the terminal controller. The "*2S" with an arrow is associated with a PHOTOPEN connected to the PPN2 connector. Like the trackball/forcestick indicators, the PHOTOPEN indicators appear on the verification test pattern whether or not PHOTOPENs are connected to the system.

If a PHOTOPEN is connected to the system, its associated indicator responds as light from various data items is sensed by the PHOTOPEN. Whenever an item of data is sensed, the sensed point is intensified and the indicator moves so that the arrow points to the location at which the data item ends. Alphanumeric data is normally stored with two characters per data item. Therefore, the arrow always points to the end of the second character in a pair. If the PHOTOPEN is also pointed at the character, an asterisk is added to the indicator. When the PHOTOPEN is pointed at the first character in a pair or at a non-character data item, the asterisk is removed from the indicator.

The "S" in each indicator provides an indication of PHOTOPEN switch operation. When the switch is actuated by pressing the PHOTOPEN against the CRT screen, the "S" is removed from the indicator. A second actuation of the switch causes the "S" to reappear with the indicator.

NOTE

The complete character set is displayed at the bottom center of the terminal verification pattern. In this area all characters are insensitive to PHOTOPEN strikes.

2.2.1.1 Hardcopy Generation

A hardcopy of the terminal verification pattern can be generated by pressing function key FØ (refer to Appendix A for function key location). When this key is pressed, a HC will appear in the serial interface port code. The HC indicates that a hardcopy request has been initiated. The successful generation of a hardcopy is indicated by the display of the characters S5 in the serial interface port code and the number 000 in the character or diagnostic code section of the terminal verification pattern. If the hardcopy request is unsuccessful, the characters HC remain displayed in the serial interface port code.

If a hardcopy multiplex switch is connected to the terminal controller, the successful generation of a hardcopy is indicated as previously described. If the hardcopy request is unsuccessful, the characters S5 are displayed in the serial interface port code and the numbers 377 are displayed in the character or diagnostic code section of the terminal verification pattern. The 377 code indicates the hardcopy unit is either off-line or busy.

NOTE

The generation of hardcopies takes approximately 10 seconds. To generate hardcopies remotely with function key FØ requires that a serial cable be connected between port 5 (multiport serial interface No. 2) and the hardcopy. The X, Y, Z cables must also be connected between the output channel card and the hardcopy unit or hardcopy multiplexer unit.

2.2.1.2 Data Tablet Testing

The data tablet can be tested by pressing function key F1. This causes the 1* and 2* trackball/forcestick indicators to change to 1# and 2#. The 1# and 2# symbols indicate that all messages received via ports 4 and 8 are in data tablet format. (Data tablet messages consist of 10 character messages, whereas the trackball and forcestick generate 2-character messages.) When the data tablet pen switch is pressed and the pen is moved along the active area of the data table surface, the appropriate cursor symbol (1# or 2#) moves at a rate proportional to the movement of the pen. The 1# symbol is associated with the data tablet connected to port 4 and the 2# symbol is associated with the data tablet connected to port 8.

NOTE

Successively pressing function key F1 causes the terminal verification pattern to switch from processing data tablet messages to trackball/forcestick messages and vice versa.

2.2.2 LOCAL MODE COMMANDS

After the GRAPHIC 7 has been initialized in the local mode and the verification test pattern is no longer required, display of the pattern may be terminated by pressing the RETURN key on the keyboard. The pattern then disappears and the letters "BØ M" are displayed in the center of the CRT screen as an indication that the system is in the local monitor mode. At this point, the operator can perform any of several operations that permit him to monitor or debug a program, transfer control, or communicate with the host computer.

NOTE

Commands are executed when the RETURN key on the keyboard is pressed.

The following paragraphs discuss commands that can be executed when the system is in the local monitor mode. A summary of the commands is given in table 2-2.

TABLE 2-2

LOCAL MODE COMMAND SUMMARY

Keyboard Entry	Operation
RETURN	Executes local mode command or returns system to local monitor level.
nnnnnn/	Displays contents of memory address nnnnnn (octal).
/	Increments memory address counter by two and displays address contents.
∧ or ↑	Decrements memory address counter by two and displays address contents.
Bn	Select different memory bank. (BØ 0-32K; B1 32-64K; B2 64-96K; B3 96-128K; and B4 16-32K RAM).
S	Transfers GRAPHIC 7 to system mode operation.
T RETURN	Transfers to the verification test pattern.
L RETURN	Loads memory from paper tape reader.
nnnnL RETURN	Loads selected option from expansion module.
U RETURN	Unload all options.
O RETURN	Display status of all options loaded.
Q	Decrements contents of display processor Q register by two and displays result. Used with diagnostics to indicate address at which display processor halted.
nnnnnnD RETURN	Directs graphic controller to display refresh file beginning at address nnnnnn (octal).
nnnnnnG RETURN	Transfers control of display processor to program beginning at memory address nnnnnn (octal).
Y RETURN P RETURN*	Calls teletypewriter emulation program. After entering emulation program, function key FØ clears CRT screen. Function key F1 selects full or half duplex operation; receipt of octal code 035 from the host computer or pressing function key F13 transfers GRAPHIC 7 to system operating mode. (Y - Serial Entry; P - Parallel Entry)
RUB OUT	Deletes last octal entry from keyboard.

*Only applicable to release 3 and greater.

2.2.2.1 Memory Commands

The content of a memory location is displayed by typing the octal address (typing of leading zeros is not required) followed by a slash (/). As soon as the slash is typed, the content of the memory location is displayed immediately to the right of the address. Successive memory locations can then be examined simply by pressing the slash key. Each time the slash key is pressed, the memory address is incremented by two and its content displayed immediately to the right of the slash.

After the slash key has been used to examine the content of a memory location, the up arror (↑ or ∧) key may be used in a similar manner to examine preceding memory locations. Each time the up arrow key is pressed, the memory address is decremented by two and its content displayed immediately to the right of the slash.

The content of a memory location may be changed after it has been examined by typing the new data (typing of leading zeros is not required) before pressing the slash or up arrow key. The new data is displayed to the right of the old data and is automatically substituted when the slash or up arrow key is pressed.

Memory locations in other banks can be examined or changed via the bank (B) select command. Typing BØ, B1, B2, B3, or B4 causes the memory bank selection to be changed to bank Ø, bank 1, bank 2, bank 3, or bank 4 respectively. Below is a table representing the associated virtual and physical addresses for each bank.

Bank Number	Virtual Address	Physical Address	Pages
0*	000000-177777	000000-177777	00-07
1	000000-177777	200000-377777	10-17
2	000000-177777	400000-577777	20-27
3	000000-177777	600000-777777	30-37
4*	100000-177777	100000-177777	04-07

NOTE

*Addresses in the range of 100000-177777 (pages 4, 5, 6, and 7) for bank Ø correspond to ROM and I/O device registers. Addresses in the range of 100000-177777 for bank 4 correspond to RAM.

Return to the monitor level is accomplished by pressing the RETURN key. When this key is pressed, any specified memory content change is completed and the system returns to monitor level as indicated by letters "Bn M" displayed at the center of the CRT screen.

2.2.2.2 Displaying a Refresh File

When the system is in the local monitor mode, the contents of a refresh file may be displayed by typing the starting address of the file (in octal notation) followed by a "D" and then pressing the RETURN key. This command instructs the graphic controller to display the entire refresh file that begins at the specified address. Display of the refresh file continues until the RETURN key is pressed again, at which time the system returns to the local monitor level. This command is subject to the bank argument presently displayed.

2.2.2.3 Transfer of Program Control

Program control may be transferred from local monitor level to any desired address location in bank Ø by typing the address location in octal notation followed by a "G" and then pressing the RETURN key. The display processor then executes instructions beginning with the instruction at the specified address. Any further operations depend on the program to which control is transferred.

2.2.2.4 Transfer to System Mode

To transfer to the system mode of operation from monitor level, type "S". This command has the same effect as pressing the SYSTEM switch on the terminal controller (paragraph 2.3). After transferring to the system mode, operation in the local mode can be reestablished only by a message from the host computer or by pressing the LOCAL switch on the terminal controller.

2.2.2.5 Teletypewriter Emulation

For purposes of communicating with a host computer, the GRAPHIC 7 can be made to emulate the functions of a teletypewriter. In this mode, the keyboard operates like the keyboard of a teletypewriter and the display indicator serves as the printout device. Scrolling of data on the display indicator is handled on a half-page basis. That is, when the CRT screen is full, the top half of the data is deleted from the display and the bottom half of the data moves up to take its place.

The emulation program is entered from the monitor level by typing the letter Y or P, followed by RETURN. Y selects serial interface port 1 as the host interface; P selects the parallel interface as the host interface.

Full-duplex or half-duplex emulation may then be selected by pressing function key F1 which changes the selection each time it is pressed. The type of emulation selected is indicated by the "TTY F" (full-duplex) or "TTY H" (half-duplex) that is displayed at the top of the CRT screen at all times during emulation. Switching between full and half duplex operation may be accomplished at any time during emulation by pressing function key F1. Pressing function key FØ during teletypewriter emulation causes the CRT screen to be cleared.

Exit from the teletypewriter emulation program occurs when octal code 035 (ASCII control character GS Group Separator) is received from the host computer. This code, which can also be generated by pressing function key F13, immediately causes the GRAPHIC 7 to transfer to the system mode of operation. Return to the local monitor level can be achieved only by a command from the host computer or by pressing the LOCAL pushbutton switch on the terminal controller.

NOTE

For releases 1 and 2 of GCP+ (see Appendix D), the teletype emulator is entered by typing Y, followed by RETURN. Then, if a parallel interface card is installed in the terminal controller, the graphic control program assumes that communications with the host computer are to be handled over the parallel interface. In this case, teletypewriter emulation signals are transmitted in parallel using only the low order byte (bits 0-7) of the 16-bit interface. If a parallel interface card is not installed, a standard 8-bit serial interface via serial interface port 1 is assumed. In either case, bit 7 is always equal to zero.

2.2.2.6 Additional Local Mode Commands

Additional commands that can be used when the GRAPHIC 7 is in the local mode at the monitor level are the L, U, O, T, Q, and RUB OUT commands, The L command enables the memory to be loaded from a paper tape reader connected to the terminal controller. After the tape has been placed in the reader, loading is initiated by typing the letter "L" followed by RETURN.

NOTE

A paper tape reader may be connected to multiport serial interface card ports 1, 2, or 3 or to the serial interface port on the ROM and status logic card.

The L command can also be used to load in options from the expansion module. The option command format is as follows:

nnnnL RETURN

where nnnn is the option number. Valid option numbers are in the ranges of 1 to 3777 and 4001 to 7776.

NOTE

The optional expansion module can store a variety of option types.

The U command is used to unload all options. Typing "U" followed by RETURN will cause all options to be unloaded.

The O command is used to detect the presence and status of all loaded options. Typing O followed by RETURN causes the display of the first option loaded. Successively pressing the RETURN key causes the display of a-1 other options loaded. The option status is displayed in the following format.

nnnn ss Where nnnn is the option number and ss is the option status

The option status code is as follows:

00	Detected but unloaded
01	Unloaded, checksum error (local)
11	Unloaded, checksum error (system)
02	Unloaded, checksum OK, hardware not present (local)
12	Unloaded, checksum OK, hardware not present (system)
03	Unloaded, checksum OK, self test = no go (local)
13	Unloaded, checksum OK, self test = no go (system)
04	Loaded, checksum OK, self test = go (local)
14	Loaded, checksum OK, self test = go (system)

The T command is used to recall the verification test pattern when the system is at the local monitor level. This command is executed by typing the letter "T"

followed by RETURN. The effect is the same as pressing the LOCAL switch on the terminal controller. Pressing RETURN a second time causes the system to return to the monitor level.

The Q command is a special command used for diagnostic and debugging purposes. Whenever a HALT instruction is executed by the display processor, the content of the program counter is stored in the Q register of the display processor. After the system has been reinitialized by pressing the LOCAL switch on the terminal controller, the Q command may be used to display the address at which the display processor halted. The Q command is executed by typing the letter "Q". This causes the content of the Q register to be decremented by two and the result displayed to indicate the address of the HALT instruction. Note that the Q command always decrements the content of the Q register by two and displays the result. The result, however, is only meaningful immediately following initialization in the local mode after a HALT instruction has been executed. After using a Q command, pressing RETURN returns the system to the local monitor level.

The RUB OUT command provides a means of correcting erroneous entries from the keyboard. At any time before a command is executed, pressing RUB OUT causes the last keystroke entry to be deleted. An additional entry is deleted each time the RUB OUT key is pressed.

2.2.3 STANDARD TRANSFER TABLE

ROM addresses 157700 through 157770 (octal) constitute a standard transfer table for certain routines of the graphic control program (GCP+). Information identifying the version of GCP+ installed in the ROM is also contained in these addresses. When the GRAPHIC 7 is operating in the local mode, the contents of locations containing information can be examined by typing the address of the location followed by a slash (paragraph 2.2.2.1). To transfer to one of the GCP+ routines, the G command should be used as described in paragraph 2.2.2.3. Table 2-3 lists the addresses in the standard transfer table and identifies the information or routine associated with each.

NOTE

When the GRAPHIC 7 is operating in the system mode, transfer to one of the GCP+ routines can be accomplished by using a host-to-GRAPHIC 7 TK message (para. 5.3).

2.3 SYSTEM MODE

The system mode of operation is the normal operating mode of the GRAPHIC 7. Initialization in the system mode occurs under any of the following conditions:

a. When primary ac power is applied to the terminal controller.

b. When the SYSTEM switch on the terminal controller is pressed.

c. When the GRAPHIC 7 is in the local operating mode and S is typed on the keyboard.

d. When the GRAPHIC 7 is in the local operating mode and 157760G RETURN is typed on the keyboard.

e. When the GRAPHIC 7 is in the system operating mode and an IZ message is sent from the host computer to the GRAPHIC 7.

f. When an initialize signal is sent from the host computer to the GRAPHIC 7 via the parallel interface or the multiport serial interface.

g. When the GRAPHIC 7 is in the teletypewriter emulation mode (paragraph 2.2.2.5) and octal code 035 (ASCII control character GS Group Separator) is generated by the host computer or by pressing function key F13 on the keyboard.

Initialization in the system mode automatically causes the built-in diagnostic routines to be performed and the results sent in an error status message (paragraph 5.4) to the host computer. The diagnostic routines are the same as those run during local mode initialization (paragraph 2.2.1) except that, of the two interface diagnostics, only the one associated with communications to the host computer is performed. A checksum of GCP+ stored in read only memory is also calculated and the results included in the error status message to the host computer.

In the system mode, responses to all operator actions are determined by the application program in the host computer. Control is exercised and data is transferred by means of messages sent between the host computer and the terminal controller. Refer to Section 5 for a complete description of the form and content of these messages.

NOTE

Transfer to routines listed in table 2-3 can be accomplished in the system operating mode by using host-to-GRAPHIC 7 TK messages (paragraph 5.3).

TABLE 2-3

STANDARD TRANSFER TABLE

Address (octal)	Information or Routine	Remarks
157700	GCP+ date (year and month)	High order byte indicates month in octal form. Low order byte indicates last two digits of year in octal form (e.g., 003115 indicates June 1977)
157702	GCP+ date (day)	Day of month is indicated in octal form.
157704	GCP+ release number	Release number is indicated in octal form.
157706	Number of GCP+ field changes	Number of field changes is indicated by the number of bits set to 1 (e.g., 000007 indicates three field changes).
157710	ZERO	Maintenance routine. Graphic controller sets X and Y positions at zero (center of CRT screen) and then halts. Used for voltmeter adjustment of vector/position generator output voltages.
157720	PLUS	Maintenance routine. Graphic controller sets X and Y positions at maximum on-screen positions (upper and right corner) and then halts. Used for voltmeter adjustment of vector/position generator output voltages.
157730	MINUS	Maintenance routine. Graphic controller sets X and Y positions at minimum on-screen positions (lower left corner) and then halts. Used for voltmeter adjustment of vector/position generator output voltages.
157740	LOADER	Enables a file to be loaded into read/write memory from various input devices.

TABLE 2-3

STANDARD TRANSFER TABLE (Cont)

Address (octal)	Information or Routine	Remarks
157750	DEBUG MODE	Enables local mode commands to be executed.
157760	SYSTEM MODE	Enables system mode.
157770	TEST PATTERN	Transfers to verification test pattern.

SECTION 3
GRAPHIC 7 INSTRUCTIONS

3.1 GENERAL

Instructions used by the GRAPHIC 7 can be divided into two categories: those executed by the display processor and those executed by the graphic controller. The two microcontrollers operate independently of one another and share common memories via the controller bus (see figure 1-3). Running of GCP+ is controlled by the display processor while generation of the display image is controlled by the graphic controller. The following paragraphs provide details concerning the instructions executed by each microprocessor.

NOTE

In the octal codes shown for each instruction, unused bits are indicated by X. Bits representing variable data are indicated by d.

3.2 DISPLAY PROCESSOR INSTRUCTIONS

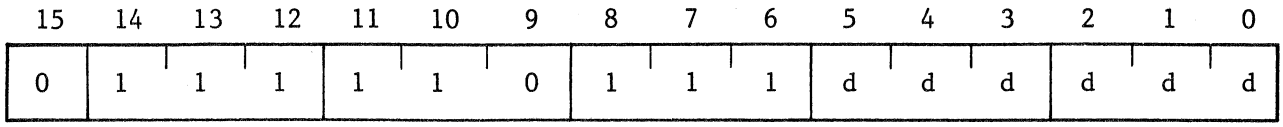
The display processor emulates a minicomputer of the PDP-11 type manufactured by Digital Equipment Corporation (DEC). As such, the display processor is capable of executing the standard set of instructions used for the PDP-11/34 minicomputer and user software may be prepared using standard DEC mnemonics. Other PDP-11 instructions that can be executed are the MUL, DIV, ASH, ASHC, and SPL instructions. Details concerning PDP-11 instructions are contained in the DEC PDP-11/04/34/45/55/60 Processor Handbook which should be used as a supplement to this manual.

An additional instruction that can be executed by the display processor is the EXCQ (exchange register Q) instruction. The format and operation of this instruction are as follows:

EXCQ

EXCHANGE REGISTER Q

Octal code: 0767dd



The EXCQ instruction causes the contents of register Q to be exchanged with the contents of the general register specified by bits 0-5. Its primary purpose is to provide a means of retrieving the contents of the program counter following the execution of a HALT instruction. When a HALT instruction is executed, the contents of the program counter (HALT instruction address + 2) is stored in register Q. The EXCQ instruction can then be used to move the value to a general register so that it can be processed as required. The EXCQ instruction is used in conjunction with the Q command of the local operating mode to enable the operator to determine the address at which the display processor halted (refer to paragraph 2.2.2.6).

Operation:

(REG Q) ← (REG DD)

(REG DD) ← (REG Q)

Cordition Codes:

N: Set if value in Q reg < Ø; cleared otherwise

Z: Set if value in Q reg = Ø; cleared otherwise

V: Cleared

C: Not affected

3.3 GRAPHIC CONTROLLER INSTRUCTIONS

The graphic controller instruction set comprises 40 instructions that are used to control the generation of the image to be displayed. These instructions can be broken down into four basic categories: beam control instructions, sequence control instructions, register instructions, and display control instructions. Beam control instructions determine basic positioning and unblanking of the CRT beam for the purpose of drawing vectors, conics, and characters. Sequence control instructions direct the graphic controller to jump, branch, halt, or wait as required for proper program execution. Register instructions permit data to be manipulated using the four general purpose registers and the stack pointer of the graphic controller. Display instructions enable various parameters to be established or modified as required to achieve the desired display image characteristics.

NOTE

Macros written in MACRO-11 assembly language are available for GRAPHIC 7 users. These macros can be used to assemble all the graphic controller instructions plus some useful instruction sequences. Refer to Appendix B for descriptions of the macros.

The complete file of instructions and data that defines a particular display image is referred to as a refresh file. This file, which may be located in read only and/or read/write memory, is accessed by the graphic controller and executed in its entirety at a rate of 60, 40, or 30 Hz (programmable) to create a visible image on the CRT of the display indicator.

Instructions are fetched by the graphic controller via the processor bus, acted upon, and the resulting data placed on the graphic bus for application to appropriate circuits (refer to figure 1-3). The following paragraphs describe the format and function of each instruction in each category of graphic controller instructions. A summary of the instructions is contained in Appendix A.

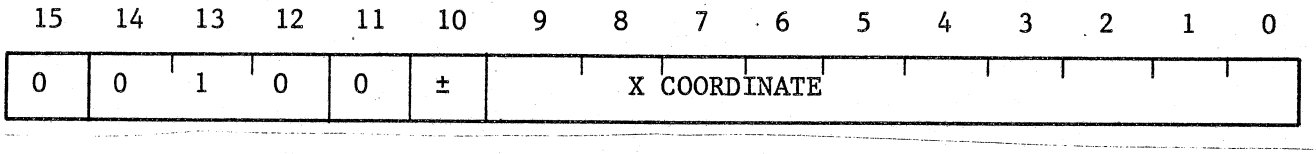
3.3.1 BEAM CONTROL INSTRUCTIONS

Seventeen instructions are used to control the motion of the CRT beam in the display indicator. These instructions include load, move, draw, text, and conic instructions.

3.3.1.1 Load Instructions

Load instructions specify X-axis positions on the CRT screen in terms of absolute data (specific coordinates) or relative data (lengths along X axis). These instructions do not initiate beam motion. Beam motion is initiated by move or draw instructions (paragraphs 3.3.1.2 and 3.3.1.3) which contain corresponding data for Y-axis positions. Therefore, except for short relative moves or draws, if both X- and Y-axis data are to be changed, a load instruction must precede a move or draw instruction.

LDXA — LOAD X ABSOLUTE — Octal code: 02dddd

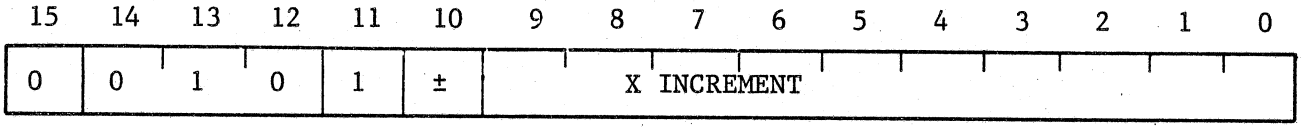


Bits 0-10 define absolute X-axis coordinate in two's complement form. These bits (sign extended) replace contents of X position register. Beam position does not change.

Operation: (DXR) ← X COORDINATE

Beam motion: None

LDXR — LOAD X RELATIVE — Octal code: 024ddd



Bits 0-10 define increment in X-axis coordinate in two's complement form. These bits (sign extended) are added to contents of X position register. Beam position does not change.

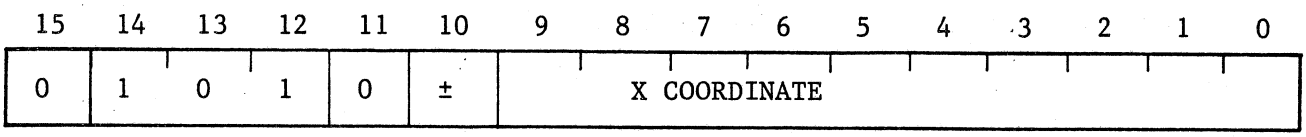
Operation: (DXR) ← (DXR) + X INCREMENT

Beam motion: None

3.3.1.2 Move Instructions

Move instructions specify X- and/or Y-axis positions on the CRT screen in terms of absolute data (specific coordinates) or relative data (lengths along X and/or Y axis) and initiate blanked beam motion to new position. Except for a move short relative instruction, a load instruction (paragraph 3.3.1.1) must precede a move instruction when both X- and Y-axis data are to be changed.

MVXA MOVE X ABSOLUTE Octal code: 05dddd

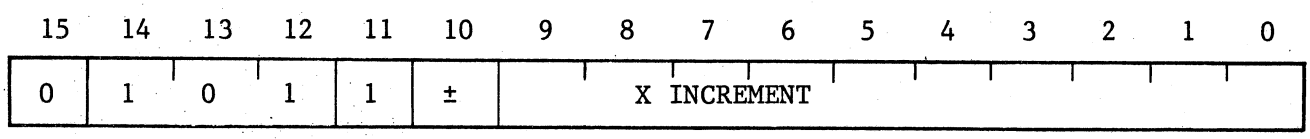


Bits 0-10 define absolute X-axis coordinate in two's complement form. These bits (sign extended) replace contents of X position register. Beam then moves blanked to coordinates specified by X and Y position registers.

Operation: (DXR) ← X COORDINATE

Beam motion: Blanked to (DXR),(DYR)

MVXR MOVE X RELATIVE Octal code: 054ddd

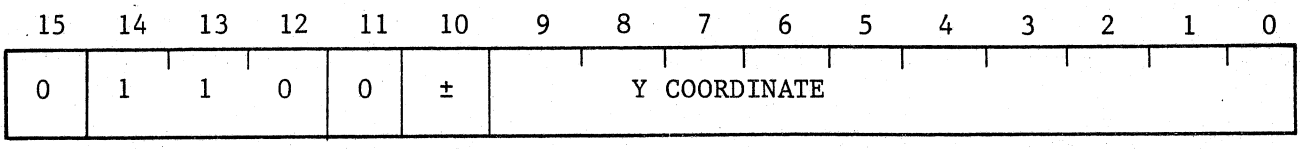


Bits 0-10 define increment in X-axis coordinate in two's complement form. These bits (sign extended) are added to contents of X position register. Beam then moves blanked to coordinates specified by X and Y position registers.

Operation: (DXR) ← (DXR) + X INCREMENT

Beam motion: Blanked to (DXR),(DYR)

MVYA — MOVE Y ABSOLUTE — Octal code: 06dddd

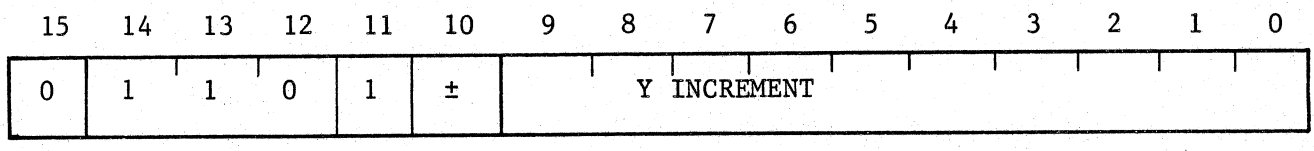


Bits 0-10 define absolute Y-axis coordinates in two's complement form. These bits (sign extended) replace contents of Y position register. Beam then moves blanked to coordinates specified by X and Y position registers.

Operation: (DYR) ← Y COORDINATE

Beam motion: Blanked to (DXR),(DYR)

MVYR — MOVE Y RELATIVE — Octal code: 064ddd

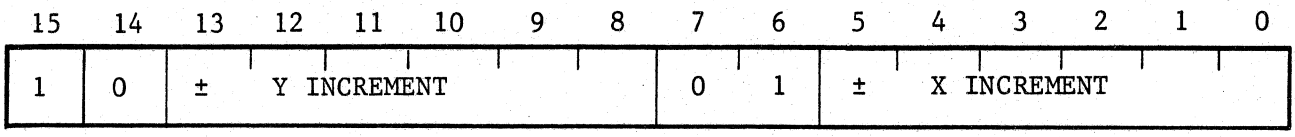


Bits 0-10 define increment in Y-axis coordinate in two's complement form. These bits (sign extended) are added to contents of Y position register. Beam then moves blanked to coordinates specified by X and Y position registers.

Operation: (DYR) ← Y INCREMENT

Beam motion: Blanked to (DXR),(DYR)

MVSR — MOVE SHORT RELATIVE — Octal code: 1dd1dd



Bits 0-5 define increment in X-axis coordinates; bits 8-13 define increment in Y-axis coordinate. Both sets of bits are in two's complement form. These bits are added (sign extended) to contents of X and Y position registers, respectively. Beam then moves blanked to coordinates specified by X and Y position registers.

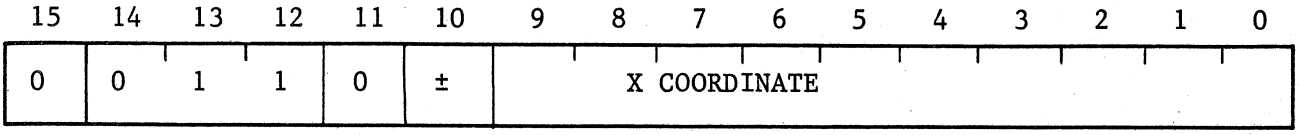
Operation: (DXR) ← (DXR) + X INCREMENT
(DYR) ← (DYR) + Y INCREMENT

Beam notion: Blanked to (DXR),(DYR)

3.3.1.3 Draw Instructions

Draw instructions are similar to move instructions (paragraph 3.3.1.2) except that they cause unblanked beam motion to enable vectors to be drawn. A point plot relative instruction is also included in the draw instructions. This instruction causes the beam to move blanked to a new position and then unblank momentarily to display a point.

DRXA DRAW X ABSOLUTE Octal code: 03dddd

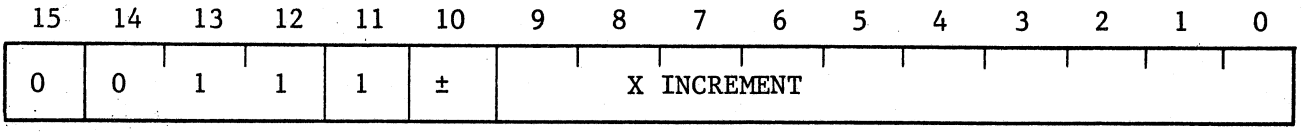


Bits 0-10 define absolute X-axis coordinate in two's complement form. These bits (sign extended) replace contents of X position register. Beam then moves unblanked to coordinates specified by X and Y position registers.

Operation: (DXR) ← X COORDINATE

Beam motion: Unblanked to (DXR),(DYR)

DRXR DRAW X RELATIVE Octal code: 034ddd



Bits 0-10 define increment in X-axis coordinate in two's complement form. These bits (sign extended) are added to contents of X position register. Beam then moves unblanked to coordinates specified by X and Y position registers.

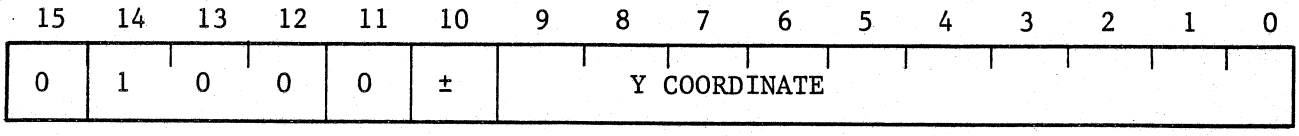
Operation: (DXR) ← (DXR) + X INCREMENT

Beam motion: Unblanked to (DXR),(DYR)

DRYA

DRAW Y ABSOLUTE

Octal code: 04dddd



Bits 0-10 define absolute Y-axis coordinate in two's complement form. These bits (sign extended) replace contents of Y position register. Beam then moves unblanked to coordinates specified by X and Y position registers.

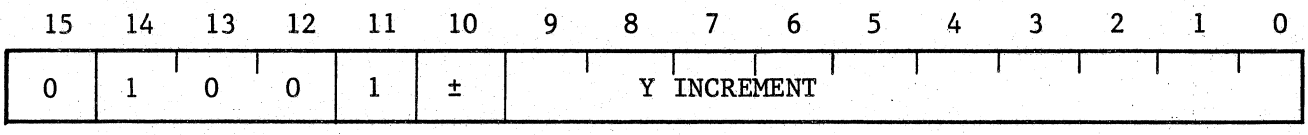
Operation: (DYR) ← Y COORDINATE

Beam motion: Unblanked to (DXR),(DYR)

DRYR

DRAW Y RELATIVE

Octal code: 044ddd

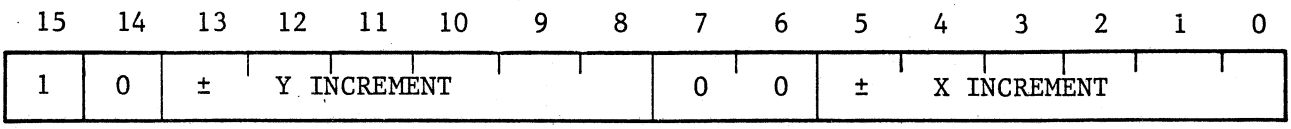


Bits 0-10 define increment in Y-axis coordinate in two's complement form. These bits (sign extended) are added to contents of Y position register. Beam then moves unblanked to coordinates specified by X and Y position registers.

Operation: (DYR) ← (DYR) + Y INCREMENT

Beam motion: Unblanked to (DXR),(DYR)

DRSR DRAW SHORT RELATIVE Octal code: 1dd0dd

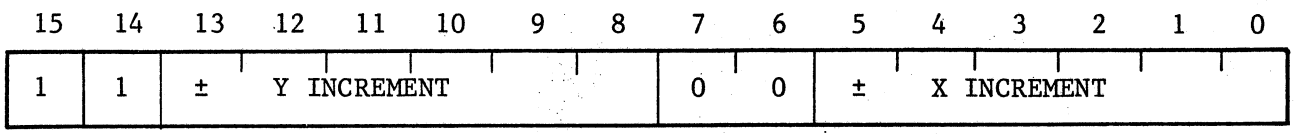


Bits 0-5 define increment in X-axis coordinate; bits 8-13 define increment in Y-axis coordinate. Both sets of bits are in two's complement form. These bits are added (sign extended) to contents of X and Y position registers, respectively. Beam then moves unblanked to coordinates specified by X and Y position registers.

Operation: (DXR) ← (DXR) + X INCREMENT
(DYR) ← (DYR) + Y INCREMENT

Beam motion: Unblanked to (DXR),(DYR)

PPLR POINT PLOT RELATIVE Octal code: 1dd0dd



Bits 0-5 define increment in X-axis coordinate; bits 8-13 define increment in Y-axis coordinate. Both sets of bits are in two's complement form. These bits are added (sign extended) to contents of X and Y position registers, respectively. Beam then moves blanked to coordinates specified by X and Y position registers at which location beam stops and unblanks to display a point.

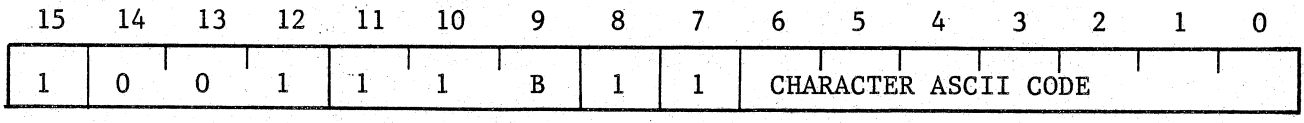
Operation: (DXR) ← (DXR) + X INCREMENT
(DYR) ← (DYR) + Y INCREMENT

Beam motion: Blanked to (DXR),(DYR); then unblanked

3.3.1.4 Text Instructions

Two instructions are used to draw characters. One instruction causes a single steady or blinking character to be drawn at the current beam position. The second instruction enables two characters to be drawn and the beam position incremented automatically when each is complete. Character size and orientation data are not included in the text instructions. These parameters as well as the values used to increment the beam position are determined by display control instructions (paragraph 3.2.4).

CHAR DRAW SINGLE CHARACTER Octal code: 116ddd (blink)
117ddd (steady)



Bits 0-6 specify ASCII code of character to be drawn at location defined by contents of X and Y position registers. Bit 9 specifies whether the character is to blink or be steady (1 indicates steady). Bits 0-6 replace contents of character register while bit 9 replaces blink bit in display Z register. After character is drawn, beam returns to location defined by X and Y position registers and previous state of blink bit is restored in display Z register.

Operation: (DCR) ← CHARACTER ASCII CODE
DZR (blink bit) ← Bit 9
DZR blink bit is restored following instruction execution

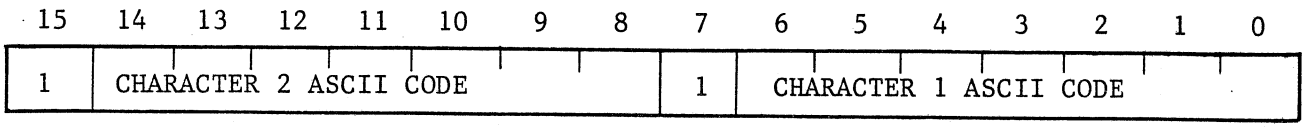
Beam motion: Draw character and return to starting location

NOTE

On customer systems containing special characters or special symbols, the shift out code (character code octal 16) must be used to display these symbols. This is done by executing a CHAR instruction containing the shift out code. Following the shift out code is a group of CHAR/TEXT instructions to display the selected special symbols. After displaying special symbols, the shift in code (character code octal 17) must be used to permit display of the standard characters again.

TXT

DRAW TWO TABULAR CHARACTERS Octal code: 1dd2dd



Bits 0-6 specify ASCII code of first character to be drawn; bits 8-14 specify ASCII code of second character to be drawn. Beginning location for each character is defined by X and Y position registers. Bits 0-6 replace contents of character register, character is drawn, then bits 8-14 replace contents of character register. When a character is completed, the X position register (Y position register for rotated characters) is automatically incremented by contents of text increment register. If ASCII code for a NULL character is specified, no character is drawn and beam position is not incremented.

Operation: Draw character 1 at position DXR,DYR
(DXR) ← (DXR) + (DTI) [(DYR) ← (DYR) + (DTI) for rotated text]
Repeat above for character 2

Beam motion: Draw character 1
Increment per DTI
Repeat above for character 2

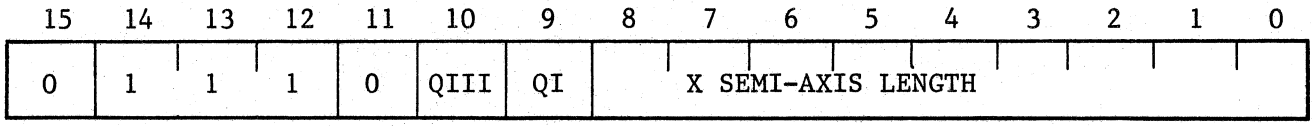
3.3.1.5 Conic Instructions

Two conic instructions are used with the optional ramp/conic generator card to specify 90-degree segments and axis lengths of ellipses to be displayed. One is a load instruction that specifies X-axis data without initiating beam motion. The second is a draw instruction that specifies Y-axis data and initiates beam motion in both axes. Bits in both instructions specify what combination of 90-degree segments will be unblanked when an ellipse is drawn. Axes of all ellipses drawn using these instructions lie parallel to the X and Y axes of the display indicator.

NOTE

Both conic instructions are required to define each ellipse even when parameters specified by one instruction do not change. The load instruction must precede the draw instruction. If the draw instruction is used alone, a circle will be displayed with a radius equal to the length specified for the semi-Y axis.

LDKX LOAD CONIC X REGISTER Octal code: 07dddd



Bits 0-8 define semi-axis length of ellipse (distance from ellipse center to its perimeter) along X axis of CRT. Bits 9 and 10 specify beam unblanking (1 indicates unblank) in quadrants QI (upper right) and QIII (lower left), respectively. All of bits 0-10 replace contents of X conic register. Beam position does not change.

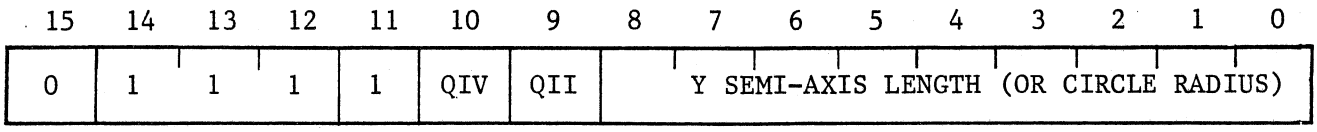
Operation: (KXR) ← X SEMI-AXIS LENGTH plus QI and QIII bits

Beam motion: None

DRKY

DRAW CONIC Y

Octal code: 07dddd



Bits 0-8 define semi-axis length of ellipse (distance from ellipse center to its perimeter) along Y axis of CRT. Bits 9 and 10 specify beam unblanking (1 indicates unblanking) in quadrants QII (upper left) and QIV (lower right), respectively. All of bits 0-10 replace contents of conic Y register. Beam then moves to draw ellipse defined in conic X and Y registers with ellipse center at location defined by X and Y position registers. Beam is unblanked in quadrants specified by bits 9 and 10 in the conic X and conic Y registers. If a DRKY instruction is not preceded by an LDKX instruction, bits 0-10 of the DRKY instruction replace the contents of the conic X as well as the conic Y register. The result is that a circle with a radius equal to the length specified by bits 0-8 is drawn. Bits 9 and 10 then specify beam unblanking for the upper and lower semicircles, respectively.

Operation: Preceded by LDKX instruction:

(KYR) ← Y SEMI-AXIS LENGTH plus QII and QIV bits

Not preceded by LDKX instruction:

(KYR) ← Y SEMI-AXIS LENGTH plus QII and QIV bits

(KXR) ← Y SEMI-AXIS LENGTH plus QII and QIV bits

Beam motion: Elliptical as defined by conic X and conic Y registers centered at point defined by X and Y position registers. Unblanked as determined by bits QI thru QIV.

3.3.2 SEQUENCE CONTROL INSTRUCTIONS

Twelve instructions are used to control the sequence of program execution and timing by the graphic controller. These instructions include unconditional jump, conditional jump, subroutine, linkage, halt, and wait instructions.

3.3.2.1 Unconditional Jump Instructions

Unconditional jump instructions permit program control of the graphic controller to be transferred directly or indirectly to a specific address in memory (absolute jump) or to an address removed from the current location by a specified increment (relative jump). The jump short relative instruction (JMPR) can also be used as a no-operation instruction (NOOP) by specifying a jump increment of zero bytes.

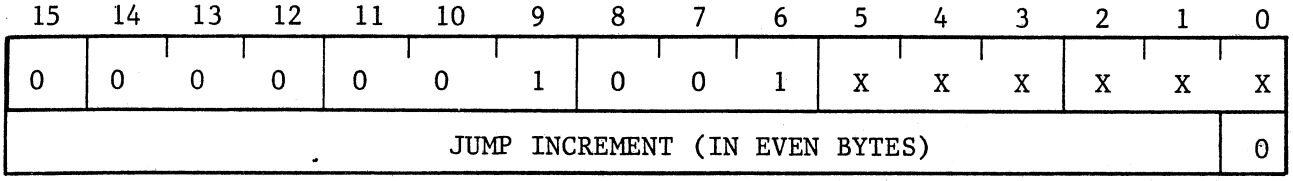
JUMP JUMP Octal code: 0010XX dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	0	X	X	X	X	X	X
I	JUMP ADDRESS														0

JUMP is a two-word instruction used for unconditional transfer of program control (direct or indirect) to a specific location in memory. The first word identifies the instruction; the second word specifies a direct or indirect address to which program control is to be transferred. The specified address may be the address of any even-numbered byte from 00000 to 77776 (octal). If bit 15 of the second word is set to 0 (direct addressing), control is transferred to the address specified by bits 0-14. If bit 15 is a 1 (indirect addressing), bits 0-14 specify the memory address that contains the required data. In this case, the contents of the specified address are used as the location to which program control is transferred. Note that direct addressing cannot be used for addresses greater than 77776 (octal). Multilevel indirect addressing may be used.

Operation: Direct: (DPC) ← JUMP ADDRESS
Indirect: (DPC) ← (JUMP ADDRESS)

JRMP JUMP RELATIVE Octal code: 0011XX dddddd



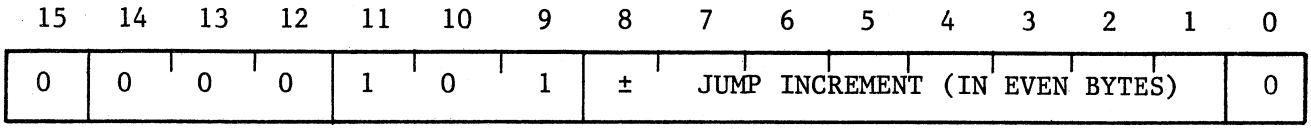
JRMP is a two-word instruction that causes unconditional transfer of program control to a relative location in memory. The first word identifies the instruction; the second word specifies an even number of bytes by which the program counter is to be incremented. The jump increment is added modulo 2^{16} (any carry is ignored) to the contents of the program counter which is pointing to the address following the location of the jump increment word. The result is used as the address of the next instruction to be executed by the graphic controller. Relative jumps of 0 to 77776 (octal) bytes can be accomplished using this instruction.

Operation: (DPC) ← (DPC) + JUMP INCREMENT

JMPR JUMP SHORT RELATIVE or NO OPERATION Octal code: 005ddd (JMPR)
005000 (NOOP)

or

NOOP



Bits 0-8 specify, in two's complement form, an even number of bytes by which the program counter is to be incremented (or decremented). These bits are added to the contents of the program counter which is pointing to the address following the location of the JMPR or NOOP instruction. The result is used as the address of the next instruction to be executed by the graphic controller. Relative jumps of 0 to 776 (octal) bytes in either direction can be accomplished using this instruction. Specifying a relative jump of 0 bytes results in a no-operation instruction.

Operation: (DPC) ← (DPC) + JUMP INCREMENT

3.3.2.2 Conditional Jump Instructions

Two conditional jump instructions are provided to permit program control to be transferred or to continue in normal sequence as determined by testing the contents of general purpose register 0. Jumps are executed when the contents of this register are not equal to zero. One instruction causes a conditional jump, direct or indirect, to a specific address in memory (absolute jump). The second instruction causes a conditional jump to an address removed from the current location by a specified increment (relative jump).

JMPZ JUMP IF DISPLAY REGISTER 0 CONTENTS ≠ 0 Octal code: 0012XX dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	0	X	X	X	X	X	X
I	JUMP ADDRESS														0

JMPZ is a two-word instruction used for the conditional transfer (direct or indirect) of program control to a specific memory location. The first word identifies the instruction and causes the contents of general register 0 (DR0) to be tested. The second word specifies a direct or indirect address to which program control is conditionally to be transferred. The specified address may be the address of any even-numbered byte from 00000 to 77776 (octal). Program control is transferred only when (DR0) ≠ 0. If bit 15 of the second word is set to 0 (direct addressing), control is transferred to the address specified by bits 0-14. If bit 15 is a 1 (indirect addressing), bits 0-14 specify the memory address that contains the required data. In this case, the contents of the specified address is used as the location to which program control is transferred. If (DR0) = 0, program control is not transferred and the program continues by executing the instruction at the address that immediately follows the second word of the JMPZ instruction (this is the address to which the program counter is pointing). Note that direct addressing cannot be used for addresses greater than 77776 (octal). Multilevel indirect addressing may be used.

Operation: Direct: (DR0) ≠ 0: (DPC) ← JUMP ADDRESS
(DR0) = 0: (DPC) ← (DPC)
Indirect: (DR0) ≠ 0: (DPC) ← (JUMP ADDRESS)
(DR0) = 0: (DPC) ← (DPC)

JPRZ

JUMP RELATIVE IF DISPLAY REGISTER 0 CONTENTS ≠ 0

Octal code: 0013XX dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	1	X	X	X	X	X	X
JUMP INCREMENT (IN EVEN BYTES)															0

JPRZ is a two-word instruction that causes a conditional transfer of program control to a relative location in memory. The first word identifies the instruction and causes the contents of general purpose register 0 (DRO) to be tested. The second word specifies an even number of bytes by which the program counter is conditionally to be incremented. Program control is transferred only when (DRO) ≠ 0. When (DRO) ≠ 0, the jump increment is added modulo 2^{16} (any carry is ignored) to the contents of the program counter which is pointing to the address following the location of the jump increment word. The result is used as the address of the next instruction to be executed by the graphic controller. When (DRO) = 0, program control is not transferred and the program continues by executing the instruction that immediately follows the second word of the JPRZ instruction. Conditional relative jumps of 0 to 77776 (octal) bytes can be accomplished using this instruction.

Operation: (DRO) ≠ 0: (DPC) ← (DPC) + JUMP INCREMENT

(DRO) = 0: (DPC) ← (DPC)

3.3.2.3 Subroutine Instructions

Four subroutine instructions are provided to permit calls to and returns from subroutines as required. Calls may be made to subroutines located at a specific address in memory (absolute calls) or to subroutines at an address removed from the current location by a specified increment (relative calls). A jump-and-mark instruction is also included which permits direct or indirect calls to be made to subroutines at specific memory locations.

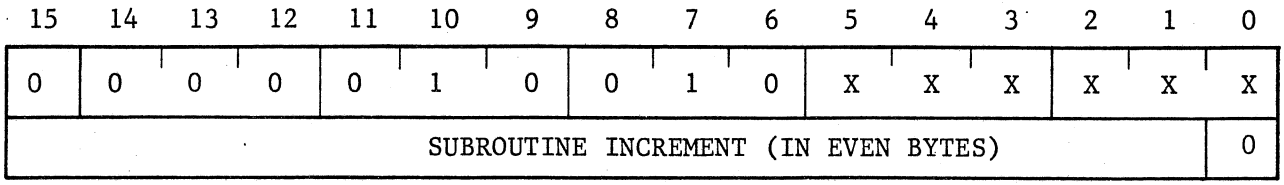
CALL CALL SUBROUTINE Octal code: 0021XX dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	1	X	X	X	X	X	X
SUBROUTINE ADDRESS															0

CALL is a two-word instruction that calls a subroutine from a specific location in memory. The first word identifies the instruction; the second word specifies the address that contains the first instruction of the desired subroutine. The specified address may be the address of any even-numbered byte from 000000 to 177776 (octal). When a CALL instruction is executed, the contents of the program counter (which is pointing to the address following the location of the subroutine address word) is pushed onto the graphic controller stack. This saves the address of the instruction to be executed following completion of the subroutine. The contents of the second word of the CALL instruction is then loaded into the program counter and used as the location of the next instruction to be executed by the graphic controller.

Operation: (DSP) ← (DSP) - 2
(Top stack location) ← (DPC)
(DPC) ← SUBROUTINE ADDRESS

CALR CALL RELATIVE Octal code: 0022XX dddddd



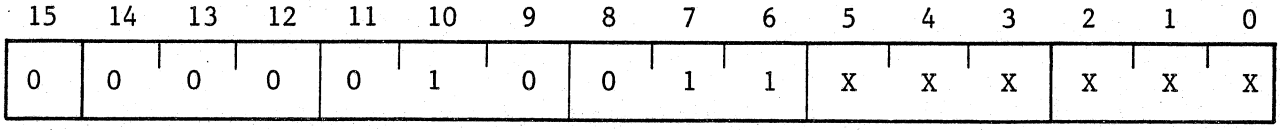
CALR is a two-word instruction that calls a subroutine from a relative location in memory. The first word identifies the instruction; the second word specifies an even number of bytes by which the program counter must be incremented to obtain the address that contains the first instruction of the desired subroutine. The specified increment may be any even number of bytes from 0 to 177776 (octal). When a CALR instruction is executed, the contents of the program counter (which is pointing to the address following the location of the subroutine increment word) is pushed onto the graphic controller stack. This saves the address of the instruction to be executed following completion of the subroutine. The contents of the second word of the CALR instruction is then added modulo 2^{16} (any carry is ignored) to the contents of the program counter and the result used as the location of the next instruction to be executed by the graphic controller.

Operation: (DSP) ← (DSP) - 2
(Top stack location) ← (DPC)
(DPC) ← (DPC) + SUBROUTINE INCREMENT

RTRN

RETURN

Octal code: 0023XX



A RTRN instruction is normally the last instruction of a subroutine called by a CALL or CALR instruction. It causes program control to return from the subroutine to the main program. When a RTRN instruction is executed, the contents of the location indicated by the graphic controller stack pointer is popped from the stack, loaded into the program counter, and used as the address of the next instruction to be executed by the graphic controller.

Operation: (DPC) ← (Top stack location)
(DSP) ← (DSP) + 2

JMPM JUMP AND MARK Octal code: 0020XX dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	X	X	X	X	X	X
I	JUMP ADDRESS (IN EVEN BYTES)														0

JMPM is a two-word instruction used for direct or indirect calls to subroutines. The first word identifies the instruction; the second specifies a direct or indirect address to be used for storage of a return address from the subroutine being called. The specified address may be the address of any even-numbered byte from 00000 to 77776 (octal). If bit 15 of the second word is set to 0 (direct addressing), the return address is stored in the location specified by bits 0-14. If bit 15 is a 1 (indirect addressing), bits 0-14 specify the memory address that contains the required data. In this case, the contents of the specified address are used to designate the location in which the return address will be stored. When a JMPM instruction is executed, the contents of the program counter (which is pointing to the address following the location of the jump address word) is stored in the direct or indirect address specified. This saves the location of the instruction to be executed following completion of the subroutine. Execution of the called subroutine then begins at the address immediately following the location in which the return address is stored. When the subroutine is completed, return to the main program is effected by an indirect JUMP instruction that references the return address storage location. Note that the JMPM instruction cannot be used for direct addressing of addresses greater than 77776 (octal). Multilevel indirect addressing may be used.

Operation: Direct:

(JUMP ADDRESS) ← (DPC)

(DPC) ← JUMP ADDRESS + 2

Indirect:

(Address contained in JUMP ADDRESS) ← (DPC)

(DPC) ← (Address contained in JUMP ADDRESS) + 2

3.3.2.4 Linkage Instruction

A linkage instruction is provided so that synchronized linkage can be effected between a program being executed by the graphic controller and a program being executed by the display processor. This enables the additional power of the display processor to be used to modify or process refresh file data as required. Details concerning applications of the linkage instruction are contained in Section 7.

LINK SYNCHRONIZED LINKAGE Octal code: 0040XX dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	0	X	X	X	X	X	X
I	LINK ADDRESS														0

LINK is a two-word instruction. The first word identifies the instruction; the second word specifies a direct or indirect address to be used for storage of the address that follows the location of the LINK instruction. The specified address may be the address of any even-numbered byte from 00000 to 77776 (octal). If bit 15 of the second word is set to 0 (direct addressing), the storage address is the location specified by bits 0-14. If bit 15 is a 1 (indirect addressing), bits 0-14 specify the memory address that contains the required data. In this case, the contents of the specified address designate the location to be used for storage. When a LINK instruction is executed, the contents of the program counter (which is pointing to the address following the location of the link address word) is stored in the direct or indirect address specified. This saves the address of the instruction that immediately follows the LINK instruction. The graphic controller then halts and interrupts the display processor. Restarting of the graphic controller is controlled by a command from the display processor as described in Section 7. Note that the LINK instruction cannot be used for direct addressing of addresses greater than 77776 (octal). Multilevel indirect addressing may be used.

All indirect addresses are accessed in the page defined by the page register. (See PGR in Section 4.) The direct link address is accessed in page 0.

Operation: Direct: (LINK ADDRESS) ← (DPC)
Indirect: (Address contained in LINK ADDRESS) ← (DPC)

3.3.2.5 Halt and Wait Instructions

One halt and one wait instruction are included in the graphic controller sequence control instructions. The halt instruction is used for debugging while the wait instruction ensures that the displayed image is synchronized with the programmed refresh rate.

HREF — HALT REFRESH — Octal code: 0000XX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X

The HREF instruction causes the graphic controller to halt and to send an interrupt to the display processor. This instruction is normally used for debugging purposes. Whether the interrupt is enabled and the manner in which it is processed are determined by the software being executed by the display processor. Restarting of the graphic controller is controlled by a command from the display processor as described in Section 7.

Operation: Graphic controller:

Halt

Send interrupt to display processor

Display processor:

Process interrupt (if enabled)

WATE

WAIT

Octal code: 0070XX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	0	X	X	X	X	X	X

The WATE instruction is used to synchronize processing of a refresh file to correspond with the selected refresh rate of 60, 40, or 30 Hz. Bits in the display parameter register determine the refresh rate and cause frame sync pulses to be generated at the selected interval. Each time a frame sync pulse is generated, a flip-flop is set. When a WATE instruction is executed, the state of the flip-flop is sensed. If no frame sync pulse has occurred since the previous WATE instruction was executed, a cleared state will be sensed. The graphic controller will then wait until a frame sync pulse sets the flip-flop before executing the next instruction in the refresh file. If a frame sync pulse has been generated since the previous WATE instruction was executed, a set state will be sensed and the graphic controller will proceed immediately to the next instruction in the refresh file. In either case, the flip-flop is reset following the execution of a WATE instruction. In this way, files requiring processing times shorter than the selected interval are processed at the selected rate while files requiring processing times longer than the selected interval are processed at the fastest rate possible. Normally, one WATE instruction should be included in each refresh file.

NOTE

It is strongly recommended that the CRT beam be centered immediately before the WATE instruction is executed. This will improve the reliability of the display indicator by reducing power consumption during the wait period. The following instructions will center the CRT beam:

```
LDXA Ø
MVYA Ø
```

Operation: Frame sync FF cleared:

Wait for frame sync pulse before processing next instruction

Clear frame sync FF

Frame sync FF set:

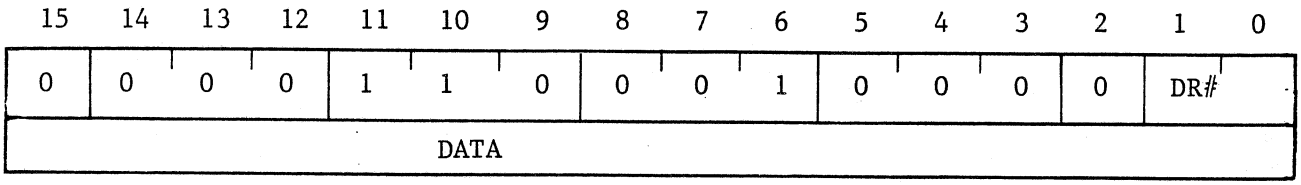
Process next instruction immediately

Clear frame sync FF

3.3.3 REGISTER INSTRUCTIONS

Register instructions are used to load data into general purpose registers of the graphic controller (display registers), to modify register contents, and to control graphic controller stack operations. A register instruction is also provided that enables data to be loaded into registers of optional devices that may be connected to the graphic bus.

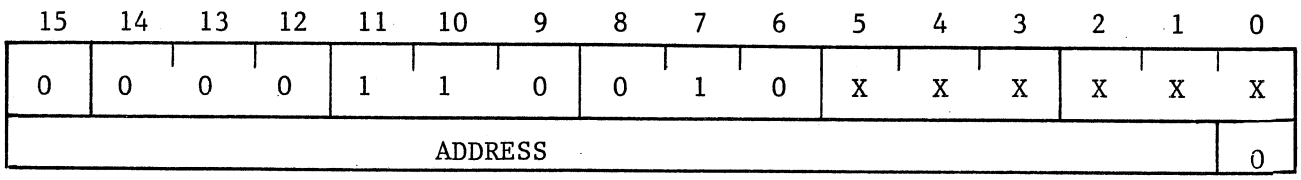
LDDI LOAD DISPLAY REGISTER IMMEDIATE Octal code: 00610d dddddd



LDDI is a two-word instruction used to load data into one of the general purpose registers of the graphic controller. The first word identifies the instruction and the display register into which data is to be loaded. The second word contains the data to be loaded into the designated register. Bits 0 and 1 of the first word specify, in binary form, the number of the register to be loaded. General purpose registers DR0 through DR3 are designated by 00, 01, 10, and 11, respectively.

Operation: (DR#) ← DATA

LDSP LOAD STACK POINTER Octal code: 0062XX dddddd



LDSP is a two-word instruction used to load an address into the graphic controller stack pointer. The first word identifies the instruction; the second word contains the memory address to be loaded into the graphic controller stack pointer. The graphic controller stack is accessed in the page defined by the page register. (See PGR in Section 4.)

Operation: (DSP) ← ADDRESS

LDRI — LOAD DEVICE REGISTER IMMEDIATE — Octal code: 0060dd XXdddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	0	0	DEV#			REG#		
X	X	X	X	DATA											

LDRI is a two-word instruction used to load data into a register in an optional device that may be connected to the graphic bus. The first word identifies the instruction and identifies the device and register into which data is to be loaded. Bits 3-5 specify the binary number assigned to the device while bits 0-2 specify the binary number assigned to the register within the device. Bits 0-11 of the second word contain the data to be loaded into the designated register; bits 12-15 of the second word are not used.

NOTE

For example, the optional 2-D coordinate converter card is designated as device 0. LDRI instructions are used to program the 2-D coordinate converter as described in Sanders publication H-78-0061.

Operation: (DEV# REG#) ← DATA

ADDI — ADD TO DISPLAY REGISTER IMMEDIATE — Octal code: 00430d dddddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	1	0	0	0	0	DR#	
±	DATA														

ADDI is a two-word instruction that enables a numerical value to be added to or subtracted from the contents of a general purpose register. The first word identifies the instruction and identifies the display register that contains the data to be modified. The second word, in two's complement form, contains the numerical value to be added (or subtracted from) the designated register. Bits 0 and 1 of the first word identify, in binary form, the number of the register containing the data to be modified. General purpose registers DR0 through DR3 are designated by 00, 01, 10, and 11, respectively.

Operation: (DR#) ← (DR#) + DATA

SAVD

SAVE DISPLAY REGISTER

Octal code: 00410d

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	DR#	

The SAVD instruction causes the contents of the general purpose register identified by bits 0 and 1 to be pushed onto the top of the graphic controller stack. Before the push operation, the graphic controller stack pointer is decremented by two. Bits 0 and 1 identify the number of the register in binary form. General purpose registers DR0 through DR3 are designated by 00, 01, 10, and 11, respectively.

Operation: (DSP) ← (DSP) - 2
(Top stack location) ← (DR#)

RESD

RESTORE DISPLAY REGISTER

Octal code: 00420d

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	DR#	

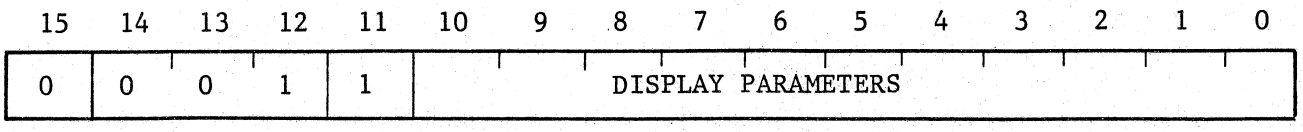
The RESD instruction causes the contents of the top of the graphic controller stack to be popped and placed into the general purpose register identified by bits 0 and 1. Following the pop operation, the graphic controller stack pointer is incremented by two. Bits 0 and 1 identify the number of the register in binary form. General purpose registers DR0 through DR3 are designated by 00, 01, 10, and 11, respectively.

Operation: (DR#) ← (Top stack location)
(DSP) ← (DSP) + 2

3.3.4 DISPLAY CONTROL INSTRUCTIONS

Display control instructions are used to establish and/or change various display parameters as required. An initialize instruction is also included to permit definite hardware conditions to be established before a refresh file is processed.

LDDP LOAD DISPLAY PARAMETER REGISTER Octal code: 014ddd



LDDP is used to modify the contents of the display parameter register. The action of individual bits is as follows:

Bit(s)	Action
0,1	Character size: 1 0 0 0 = size 0 (smallest) 0 1 = size 1 (1.5 times size 0) 1 0 = size 2 (2.0 times size 0) 1 1 = size 3 (3.0 times size 0)
2	Character orientation: 0 = normal 1 = rotate 90° ccw
3	Character parameter change enable: 0 = no change 1 = change character size and/or orientation status to that indicated by bits 0-2
4,5	PHOTOPEN select: 5 4 X 1 = PHOTOPEN 1 enabled 1 X = PHOTOPEN 2 enabled

NOTE

The PHOTOPEN select bits can be used to sensitize and desensitize selected objects of the display image. PHOTOPEN strike interrupts can be generated for sensitized objects. For example, if the display image were to consist of a box inscribed in a triangle, the box could be sensitized for the detection of PHOTOPEN interrupts and the triangle could be desensitized so that no PHOTOPEN interrupts could be generated when the PHOTOPEN is pointed at the triangle. The refresh code would be blocked off as follows:

```
LDDP          ;enable PHOTOPENS
  .
  .
  .           ;code to DISPLAY BOX
  .
  .
  .
LDDP          ;disable PHOTOPENS
  .
  .
  .           ;code to DISPLAY TRIANGLE
  .
  .
  .
LDDP          ;re-enable PHOTOPENS
```

Bit(s)	Action
6	PHOTOPEN select change enable: 0 = no change 1 = change PHOTOPEN select status to that indicated by bits 4,5
7,8	Frame sync select: 8 7 0 0 = no change 0 1 = 60 Hz 1 0 = 40 Hz 1 1 = 30 Hz

NOTE

To insure proper display operation at a known frame sync rate, an LDDP instruction should be used to set bits 7 and 8 for the required rate before a WATE instruction is used.

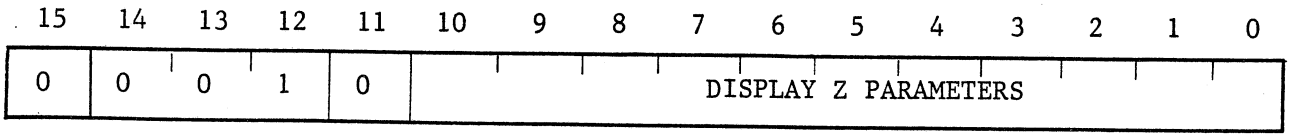
9	Writing speed select: 0 = fast 1 = slow
10	Writing speed change enable: 0 = no change 1 = change writing speed select status to that indicated by bit 9

Operation: (DPR) ← DISPLAY PARAMETERS

LDDZ

LOAD DISPLAY Z REGISTER

Octal code: 01dddd

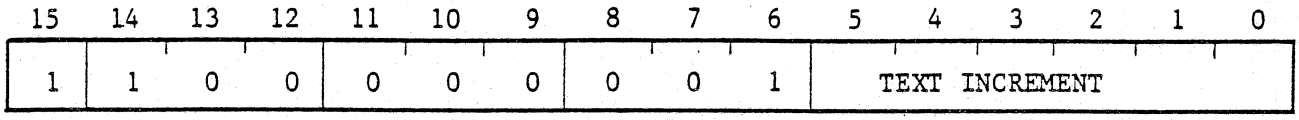


LDDZ is used to modify the contents of the display Z register. The action of individual bits is as follows:

Bit(s)	Action
0-2	Gray level select: 2 1 0 0 0 0 = intensity level 0 (off) 0 0 1 = intensity level 1 thru thru 1 1 1 = intensity level 7 (brightest)
3,4	Line structure select: 4 3 0 0 = solid vector 0 1 = dotted vector 1 0 = dashed vector 1 1 = dot-dashed vector (center line)
5	Blink select: 0 = steady 1 = blink
6-9	Display select: 9 8 7 6 1 X X X = Display no. 1 Z axis enabled X 1 X X = Display no. 2 Z axis enabled X X 1 X = Display no. 3 Z axis enabled X X X 1 = Display no. 4 Z axis enabled
10	Display select change enable: 0 = no change 1 = change display select status to that indicated by bits 6-9

Operation: (DZR) ← DISPLAY Z PARAMETERS

LDTI — LOAD TEXT INCREMENT REGISTER — Octal code: 1401dd



LDTI is used in conjunction with the TXT (draw two tabular characters) instruction to specify the amount by which the CRT beam position is incremented after each character is drawn. For normal characters, the contents of the X-position register is incremented. For rotated characters, the contents of the Y position register is incremented. The text increment specified by bits 0 through 5 replaces the contents of the test increment register. Text increments (octal) recommended for character sizes 0 through 3 are 12, 17, 24, and 36, respectively.

Operation: Normal characters

(DTI) ← TEXT INCREMENT

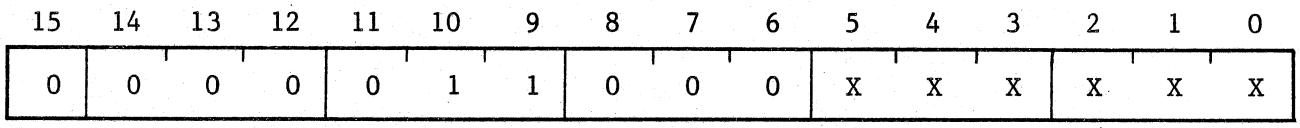
(DXR) ← (DXR) + TEXT INCREMENT (following display of each character)

Rotated characters:

(DTI) ← TEXT INCREMENT

(DYR) ← (DYR) + TEXT INCREMENT (following display of each character)

IZPR — INITIALIZE — Octal code: 0030XX



IZPR is used to reset a flip-flop on the ramp generator card so that a known status of the hardware circuits can be established. An IZPR instruction should always be used as the first instruction of a refresh file to ensure that no variations in the displayed image are introduced by the hardware.

Operation: Reset toggle flip-flop on ramp generator card

SECTION 4
GRAPHIC 7 REGISTERS

4.1 GENERAL

GRAPHIC 7 registers fall into three major categories: display processor registers, graphic controller registers, and interface registers. This section describes the application and format of each register in each category and identifies the address assigned to each. A summary of the data contained in this section is provided in Appendix A.

4.2 DISPLAY PROCESSOR REGISTERS

The display processor contains eight general registers designated R0 through R7. These registers function in a manner similar to the corresponding registers in a minicomputer of the PDP-11 type manufactured by Digital Equipment Corporation (DEC). Details concerning the applications and formats of these registers are contained in the DEC PDP-11/04/34/45/55/60 Processor Handbook which should be used as a supplement to this manual. Note, however, that addresses are not assigned to the display processor registers.

4.3 GRAPHIC CONTROLLER REGISTERS

Graphic controller registers can be divided into four groups: processor registers, function registers, sense and mask registers, and function control registers. The following paragraphs provide details concerning the application, format, and address of each register of each group. Refer to Appendix A for a summary of the data applicable to graphic controller registers.

NOTE

Except for the sense and mask, and the function control registers, all graphic controller registers are 16-bit registers. In several cases, however, fewer than 16 bits are used. The descriptions in the following paragraphs consider the size of each register to be equal to the number of bits used.

4.3.1 PROCESSOR REGISTERS

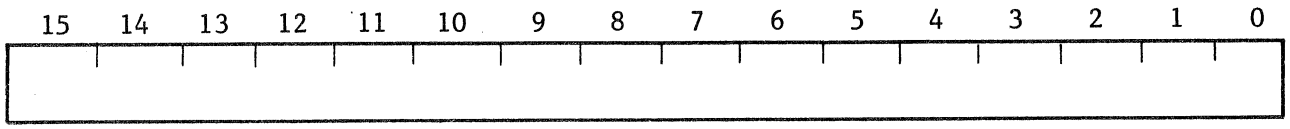
Processor registers of the graphic controller comprise four general purpose registers, a stack pointer, a program counter, and an instruction register. These registers are the general working registers of the graphic controller. Each has an octal address and, when the graphic controller is halted, the contents of each can be read by the display processor using programmed data transfers.

DR0

thru

DR3

GENERAL PURPOSE REGISTER n Octal address: 165002 (DR0)
165004 (DR1)
165032 (DR2)
165034 (DR3)



Each of the four general purpose registers (display registers) in the graphic controller is a 16-bit register than can be used as required for general operations or for temporary storage of data. Additionally, the contents of DR0 can be tested and a jump executed if the value is not equal to zero. Data is written into the general purpose registers using graphic controller instructions.

Associated instructions: DR0 thru DR3:

LDDI SAVD

ADDI RESD

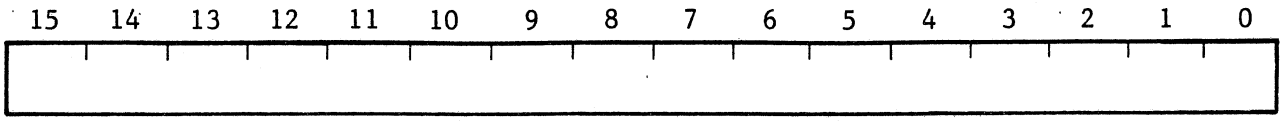
DR0 only:

JMPZ JPRZ

DSP

STACK POINTER

Octal address: 165000



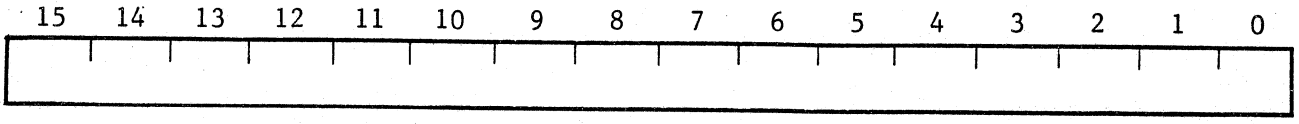
The stack pointer is a 16-bit register that contains the address of the top location in the memory stack. It is loaded by the graphic controller LDSP instruction. When a SAVD instruction is used to push data onto the stack, the contents of the stack pointer is automatically decremented by two before the push operation occurs. When a RESD instruction is used to pop data from the stack, the contents of the stack pointer is automatically incremented by two following the pop operation. Call and return instructions make similar use of the stack pointer to save and restore the contents of the program counter when a subroutine is performed.

Associated instructions:	LDSP	CALL
	SAVD	CALR
	RESD	RTRN

DPC

PROGRAM COUNTER

Octal address: 165006



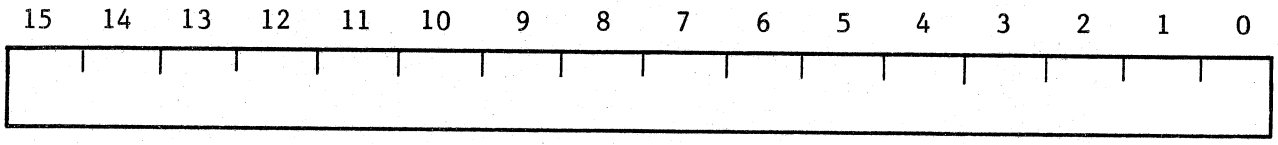
The program counter is a 16-bit register that contains the address of the next instruction to be executed by the graphic controller. The program counter is initially loaded by the display processor with the starting address of a refresh file. This automatically starts the graphic controller. As instructions are executed by the graphic controller, the contents of the program counter is incremented automatically. A one-word instruction causes the contents to be incremented by two while a two-word instruction causes the contents to be incremented by four (bit 0 is always zero). For this reason, increments used for relative jumps or calls must be calculated from the address immediately following the location of the jump or call instruction.

Associated instructions: All

DIR

DISPLAY INSTRUCTION REGISTER

Octal code: 165010

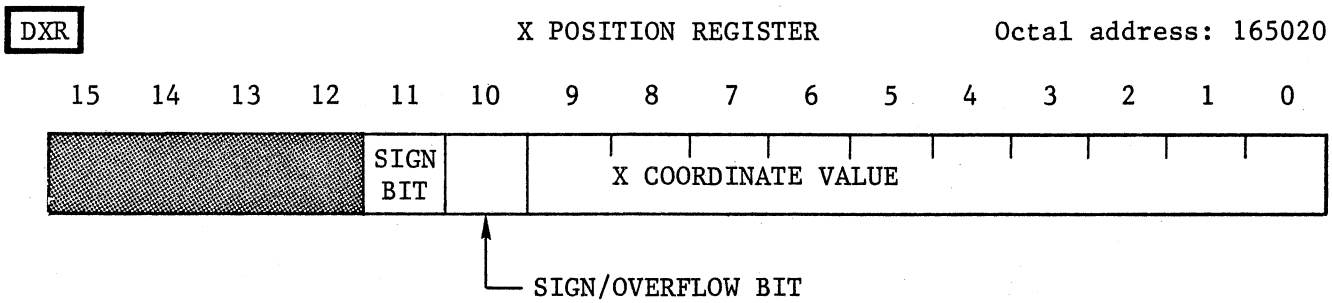


The display instruction register is a 16-bit register into which each instruction or data word fetched by the graphic controller is placed.

Associated instructions: All

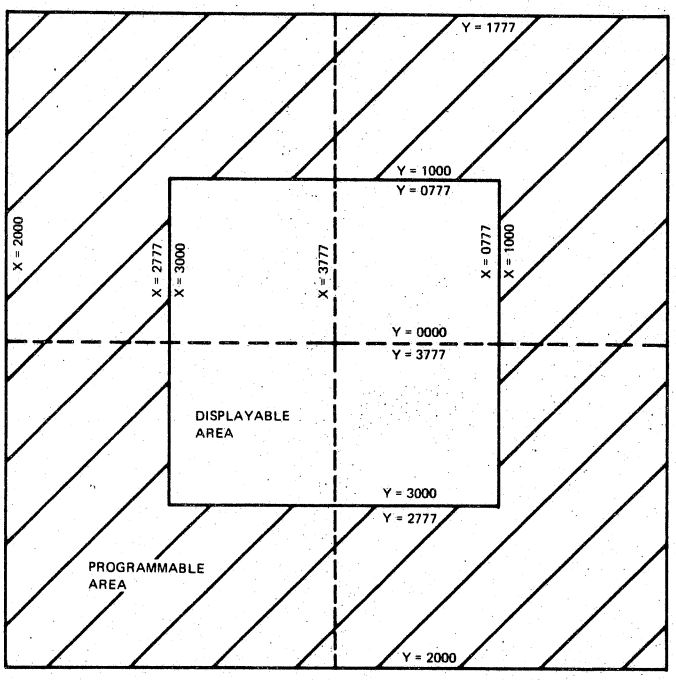
4.3.2 FUNCTION REGISTERS

Function registers in the graphic controller correspond to registers located on circuit cards that are connected to the graphic bus. These registers are loaded by graphic controller instructions as required to control the functions performed by the circuit cards. Each function register has an octal address so that, when the graphic controller is halted, the contents of the registers can be read by the display processor using programmed data transfers.



The position register is a 12-bit register that contains the value of the X coordinate of the CRT beam position. When a graphic controller loads absolute data into the X position register, the 11 bits that specify the value of the X coordinate are sign extended to fill the 12 bits of the register. When an instruction specifying relative X position data is executed, the specified data is added to the contents of the X position register. Bit 10 serves as an indicator of overflow condition. Whenever the addition of relative data causes the value in the X position register to exceed programmable limits, bit 10 will differ from bit 11. Under these conditions, if the X/Y overflow bit in the mask register (MKR) is set, the graphic controller will halt and interrupt the display processor. If the X/Y overflow bit is not set, relative data will still modify the register contents but the CRT beam will be blanked until bits 10 and 11 are no longer different. Coordinate values are expressed in two's complement form and may range from 1777_8 (+1023) to 2000_8 (-1024). The zero X coordinate defines the vertical center line of the CRT screen. Positive coordinates are to the right of center; negative coordinates are to the left of center. Note that only the values from 0777_8 (+511) to 3000_8 (-512) fall into the displayable area of the CRT. Values outside these limits cause the CRT beam to be blanked (see figure 4-1).

Associated instructions: LDXA MVXA PPLR
LDXR MVXR TXT (for normally oriented characters)
DRXA DRSR
DRXR MVSR



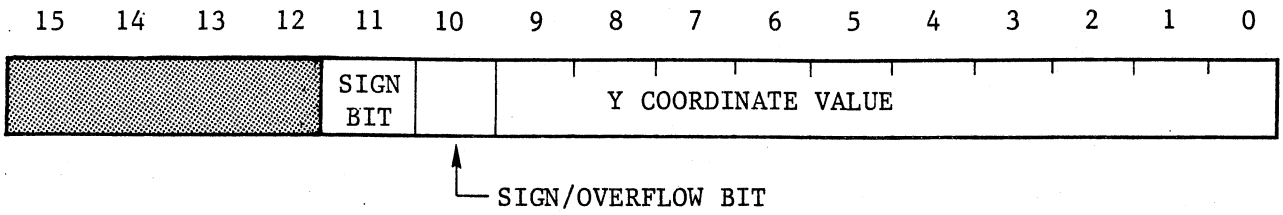
GA-77-419-05

NOTE

Coordinate designations are in octal format

Figure 4-1 CRT Programmable vs. Displayable Areas.

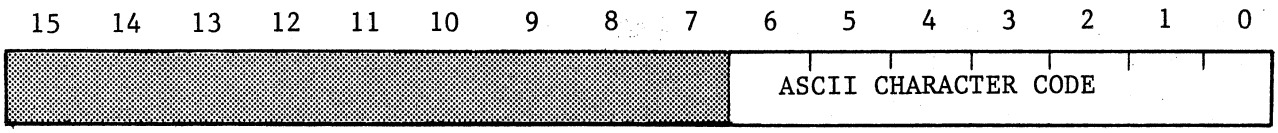
DYR — Y POSITION REGISTER — Octal address: 165022



The Y position register is a 12-bit register that contains the value of the Y coordinate of the CRT beam position. This register is identical to the X position register and performs the same functions for Y coordinate data that the X position register performs for X coordinate data. The X/Y overflow bit in the mask register (MKR) is applicable to the Y as well as the X position register. The zero Y coordinate defines the horizontal center line of the CRT screen. Positive coordinates are above the center; negative coordinates are below the center.

Associated instructions: DRYA DRSR
DRYR MVSR
MVYA PPLR
MVYR TXT (for rotated characters)

DCR — DISPLAY CHARACTER REGISTER — Octal address: 165024



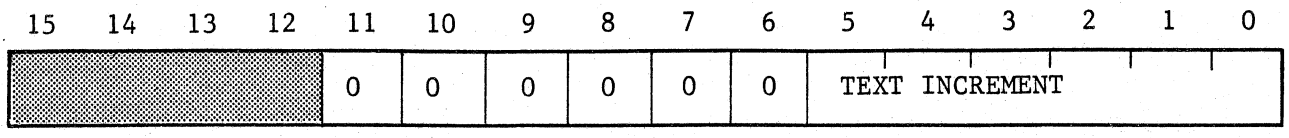
The display character register is a seven-bit register that contains the code of the character or symbol to be displayed. ASCII codes are used for standard and optional characters and symbols as shown in Appendix A.

Associated instructions: TXT
CHAR

DTI

TEXT INCREMENT REGISTER

Octal address: 165012



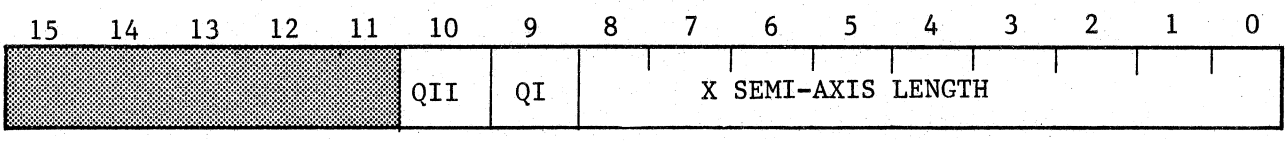
The text increment register is a 12-bit register that contains the value by which the CRT beam position is to be incremented after each tabular character is displayed. Bits 0-5 may be programmed as required; bits 6-11 are always zero. After a normally oriented character is drawn, the value in the text increment register is added to the value in the X position register. After a rotated character is drawn, the text increment value is added to the value in the Y position register. Note that this register is associated only with the TXT (draw two tabular characters) instruction. No automatic incrementing of the CRT beam position occurs when the CHAR (draw single character) is used.

Associated instructions: TXT

KXR

CONIC X DATA REGISTER (OPTIONAL)

Octal code: 165026



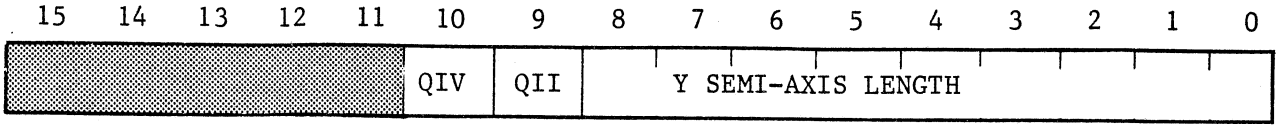
The conic X data register is associated with the optional conic generator card. It is an 11-bit register that contains the value of the length of the X semi-axis (distance from the ellipse center to its perimeter on the X axis) for an ellipse to be displayed. The X semi-axis length is contained in bits 0-8. Bits 9 and 10, respectively, designate CRT beam unblanking for quadrants I (upper right) and III (lower left). A zero indicates the beam is to be blanked while a one indicates the beam is to be unblanked. Loading this register does not initiate motion of the CRT beam. Normally, this register is loaded by a LDKX (load conic X register) instruction. If a DRKY (draw conic Y register) instruction is not preceded by a LDKX instruction, the data specified by the LDKY instruction will be loaded into both the conic X and the conic Y data registers.

Associated instructions: LDKX
DRKY (when not preceded by LDKX)

KYR

CONIC Y DATA REGISTER (OPTIONAL)

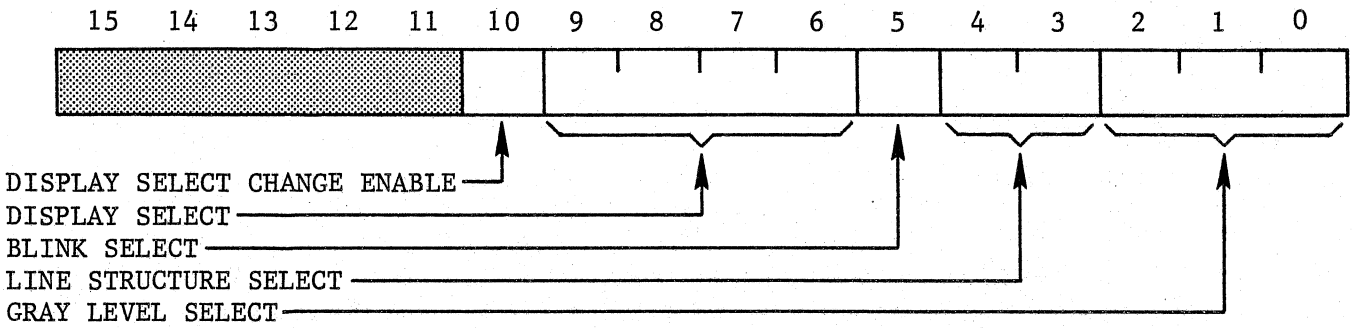
Octal address: 165030



The conic Y register is associated with the optional conic generator card. It is an 11-bit register that contains the value of the length of the Y semi-axis (distance from the ellipse center to its perimeter on the Y axis) for an ellipse to be displayed. The Y semi-axis length is contained in bits 0-8. Bits 9 and 10, respectively, designate CRT beam unblanking for quadrants II (upper left) and IV (lower right). A zero indicates the beam is to be blanked while a one indicates the beam is to be unblanked. Loading this register initiates CRT beam motion in accordance with the data in the conic X and the conic Y data registers (the ellipse center is defined by data in the X and Y position registers). Data is loaded into the conic Y data register using a DRKY instruction.

Associated instructions: DRKY

DZR DISPLAY Z REGISTER Octal address: 165016



The display Z register is an 11-bit register containing data that controls the Z-axis parameters of the associated display indicators. The action of the individual bits is as follows:

Bit(s)	Action
0-2	Gray level select: 2 1 0 0 0 0 = intensity level 0 (off) 0 0 1 = intensity level 1 thru 1 1 1 = intensity level 7 (brightest)
3,4	Line structure select: 4 3 0 0 = solid vector 0 1 = dotted vector 1 0 = dashed vector 1 1 = dot-dashed vector (center line)
5	Blink select: 0 = steady 1 = blink

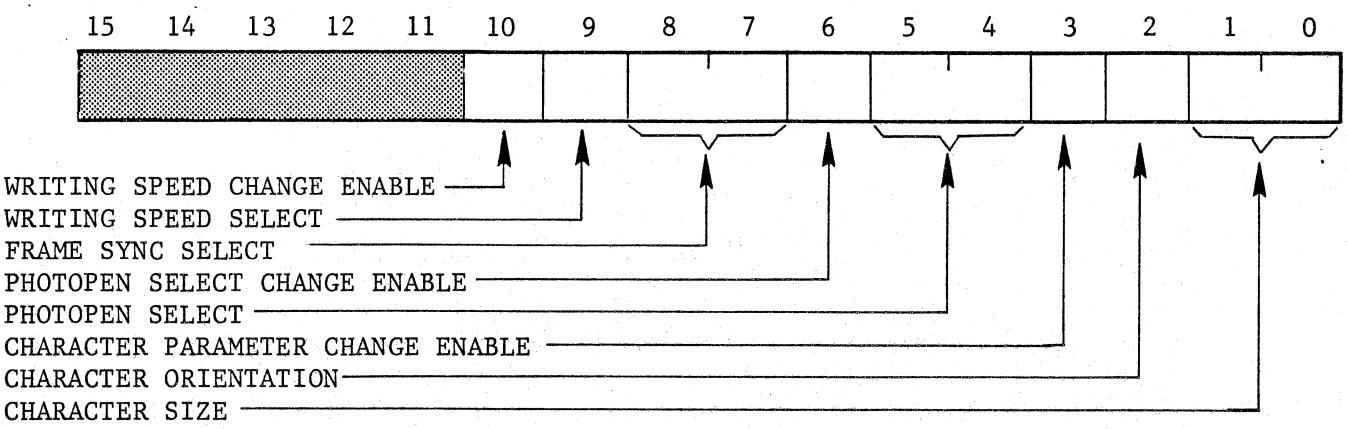
Bit(s)	Action
6-9	Display select: 9 8 7 6 1 X X X = Display no. 1 Z axis enabled X 1 X X = Display no. 2 Z axis enabled X X 1 X = Display no. 3 Z axis enabled X X X 1 = Display no. 4 Z axis enabled
10	Display select change enable: 0 = no change 1 = change display select status to that indicated by bits 6-9

Associated instructions: LDDZ

DPR

DISPLAY PARAMETER REGISTER

Octal code: 165014



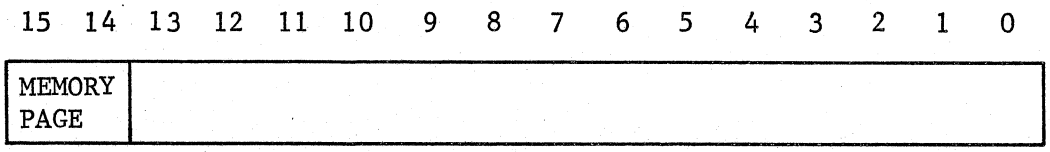
The display parameter register is an 11-bit register containing data that controls various parameters of the associated display indicators. The action of the individual bits is as follows:

Bit(s)	Action
0,1	Character size: 1 0 0 0 = size 0 (smallest) 0 1 = size 1 (1.5 times size 0) 1 0 = size 2 (2.0 times size 0) 1 1 = size 3 (3.0 times size 0)
2	Character orientation: 0 = normal 1 = rotate 90° ccw
3	Character parameter change enable: 0 = no change 1 = change character size and/or orientation status to that indicated by bits 0-2
4,5	PHOTOPEN select: 5 4 X 1 = PHOTOPEN 1 enabled 1 X = PHOTOPEN 2 enabled

Bit(s)	Action
6	PHOTOPEN select change enable: 0 = no change 1 = change PHOTOPEN select status to that indicated by bits 4,5
7,8	Frame sync select: 8 7 0 0 = no change 0 1 = 60 Hz 1 0 = 40 Hz 1 1 = 30 Hz
	NOTE Frame sync select bits should be programmed for a condition other than 00 before a WATE instruction is executed.
9	Writing speed select: 0 = fast 1 = slow
10	Writing speed change enable: 0 = no change 1 = change writing speed select status to that indicated by bit 9

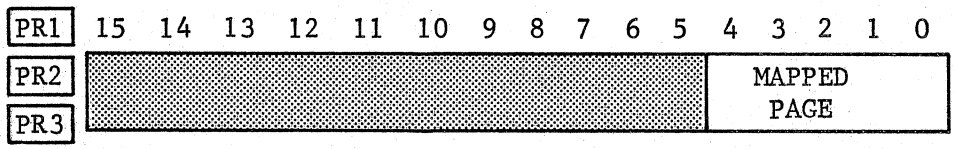
Associated instructions: LDDP

PGR GRAPHIC CONTROLLER PAGE REGISTER Octal Address: 165014



The graphic controller page register is a two-bit register used to extend the memory addressing capability of the graphic controller up to a total of 131,072 (128K) words.

DISPLAY PROCESSOR PAGE REGISTERS



Octal Address: 172342
Octal Address: 172344
Octal Address: 172346

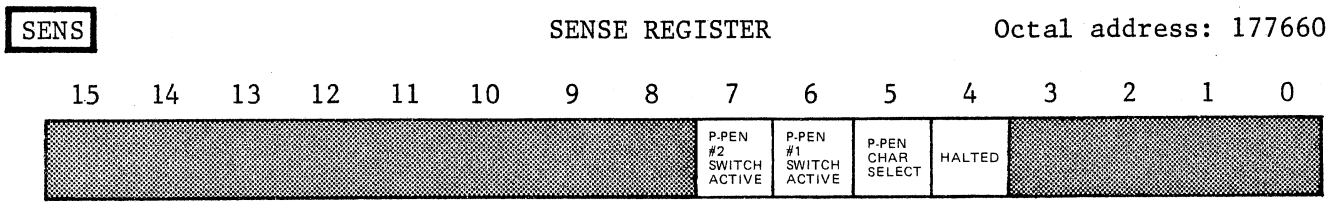
The display processor page registers are 5-bit registers that are used to extend the memory addressing capability of the display processor up to a total of 131,072 (128K) words.

NOTE

Refer to Sanders publication H-78-0408 for detailed information on programming the page registers of the GRAPHIC 7.

4.3.3 SENSE AND MASK REGISTERS

One sense and one mask register are associated with the graphic controller. Both registers are assigned octal addresses and may be read at any time by the display processor using programmed data transfers. Programmed data transfers may also be used by the display processor at any time to write data into the mask register.



The sense register is a four-bit register that indicates PHOTOPEN activity and the halt status of the graphic controller. Bits in the register are designated 4 through 7 (bits 0-3 and 8-15 are not used). The condition indicated by each bit when it is set to 1 is as follows:

Bit	Condition Indicated When Set to 1
4	Graphic controller is halted for one of the following reasons: 1) Display processor sends stop function code (165040) to graphic controller 2) Display processor executes RESET instruction to initialize devices on controller bus 3) HREF instruction executed by graphic controller 4) LINK instruction executed by graphic controller 5) Invalid instruction executed by graphic controller 6) Bus timeout (memory fails to respond to a fetch command) 7) PHOTOPEN strike (PHOTOPEN pointing to a display element and the corresponding PHOTOPEN strike bit in the mask register is set to 1) 8) X or Y position overflow (Bits 10 and 11 in the X position or the Y position register are different and the X/Y overflow bit in the mask register is set to 1)

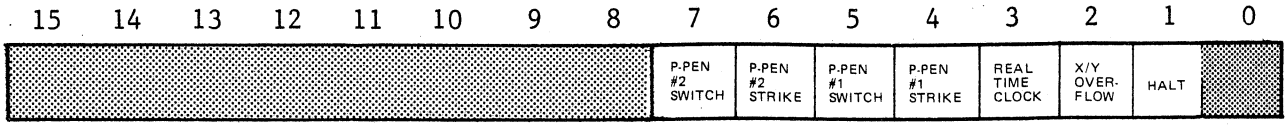
Bit	Condition Indicated When Set to 1
5	For a PHOTOPEN strike associated with tabular characters, the PHOTOPEN is pointing at the second character in a TXT instruction (bits 8-14)
6	Switch of PHOTOPEN 1 is actuated
7	Switch of PHOTOPEN 2 is actuated

Associated instructions: Display processor programmed data transfer (read only)

MKR

MASK REGISTER

Octal address: 177662



The mask register is a seven-bit register on the ROM and Status card that enables the graphic controller to report conditions to the display processor on an interrupt basis. An interrupt occurs when the condition is met and the corresponding bit in the mask register is set to 1. Bits in the mask register can be set or cleared as required by the display processor using programmed data transfers. Additionally, programmed data transfers may be used at any time by the display processor to read the contents of the mask register. Seven different interrupts are enabled or inhibited by bits 1 through 7 (bits 0 and 8-15 are not used). The interrupt and the interrupt vector address associated with each bit are as follows:

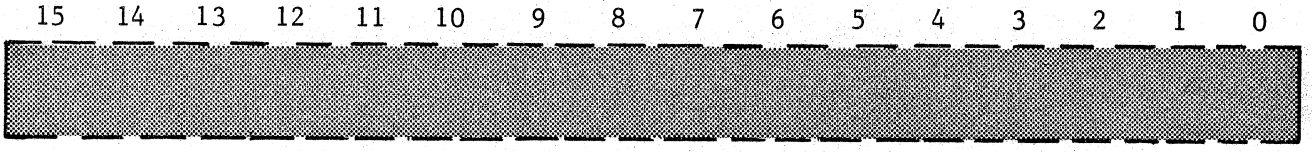
Bit	Associated Interrupt	Interrupt Vector Address (octal)
1	Graphic controller halted	000140
2	X or Y position overflow (bits 10 and 11 in X position or Y position register are different)	000144
3	Real time clock (interrupts at rate of 60 Hz)	000100
4	PHOTOPEN 1 strike	000150
5	Switch of PHOTOPEN 1 actuated	000160
6	PHOTOPEN 2 strike	000154
7	Switch of PHOTOPEN 2 actuated	000160

Associated instructions: Display processor programmed data transfers (read or write)

4.3.4 FUNCTION CONTROL REGISTERS

Two function control registers are associated with the graphic controller. These registers are actually only addresses that may be accessed by the display processor. Simply by accessing function control register addresses, the display processor can halt or restart the graphic controller as required.

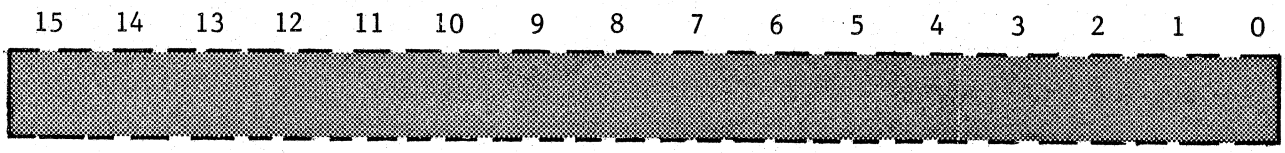
FUNS FUNCTION CONTROL STOP REGISTER Octal address: 165040



The function control stop register is a function control register used to halt the graphic controller. Whenever the display processor accesses the address assigned to this register, the graphic controller halts.

Associated instructions: Accessing address 165040 by display processor

FUNC FUNCTION CONTROL CONTINUE REGISTER Octal address: 165036



The function control continue register is a function control register used to restart the graphic controller after it has been halted. Whenever the display processor accesses the address assigned to this register, the graphic controller resumes processing from the point at which it last halted. Note that this register should not be accessed unless the graphic controller is halted.

Associated instructions: Accessing address 165036 by display processor

4.4 INTERFACE REGISTERS

Interface registers are associated with the various serial and parallel interface ports of the GRAPHIC 7. The following paragraphs provide details concerning the format and the address assigned to each interface register. Refer to Appendix A for a summary of the data applicable to the interface registers.

4.4.1 SERIAL INTERFACE REGISTERS

Up to nine serial interface ports are available for external devices to communicate with the GRAPHIC 7. One is located on the ROM and status logic card and four are located on each multiport serial interface card (two multiport serial interface cards may be installed in the terminal controller). The designations of these ports and the associated devices are as follows (note that ports 1 and 5 can be used either as basic serial interface ports or as full RS-232C interface ports):

Port Designation	Associated Device	Location
1 (RS-232C)	Host computer	Multiport serial interface card no. 1
2	Unused	
3	Alphanumeric/Function Keyboard no. 1	
4	PED no. 1	
5 (RS-232C)	Hard copy unit	Multiport serial interface card no. 2
6	Unused	
7	Alphanumeric/Function Keyboard No. 2	
8	PED no. 2	
TTY	Teletypewriter	ROM and status logic card

Four registers are associated with each serial interface port: a receive status register, a receive data buffer, a transmit status register, and a transmit status register. Mnemonics for serial interface ports are suffixed with the number of the associated port. No suffix is used for TTY port registers. Each register has an octal address and can be accessed as required by the display processor.

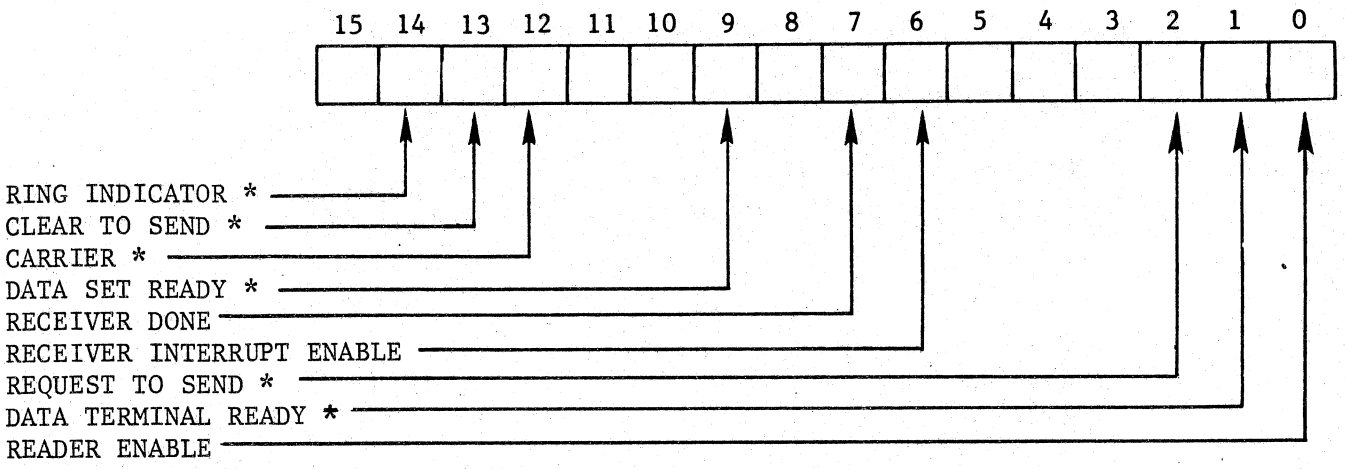
NOTE

With respect to serial interface registers, receive data is data received from the external device. Transmit data is data transmitted to the external device.

RSRn

RECEIVE STATUS REGISTER N

Octal address: 176500 (RSR1)
176510 (RSR2)
176520 (RSR3)
176530 (RSR4)
176540 (RSR5)
176550 (RSR6)
176560 (RSR7)
176570 (RSR8)
177560 (TTYRSR)



NOTES

1. Unidentified bits are not used.
2. Bits marked with an asterisk (*) are used on full RS-232C interface ports (ports 1 and 5) only.

The following receive status register bits are used on all serial interface ports:

Bit	Function	Remarks
0	Reader enable	1. Program write (set) only 2. Cleared by controller bus reset 3. Cleared when start bit received 4. When set, places ground on pin 9 of 10-pin I/O connector
6	Receiver interrupt enable	1. Program read/write 2. Cleared by controller bus reset 3. When set, a display processor interrupt is generated when data ready (bit 7) is set

Bit	Function	Remarks
7	Data ready	1. Program read only 2. Cleared by controller bus reset 3. Set when receiver has transferred a character into associated receive data buffer (RDBn) 4. Cleared by setting reader enable (bit 0) or by reading RDBn 5. If receiver interrupt enable (bit 6) is set, setting this bit cuases display processor interrupt to be generated 6. Interrupt trap addresses (octal) for each register are: RSR1 - 000300 RSR5 - 000340 RSR2 - 000310 RSR6 - 000350 RSR3 - 000320 RSR7 - 000360 RSR4 - 000330 RSR8 - 000370 TTYRSR - 000060

The following receive status register bits are used on full RS-232C interface ports (ports 1 and 5) only:

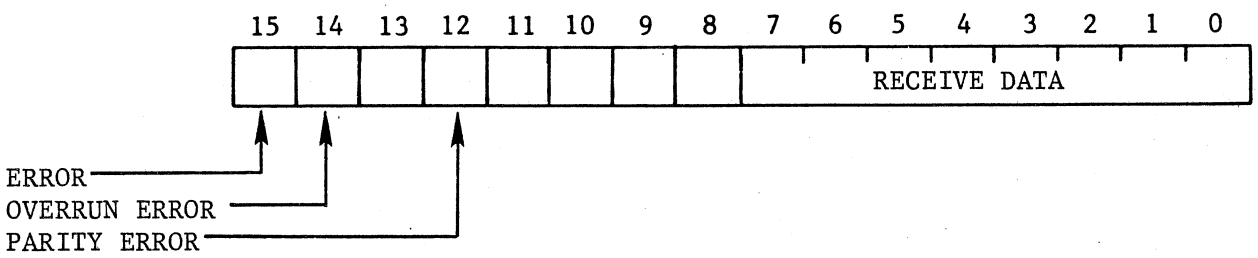
Bit	Function	Remarks
1	Data terminal ready	1. Program read/write 2. Cleared by controller bus reset 3. Status of this bit is placed on pin 15 of 26-pin I/O connector
2	Request to send	1. Program read/write 2. Cleared by controller bus reset 3. Status of this bit is placed on pin 7 of 26-pin I/O connector
9	Data set ready	1. Program read only 2. Status of this bit reflects level at pin 11 of 26-pin I/O connector
12	Carrier	1. Program read only 2. Status of this bit reflects level at pin 16 of 26-pin I/O) connector

Bit	Function	Remarks
13	Clear to send	1. Program read only 2. Status of this bit reflects level at pin 9 of 26-pin I/O connector
14	Ring indicator	1. Program read only 2. Status of this bit reflects level at pin 19 of 26-pin I/O connector 3. A jumper option permits a high ring indicator input at pin 19 to initialize the GRAPHIC 7 in the system mode.

RDBn

RECEIVE DATA BUFFER n

Octal address: 176502 (RDB1)
176512 (RDB2)
176522 (RDB3)
176532 (RDB4)
176542 (RDB5)
176552 (RDB6)
176562 (RDB7)
176572 (RDB8)
177562 (TTYRDB)



NOTES

1. Unidentified bits are not used
2. Parity error (bit 12) is used on full RS-232C interface ports (ports 1 and 5) only

The following receive data buffer bits are used on all serial interface ports:

Bit(s)	Function	Remarks
0 thru 7	Receive data	1. Program read only 2. These bits contain last serial character received 3. If character is less than 8 characters, unused high-order bits will be zeros
14	Overrun error	1. Program read only 2. Cleared by controller bus reset 3. Updated each time a character is received 4. This bit is set when a new character is received before preceding character is read by program

Bit(s)	Function	Remarks
15	Error	1. Program read only 2. Cleared only when parity error (bit 12) and overrun error (bit 14) are cleared 3. This bit is set whenever parity error (bit 12) or overrun error (bit 14) is set (parity error is used only on full RS-232C ports)

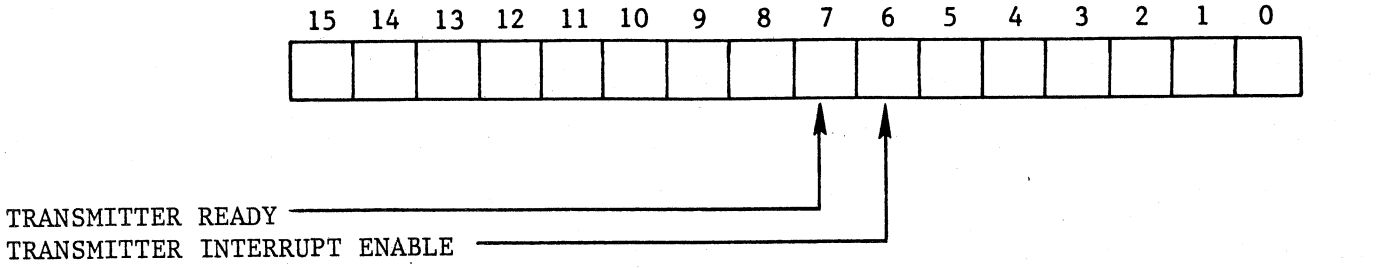
The following receive data buffer bit is used on full RS-232C interface ports (ports 1 and 5) only:

Bit	Function	Remarks
12	Parity Error	1. Program read only 2. Cleared by controller bus reset 3. Cleared when buffer is read 4. Updated each time a new character is received 5. This bit is set when the receiver detects a parity error in the character received

TSRn

TRANSMIT STATUS REGISTER n

Octal address: 176504 (TSR1)
176514 (TSR2)
176524 (TSR3)
176534 (TSR4)
176544 (TSR5)
176554 (TSR6)
176564 (TSR7)
176574 (TSR8)
177564 (TTYTSR)



NOTE

Unidentified bits are not used.

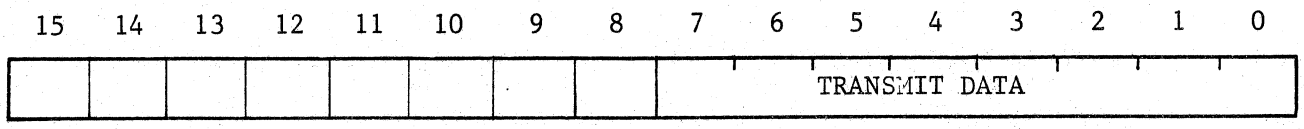
Bits in the transmit status register function as follows (the bits are used on all serial interface ports):

Bit	Function	Remarks
6	Transmitter interrupt enable	1. Program read/write 2. Cleared by controller bus reset 3. When set, a display processor interrupt is generated when transmitter ready (bit 7) is set 4. Interrupt trap addresses (octal) for each register are: TSR1 - 000304 TSR5 - 000344 TSR2 - 000314 TSR6 - 000354 TSR3 - 000324 TSR7 - 000364 TSR4 - 000334 TSR8 - 000374 TTYTSR - 000064
7	Transmitter ready	1. Program read only 2. Cleared by writing into associated transmit data buffer (TDBn) 3. This bit is set when first bit of character is presented to the line and TDBn is ready to accept another character

TDBn

TRANSMIT DATA BUFFER n

Octal address: 176506 (TDB1)
176516 (TDB2)
176526 (TDB3)
176536 (TDB4)
176546 (TDB5)
176556 (TDB6)
176566 (TDB7)
176576 (TDB8)
177566 (TTYTDB)



NOTE

Unidentified bits are not used.

Bits 0 through 7 in the transmit data buffer are program write only bits. They are loaded by the program with the code of the character to be transmitted to the external device. Bits 8 through 15 are not used.

4.4.2 PARALLEL INTERFACE REGISTERS

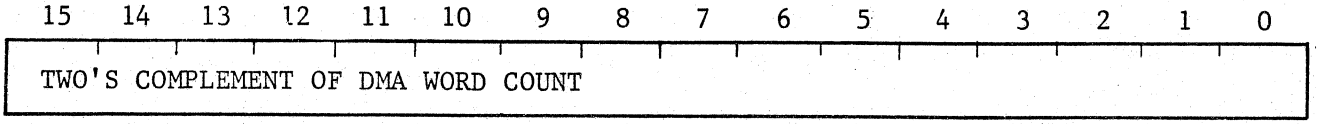
Up to four parallel interfaces can be used by external devices to communicate with the GRAPHIC 7; a separate card is required for each. Four registers are associated with each parallel interface. These are a word count register, a memory address register, a status register, and a data register. Each register has an octal address and can be accessed as required by the display processor. Mnemonics for the registers are suffixed with the number of the associated interface.

NOTE

Parallel interface ports are optional. Normally, if a parallel interface port is used, a single parallel interface card (for the host computer) is installed in the terminal controller. For special applications, however, up to four parallel interface cards may be installed.

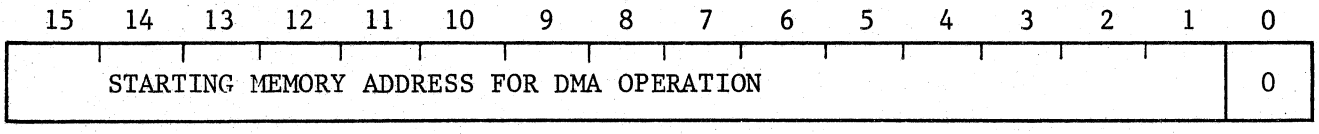
With respect to parallel interface registers, input data is data sent from the GRAPHIC 7 to the host computer; output data is data sent from the host computer to the GRAPHIC 7.

WCRn — WORD COUNT REGISTER n — Octal address: 172410 (WCR1), 172430 (WCR2), 172450 (WCR3), 172470 (WCR4)



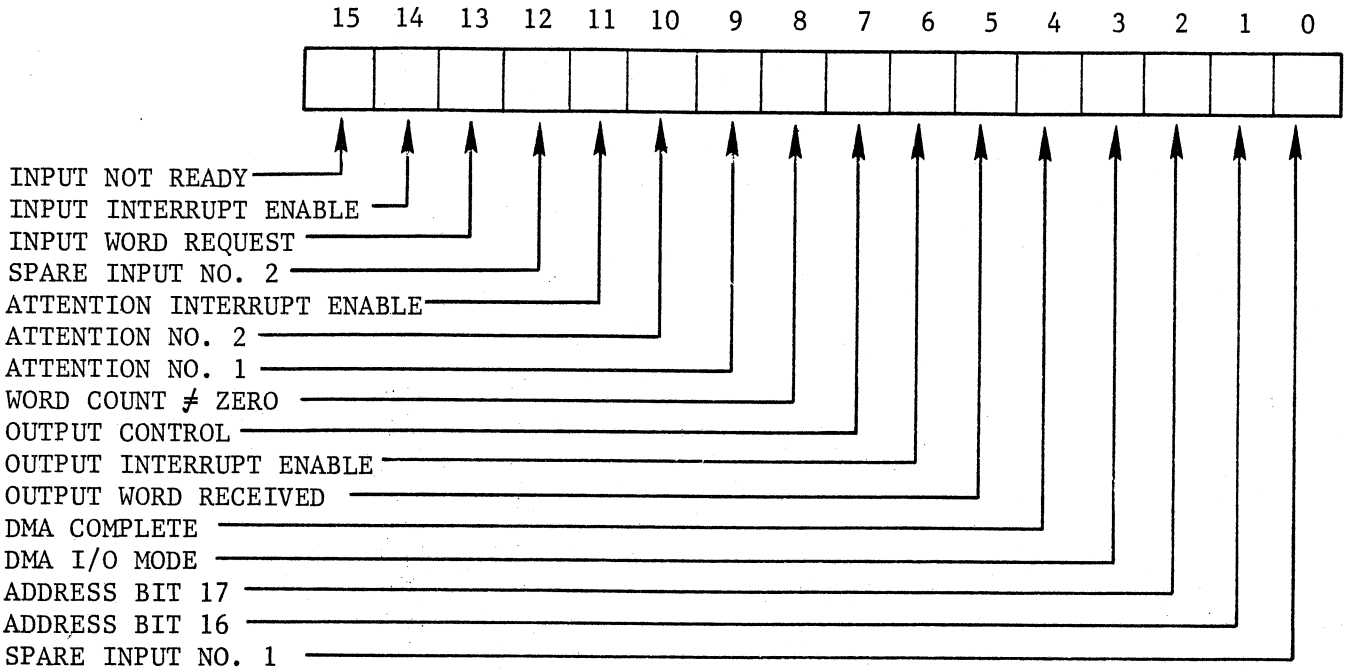
The word count register is a program read/write register used for direct memory access (DMA) operations. It is cleared by a controller bus reset. To initiate a DMA operation, the program writes into the word count register the two's complement of the number of memory words to be transferred between the GRAPHIC 7 and the host computer. Each time the parallel interface completes a DMA word transfer, the word count is incremented by one. The DMA operation continues until the word count equals zero at which time the interface generates an interrupt to the display processor (by setting the DMA complete bit in the associated status register).

MARn — MEMORY ADDRESS REGISTER n — Octal address: 172412 (MAR1), 172432 (MAR2), 172452 (MAR3), 172472 (MAR4)



The memory address register is a program read/write register used for direct memory access (DMA) operations. It is cleared by a controller bus reset. Before initiating a DMA operation, the program writes into the memory address register the memory address of the first word to be transferred between the GRAPHIC 7 and the host computer. This address must be the address of an even-numbered byte. Each time the parallel interface completes a DMA word transfer, the address in the memory address register is incremented by two bytes.

STRn STATUS REGISTER n Octal address: 172414 (STR1)
172434 (STR2)
172454 (STR3)
172474 (STR4)



The status register contains the necessary control and status bits to operate the parallel interface in either a DMA mode or a program control mode. The function of each bit is as follows:

Bit	Function	Remarks
0	Spare input no. 1	1. Program read/write 2. Cleared by controller bus reset 3. The status of this bit is directly presented to the host computer for programming as required
1	Address bit 16	1. Program read/write 2. Cleared by controller bus reset 3. This bit and address bit 17 (bit 2) are used in conjunction with the address in the memory address register (MARn) to expand the DMA addressing capability to 128K words

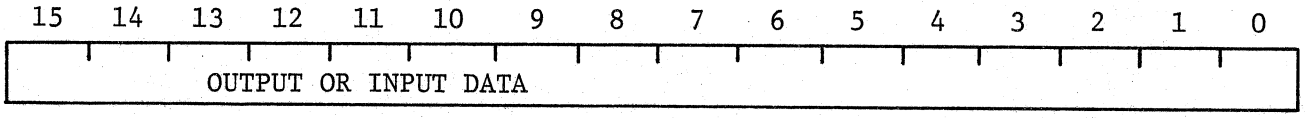
Bit	Function	Remarks
2	Address bit 17	1. Program read/write 2. Cleared by controller bus reset 3. This bit and address bit 16 (bit 1) are used in conjunction with the address in the memory address register (MARn) to expand the DMA addressing capability to 128K words
3	DMA I/O mode	1. Program read/write 2. Cleared by controller bus reset 3. When set, indicates DMA input operation (transfer of words from GRAPHIC 7 to host computer). When cleared, indicates DMA output operation (transfer of words from host computer to GRAPHIC 7) 4. This bit is written by the program prior to a DMA operation; it must not be changed until the DMA operation is complete
4	DMA complete	1. Program read only 2. Cleared by controller bus reset 3. Cleared when DMA operation is initiated 4. This bit is set at the completion of a DMA operation
5	Output word received	1. Program read/write (set only) 2. Cleared by controller bus reset 3. Cleared whenever output control (bit 7) is cleared 4. This bit is sent to the host computer to indicate that data has been received. It is set by the program either when a data ready interrupt occurs or when output control (bit 7) is sensed as being set 5. During a DMA output operation, this bit is set by the interface

Bit	Function	Remarks
6	Output interrupt enable	1. Program read/write 2. Cleared by controller bus reset 3. Setting this bit enables the interface to generate a data ready or a DMA complete interrupt 4. Interrupt trap address (octal) for each register is: STR1 - 000124 STR3 - 000424 STR2 - 000404 STR4 - 000444
7	Output control	1. Program read only 2. This bit, when set, interrupts the display processor to indicate that output data is available from the host computer. It reflects the status of the output control signal from the host computer
8	Word count ≠ zero	1. Program read only 2. Cleared by controller bus reset 3. Cleared when value in word count register (WCRn) equals zero 4. Set when WCRn contains non-zero value
9	Attention no. 1	1. Program read only 2. This bit reflects status of attention no. 1 signal from host computer. If attention interrupt enable (bit 11) is set, a high attention no. 1 input will cause an optional interrupt to the display processor to be generated
10	Attention no. 2	1. Program read only 2. This bit reflects status of attention no. 2 signal from host computer. If attention interrupt enable (bit 11) is set, a high attention no. 2 input will cause an optional interrupt to the display processor to be generated

Bit	Function	Remarks
11	Attention interrupt enable	1. Program read/write 2. Cleared by controller bus reset 3. This bit, when set, allows the interface to generate an optional interrupt to the display processor when either attention no. 1 (bit 9) or attention no. 2 (bit 10) goes high 4. Interrupt trap address (octal) for each register is: STR1 - 000130 STR3 - 000430 STR2 - 000410 STR4 - 000450
12	Spare input no. 2	1. Program read/write 2. Cleared by controller bus reset 3. The status of this bit is directly presented to the host computer for programming as required
13	Input word request	1. Program read/write (set only) 2. Cleared by controller bus reset 3. If a single word transfer to the host computer is desired, the program loads the input data register (IDRn) with the word and then sets this bit to indicate that the data is available. Either an input interrupt or sensing that input not ready (bit 15) is cleared indicates that the transfer is complete. 4. During a DMA input operation, the interface loads data from memory into the IDRn and then sets this bit. The bit is cleared whenever a new data ready (NDRY) pulse occurs (the interface generates an NDRY pulse for the host computer whenever the input control signal from the host computer goes high).

Bit	Function	Remarks
14	Input interrupt enable	1. Program read/write 2. Cleared by controller bus reset 3. This bit, when set, enables the interface to generate an interrupt to the display processor to indicate either that data has been accepted by the host computer or that a DMA transfer of data to the host computer is complete 4. Interrupt trap address (octal) for each register is: STR1 - 000120 STR2 - 000400 STR3 - 000420 STR4 - 000440
15	Input not ready	1. Program read only 2. When set, this bit indicates that a transfer of data to the host computer is in process. It is cleared when input word request (bit 13) is cleared and the input control signal from the host computer is low.

ODRn	OUTPUT DATA REGISTER	Octal address: 172416 (ODR1 or IDR1) 172436 (ODR2 or IDR2) 172456 (ODR3 or IDR3) 172476 (ODR4 or IDR4)
or	or	
IDRn	INPUT DATA REGISTER	



The data register is a dual-purpose register referred to as an output data register (ODRn) when data is being transferred from the host computer to the GRAPHIC 7 and as an input data register (IDRn) when data is being transferred from the GRAPHIC 7 to the host computer.

When used as an ODRn, the register is a program read only register the contents of which reflect the states of the data lines from the host computer. The program reads the contents of the register either when a data ready interrupt occurs or when output control (status register bit 7) is sensed as being set. During an output DMA operation (transferring data from the host computer to the GRAPHIC 7) the interface loads the ODRn contents into the GRAPHIC 7 memory.

When used as an IDRn, the register is a program write only register the contents of which are directly presented to the host computer. It is cleared by a controller bus reset. During an input DMA operation (transferring data from the GRAPHIC 7 to the host computer) the interface loads the IDRn with data from the GRAPHIC 7 to memory. During transfers of single words, the program loads the data into the IDRn and then sets input word request (status register bit 13).

NOTE

The parallel interface will provide either high-true or low-true data to the host computer. Similarly, the interface will accept either high-true or low-true data from the host computer.

SECTION 5
GRAPHIC CONTROL PROGRAM (GCP+)

5.1 DESCRIPTION AND PURPOSE

The standard graphic control program (GCP+) for the GRAPHIC 7 is supplied in read-only memory (ROM). This program handles all the tasks for the GRAPHIC 7 that must normally be programmed for other display systems. The programmer, therefore, need only be concerned with the generation of software for the host computer. Specific tasks performed by GCP+ with no requirement for host intervention include:

- Routine housekeeping
- Handling of all operator inputs
- Handling of trackball/forcestick or data tablet manipulations
- Handling of all graphic controller interrupts
- Automatic handling of PHOTOPEN strikes and the associated data
- Insertion of keyboard data directly into a refresh file
- Formatting of messages for GRAPHIC 7-to-host communications

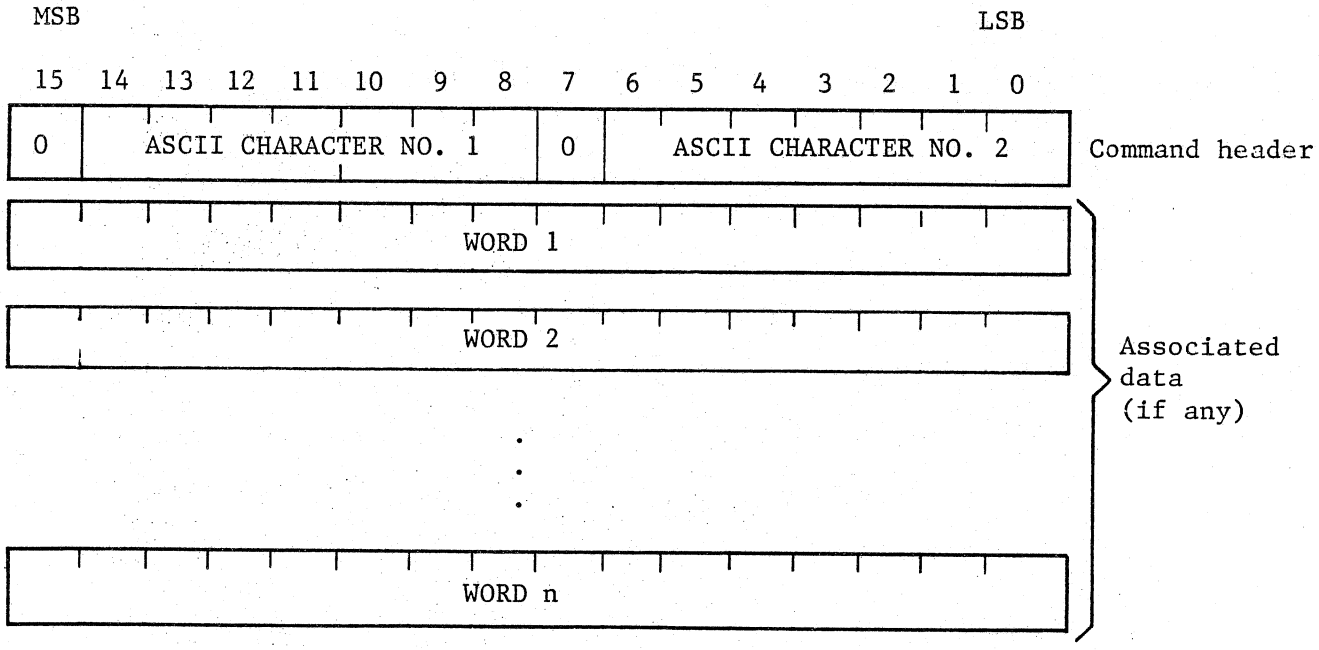
When the GRAPHIC 7 is initialized in the system mode (refer to Section 2), all peripheral devices are automatically initialized without any action by the host and GCP+ is able to accept messages from the host. As determined by the host application program, the GRAPHIC 7 is also enabled to format and transmit various types of messages to the host. The host application program determines the manner in which data in messages from the GRAPHIC 7 will be processed and the type of data that will be returned in messages to the GRAPHIC 7. For controlling these operations, the application programmer has full access to all control registers of the terminal controller.

Generation of all display instruction codes and management of the refresh file must be accomplished by the application program resident in the host computer or by software down-loaded into the GRAPHIC 7. For most computers, display instruction

macros can be used to simplify this task. Extended macro assemblers that contain the display instruction macros already exist for some computers (refer to Appendix B). Other methods of generating display instruction codes include host-resident graphic support packages and data statements. A package of this type available as an option for the GRAPHIC 7 is a remote-based FORTRAN support package (FSP).

5.2 HOST/GRAPHIC 7 COMMUNICATIONS

All communications between the host computer and the GRAPHIC 7 are handled by GCP+. Transmissions in either direction are referred to as messages. Each message begins with a command header that contains two ASCII characters to define the message type. The header is then followed by as many 16-bit words as are required to transmit the associated data. The general form of all messages is as follows:



5.2.1 SERIAL INTERFACE COMMUNICATIONS

When communications with the host computer are handled over a serial interface, the data portion of each message must be converted to an ASCII format. This translation is required for messages transmitted in either direction. For GRAPHIC 7-to-host messages, the translation is accomplished by GCP+. For host-to-GRAPHIC 7 messages, the translation must be accomplished by the host computer and GCP+ is used to restore the data to its original format. The resulting messages, regardless of content, consist entirely of the alphanumeric ASCII characters A through Z and 0 through 9

terminated with the ASCII code for a carriage return. The reason for the translation is to ensure that no ASCII code is transmitted that might interfere with a host operating system or with the serial interface itself.

ASCII characters used in message command headers are limited to G through Z. Since these headers are originally generated in ASCII format, no translation is required. Translation is required only for the information contained in the associated data words. The information in these words is translated into ASCII characters 0 through 9 and A through F. Each data word is translated in the following manner:

a. The data word is divided into four 4-bit nibbles.

b. Beginning at the left (the most significant nibble), each nibble is considered as if it represented its hexadecimal equivalent (0 through F).

c. The ASCII code for the hexadecimal number is transmitted over the serial interface (all ASCII codes are transmitted as eight-bit codes with a 0 in the most significant bit position).

After all data words have been translated and transmitted, the ASCII code for a carriage return is transmitted as an end-of-message indicator. Table 5-1 shows all possible bit combinations for nibbles and the resulting ASCII character codes into which they are translated.

As an example of the translation process, consider the host-to-GRAPHIC 7 message GI 013700_8 000746_8. This message instructs the GRAPHIC 7 to transmit 486 decimal words of data in its memory to the host computer beginning at octal address 013700. As originally constituted, the message would have the following form:

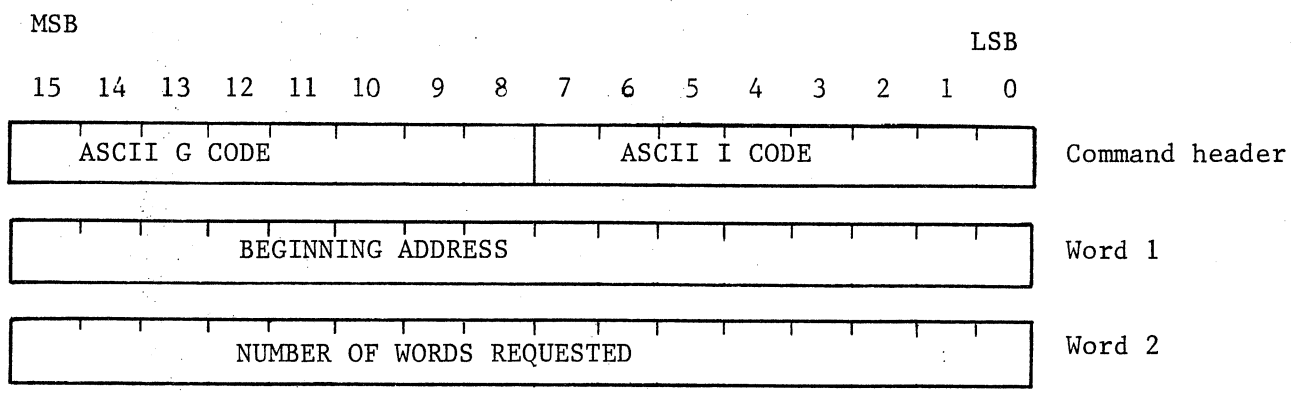
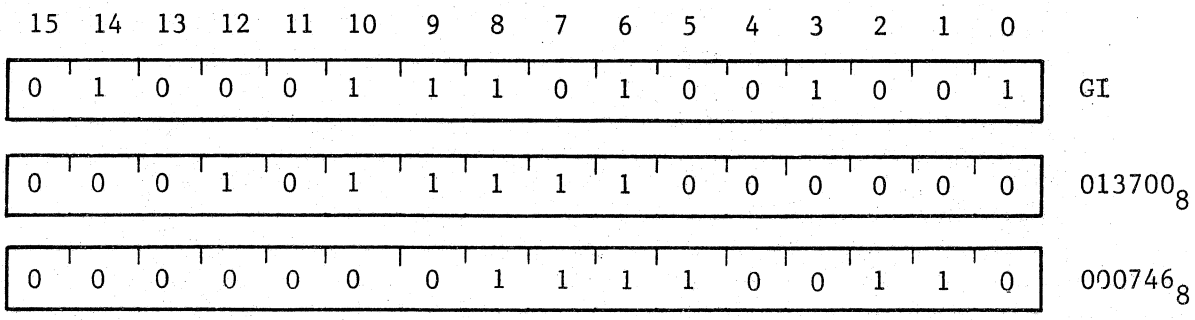


TABLE 5-1

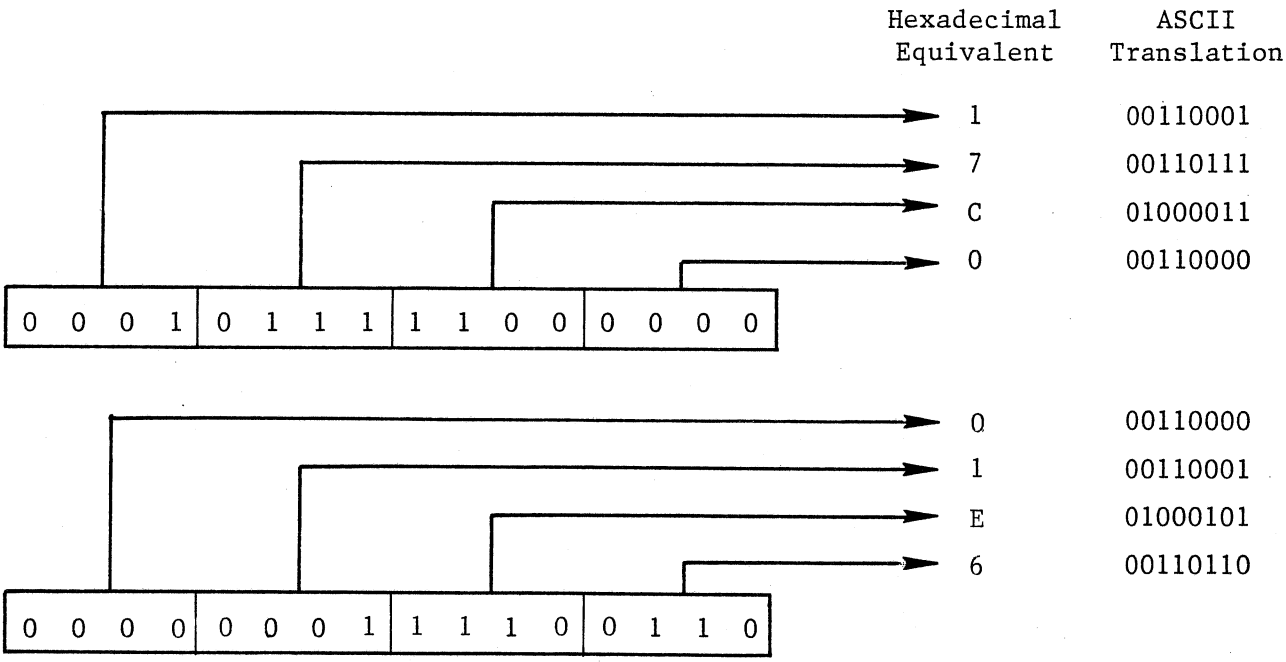
DATA WORD TRANSLATION CODES

Nibble	Hexadecimal Equivalent	ASCII Code for Hexadecimal Equivalent
0000	0	00110000
0001	1	00110001
0010	2	00110010
0011	3	00110011
0100	4	00110100
0101	5	00110101
0110	6	00110110
0111	7	00110111
1000	8	00111000
1001	9	00111001
1010	A	01000001
1011	B	01000010
1100	C	01000011
1101	D	01000100
1110	E	01000101
1111	F	01000110

Which, in binary form is:



The command header, which is already in ASCII form, is transmitted as is in two bytes with the high-order byte being transmitted first. The two data words are then divided into eight nibbles and translated into ASCII codes as follows:



After the ASCII code for a carriage return has been added at the end, the final message resulting from this example would appear as follows to be transmitted one byte at a time over a serial interface:

Command header	ASCII G CODE	0 1 0 0 0 1 1 1	
	ASCII I CODE	0 1 0 0 1 0 0 1	
Starting address data word	ASCII 1 CODE	0 0 1 1 0 0 0 1	
	ASCII 7 CODE	0 0 1 1 0 1 1 1	
	ASCII C CODE	0 1 0 0 0 0 1 1	
	ASCII 0 CODE	0 0 1 1 0 0 0 0	
		or	
Ending address data word	ASCII 0 CODE	0 0 1 1 0 0 0 0	
	ASCII 1 CODE	0 0 1 1 0 0 0 1	
	ASCII E CODE	0 1 0 0 0 1 0 1	
	ASCII 6 CODE	0 0 1 1 0 1 1 0	
Terminator	ASCII CR CODE	0 0 0 0 1 1 0 1	

5.2.2 PARALLEL INTERFACE COMMUNICATIONS

When communications between the host computer and the GRAPHIC 7 are handled over a parallel interface, messages in both directions are transmitted in the binary 16-bit word format in which they are originally constituted. No translation of the data words is necessary and no end-of-message indicator is required. Note, however, that ASCII codes are used for the two characters in the command header regardless of whether the message is handled over a parallel or a serial interface.

5.3 HOST/GRAPHIC 7 MESSAGES

There are eleven different groups of messages transmitted between the host computer and the GRAPHIC 7. These groups are listed below:

1. Initialize and error messages
2. Establish I/O transmission mode (polling/non-polling)
3. Memory related messages
4. Interrupt related messages
5. Keyboard related messages
6. Positional entry device related messages
7. PHOTOPEN related messages
8. Hardcopy messages
9. Fortran support (FSP) messages
10. Option messages
11. 3D coordinate converter messages

5.3.1 INITIALIZE AND ERROR MESSAGES

The initialize and error group consist of the following messages:

HOST-to-GRAPHIC 7 (H→G7)

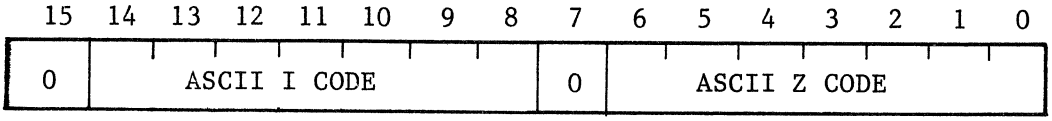
IZ Initialize

GRAPHIC 7-to-HOST (G7→H)

XX

The following paragraphs discuss these messages and give details concerning the format and application of each.

IZ (H→G7) INITIALIZE Command header code (octal): 044532



Command header

The initialize message is a single-word message that causes the GRAPHIC 7 to initialize in the system operating mode (refer to Section 2). Initialization in the system mode results in the following:

a. Associated display indicator(s) goes blank.

b. Associated keyboard(s) and PED's are enabled.

c. Built-in diagnostic tests are performed.

d. The results of the diagnostic tests are sent in an XX message to the host computer.

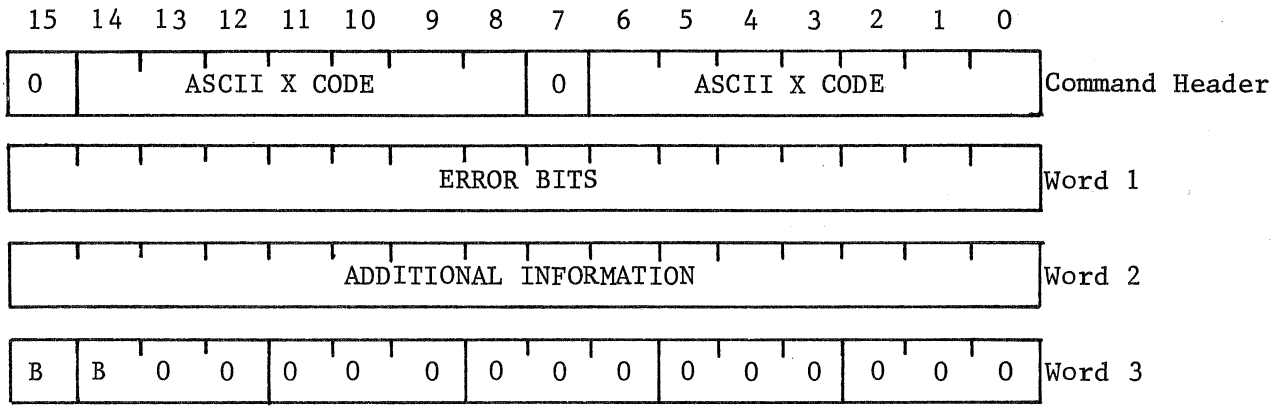
NOTES

1. An IZ message is recognized by the GRAPHIC 7 only when the GRAPHIC 7 is operating in the system mode. If the GRAPHIC 7 is operating in the local mode, the host computer must first generate a hard-wired INIT signal (if a parallel interface is used) or a RING+ signal (if a serial interface is used) or the operator must press the SYSTEM switch on the GRAPHIC 7 front panel.

2. After an IZ message has been sent, no further messages should be sent from the host computer to the GRAPHIC 7 until an XX message has been sent from the GRAPHIC 7 to the host computer.

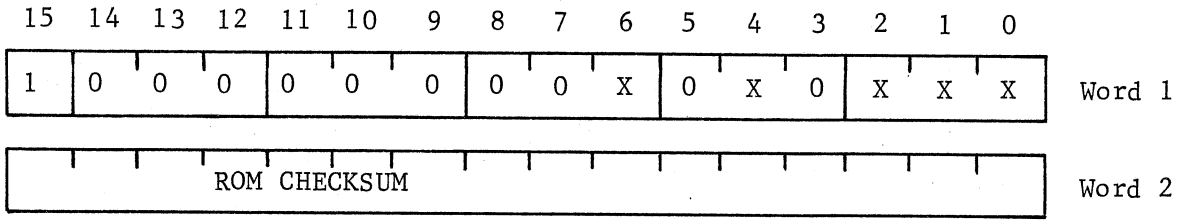
3. When the GRAPHIC 7 is operated in the teletypewriter emulation mode (refer to Section 2), initialization in the system mode can be accomplished by sending the code for ASCII character group separator (octal code 035) from the host computer to the GRAPHIC 7.

XX (G7→H) ERROR STATUS Command Header code (octal): 054130



Whenever the GRAPHIC 7 is initialized in the system mode (refer to paragraph 2.3), an XX message is automatically sent to the host computer to indicate the results of the diagnostic tests performed and the ROM checksum calculated during the initialization routine. When the GRAPHIC 7 is operating in the system mode, XX messages are also automatically sent to the host computer (provided that error detection has been enabled via the IM message) whenever an error condition is sensed by GCP+. There are four basic categories of XX messages, each of which has a slightly different format for words 1 and 2. The make-up of words 1 and 2 for each category is as follows (a 1 in a bit position marked "X" indicates an error condition or failure of a diagnostic test):

a. Initialization XX message:



Bits in word 1 indicate the following:

Bit 0 - results of interface diagnostic test.

Bit 1 - results of graphic controller diagnostic test.

Bit 2 - results of display processor diagnostic test.

Bit 4 - results of 3-D Converter diagnostic test.

Bit 6 - results of read/write memory diagnostic test.

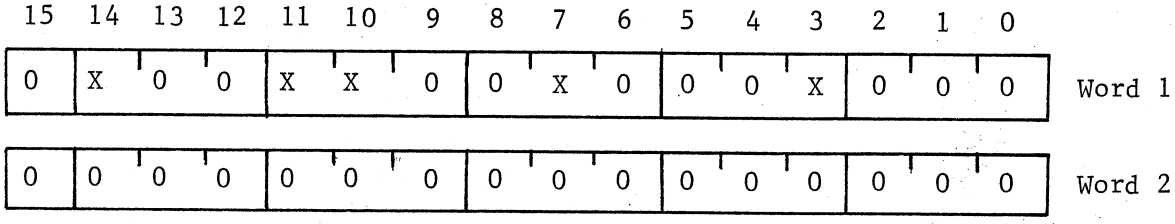
Bit 15 - set to 1 indicates initialization XX message.

All other bits are always zero.

Any possible combination of test results can be indicated.

Word 2 contains the result of the ROM checksum calculation.

b. Normal running XX message:



Bits in word 1 indicate the following (when set to 1):

Bit 3 - incorrect command header format sent by host computer.

Bit 7 - unidentified internal interrupt detected by display processor.

Bit 10 - GCP+ serial interface buffer is full.

Bit 11 - GCP+ serial interface buffer is 7/8 full.

Bit 14 - Command header not recognized by GCP+.

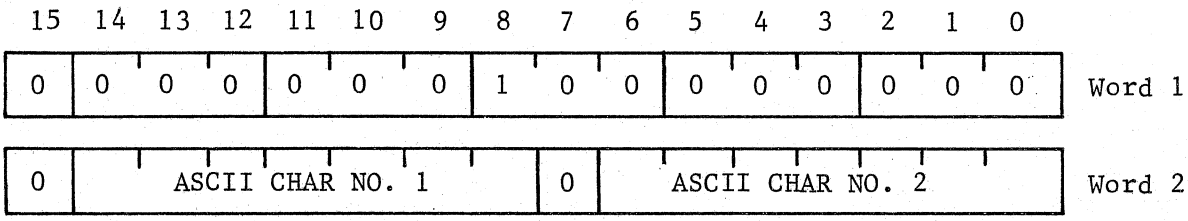
All other bits are always zero.

Any combination of errors can be indicated.

Word 2 contains all zeros for bit 3, 10, 11 and 14 type errors.

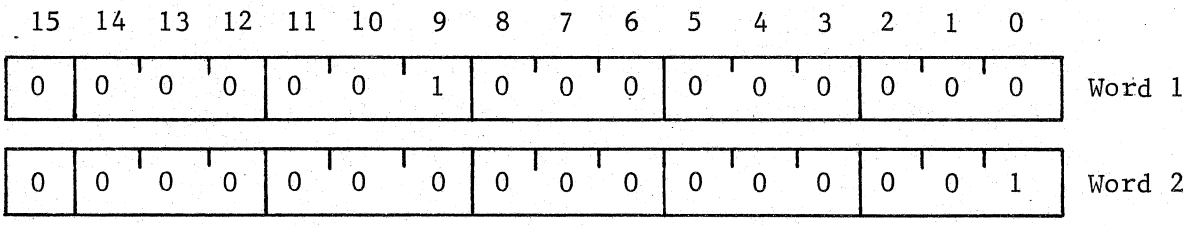
For bit 7 errors, word 2 contains the address plus 4 bytes to identify the address of the unidentified internal interrupt.

c. Buffer XX message:



A buffer XX message is sent when no output buffer is available to GCP+ for a message to be sent to the host computer. Bit 8 in word 1 identifies the message as a buffer XX message; all other bits in word 1 are always zero. Word 2 is the command header for the message that could not be sent to the host computer.

d. Character overrun XX message:



A character overrun XX message is sent whenever a character overrun condition or parity error is detected at the serial interface port used for communications with the host computer (normally port 1). Bit 9 in word 1 identifies the XX message as a character overrun XX message; all other bits in word 1 are always zero. Word 2 identifies the port on which the overrun was detected (GCP+ assumes serial communications with the host computer are handled via port 1. Therefore, word 2 of a character overrun XX message always has a binary value equal to 1).

For all XX messages:

Bits 14 and 15 of word 3 contain the bank number associated with the XX Message. Bits 14 and 15 are defined as follows:

BITS		BANK NUMBER
15	14	
0	0	0
0	1	1
1	0	2
1	1	3

5.3.2 ESTABLISH I/O TRANSMISSION MODE (POLLING/NON-POLLING)

The establish I/O transmission group consists of the following messages:

HOST-to-GRAPHIC 7

IM	Initialize I/O message formats
PL	Poll GRAPHIC 7 for next message
NO	No operation

GRAPHIC 7-to-HOST

NM	No messages ready

The following paragraphs discuss these messages and give details concerning the format and application of each.

IM (H→G7) INITIALIZE I/O MESSAGE FORMATS Command Header Code (octal): 044515

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	ASCII I CODE							0	ASCII M CODE							Command Header
0	D	D	D	D	D	D	D	0	0	0	0	D	D	D	D	Word 1

The IM message is used to activate or de-activate error detection and to initialize GCP+ to operate in either a polling or non-polling mode.

A detailed description of the meaning of all bits in WORD 1 is given below:

Bit		Value	Description
0	Polling	0	GCP+ operates in non-polling mode (i.e., messages are automatically sent to HOST when a message is ready.)
		1	GCP+ operates in a polling mode (i.e., the HOST must issue a poll (PL) message each time the Host wants the next message from the GRAPHIC 7.)
1	Send poll message back	0	In this mode, GCP+ does not respond to polls until a message is ready for transmission to the Host.
		1	In this mode, if the GCP+ output buffer is empty, a dummy NM message is sent back to the Host to indicate that the GCP+ output buffer is empty.
*2	Host to GRAPHIC 7 data not packed	0	GCP+ interprets all data words sent from the HOST to be in packed format.
		1	GCP+ interprets all data words sent from the HOST to be in image format.
3	Activate error detection	0	GCP+ ignores all command header errors. This permits the operation of GCP+ in serial full-duplex mode. Messages echoed back to GRAPHIC 7 from Host are ignored.
		1	GCP+ validates all command headers; when errors are detected, an appropriate XX error message is sent to the Host.
4-6	Reserved		These bits are reserved for future expansion.

*Planned as a future enhancement to GCP+.

Bit		Description
8-14	Special poll character	These bits operate in conjunction with bits 0 and 1. If bits 8-14 are all zeroes, GCP+ sends messages back to the Host anytime a PL message is received. If bits 8-14 contain any non-zero value, then GCP+ does not send a message to the Host until the Host sends a PL message followed by the special poll character. Effectively, when GCP+ receives the PL message, it prepares a message for transmission to the Host. It then waits for the special poll character before the message is sent to the Host. The special poll character is sent by the Host to indicate that it is ready to receive the next GRAPHIC 7 message. The special poll character permits operation with operating systems that use a special character to turn around (change direction) a serial communications line from output to input.

NOTE

By default, GCP+ is initialized so that all bits (functions) represented by word 1 of the IM message are set to Ø.

When GCP+ is initialized by the IZ message, it is set up to operate in a non-polling mode. In this mode, whenever GCP+ has a message stored in its output buffer, the message is automatically sent to the host computer (i.e., GCP+ is operating in an asynchronous or non-polling environment).

By setting bit Ø of word 1 of the IM message to a 1, GCP+ can be set up to operate in a polling mode. In this mode GCP+ only sends a message to the host computer when the following two conditions exist.

1. GCP+ has a message stored in its output buffer.

2. The host has issued a PL message. The PL message tells GCP+ that the host computer is ready to receive the next message and that GCP+ should send it. If a message is stored in the output buffer, GCP+ immediately sends it to the host computer. If no message is stored in the output buffer, GCP+ waits until a message gets stored in the output buffer, then it sends the message to the host computer.

Bit 1 of word 1 works in conjunction with the way bit Ø has been set up. When bit Ø is set up for non-polling mode (bit Ø = 0), the value of bit 1 is ignored. When bit Ø is set up for polling mode (bit Ø = 1), GCP+ operates as follows:

1. When bit 1 is Ø, GCP+ operates in a polling mode as described in the previous paragraph.

2. When bit 1 is 1, GCP+ operates in a polling mode that is slightly different. In this mode, after GCP+ receives the PL message, it does one of the following:

   (1) If a message is stored in the output buffer, GCP+ immediately sends it to the host computer.

   (2) If the output buffer is empty, GCP+ immediately sends a NM dummy message to the host computer. The NM message indicates to the host computer that the GCP+ output buffer is empty and that the communications line between the GRAPHIC 7 and the host computer is still active.

The special poll character is applicable to serial communications only. This character is defined in bits 8 through 14 of word 1. The special character poll mode works in conjunction with the way bits 0 and 1 have been set up. When bit Ø is Ø, the special poll character is ignored by GCP+. Setting bit Ø to a 1 activates the special poll character bits. If the special poll character is set up for Ø (i.e., bit 8 through 14 are Ø), the polling modes previously described are in effect. If the special poll character is set up with non-zero value, then the following special polling mode is activated.

(1) Host computer sends a PL message to GCP+.

(2) Host computer sends the special poll character to GCP+.

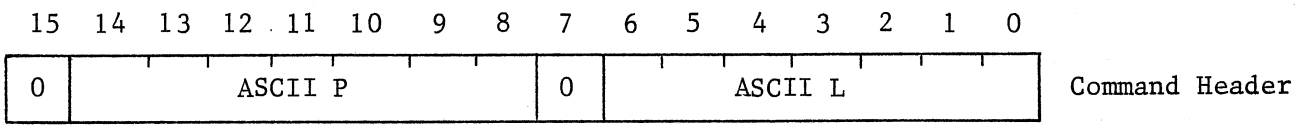
(3) After GCP+ receives the PL message followed by the special poll character, the next message is sent to the host computer. The sending of this message is based on whether the output buffer has a message and the way bit 1 has been set up.

Bit 3 of word 1 of the IM message is used to activate the detection of command header errors. By default, GCP+ is initialized to ignore all command header errors. By setting bit 3 to a 1, GCP+ can be activated to validate all command headers. When GCP+ detects that the host computer has sent an invalid GCP+ message, it stores an appropriate XX error message in its output buffer.

The recommended values for word 1 of the IM message are given below:

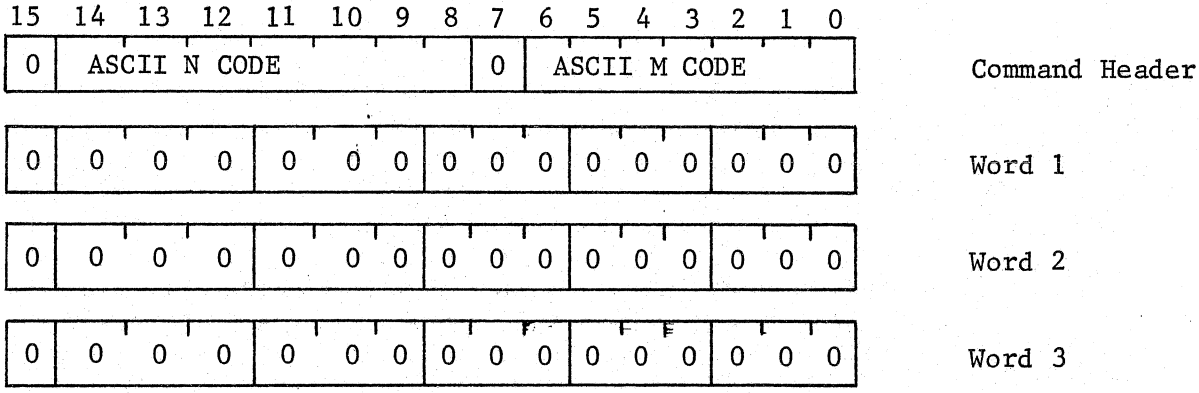
Type of Interface	Polling Mode	Non-Polling Mode
Parallel	9 or 11	8
Serial half-duplex	9 or 11	8
Serial full-duplex (echoing enabled)	1 or 3	0
Serial full-duplex (echoing disabled)	9 or 11	8

PL (H→G7) POLL GRAPHIC 7 FOR NEXT MESSAGE Command Header Code (octal): 050114



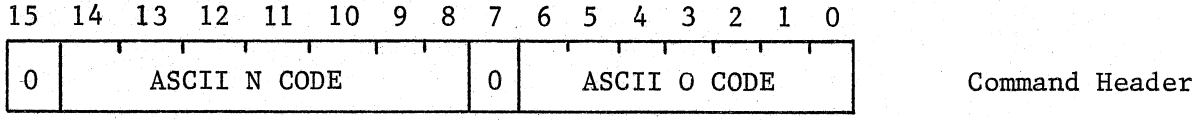
The poll message is sent by the Host to request that the GRAPHIC 7 send the next message. This command works in conjunction with the way the IM command has initialized GCP+.

NM (G7→H) NO MESSAGES READY Command header code (octal): 047115



This message is sent by GCP+ (in response to the PL command) when the output buffer is empty and the poll message bit has been set previously by the IM command.

NO (H→G7) NO OPERATION Command header code (octal): 047117



The NO message is a single-word message that causes no operation to be performed by GCP+. NO messages are used primarily as fillers when the host computer application program requires that all messages sent to the GRAPHIC 7 be of constant length.

5.3.3 MEMORY RELATED MESSAGES

The memory related messages consist of the following:

Host-to-GRAPHIC 7

MS	Memory bank select
MU	Memory update
SU	Selective update
RU	Register update
SP	Start picture
HP	Halt picture
KP	Continue picture
TK	Transfer control

GI Give image

GR Give register

GRAPHIC 7-to-Host

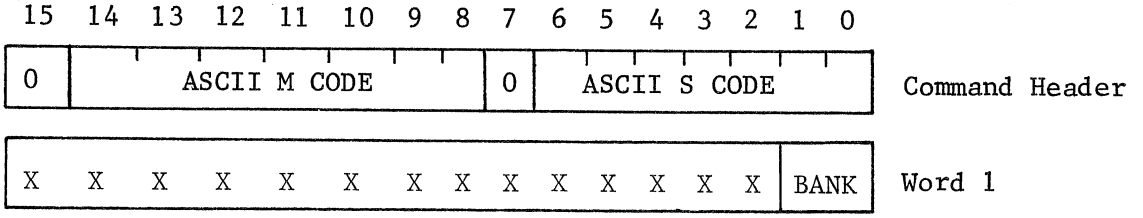
RI Return image

VL Variable length

RR Return register

The following paragraphs discuss these messages and give details concerning the format and application of each.

MS (H→G7) MEMORY BANK SELECT Command Header Code (octal): 046523



The MS message is used to select the desired memory bank. This message should be issued prior to such commands as MU, SU, GU and GI if a large memory system is in use. Bits 2 through 15 in word 1 are ignored by GCP+. Bits 0 and 1 represent the bank number selected as given below:

Bits 0	Bits 1	BANK NUMBER SELECTED
0	0	0
0	1	1
1	0	2
1	1	3

Below is a table showing the relation of virtual addresses (i.e., addresses specified in MU, SU, GU, and GI messages) to physical addresses when different memory banks are selected.

BANK NUMBER	VIRTUAL ADDRESS	PHYSICAL ADDRESS	RAM PAGES
0	000000-177777	000000-137777	00-05
1	000000-177777	200000-377777	10-17
2	000000-177777	400000-577777	20-27
3	000000-177777	600000-777777	30-37

NOTE

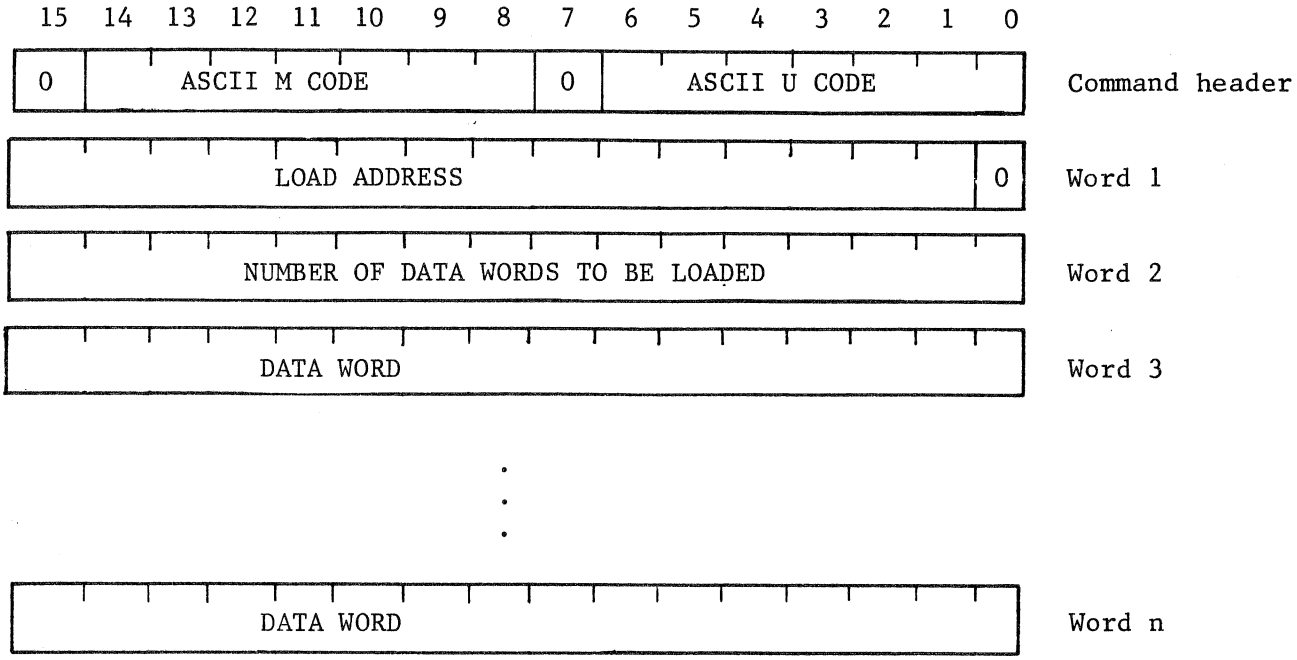
When GCP+ is initialized by default, bank Ø is selected. Memory bank selection should be taken into account for the following Host-to-GRAPHIC 7 messages:

MU, SU, SP, GI, MS, IP, IT, TK, ZR, ZT, MI, GU

Memory bank selection should be taken into account for the following GRAPHIC 7-to-Host messages:

RI, XX, PN, PT, HI, XI

MU (H→G7) MEMORY UPDATE Command header code (octal): 046525

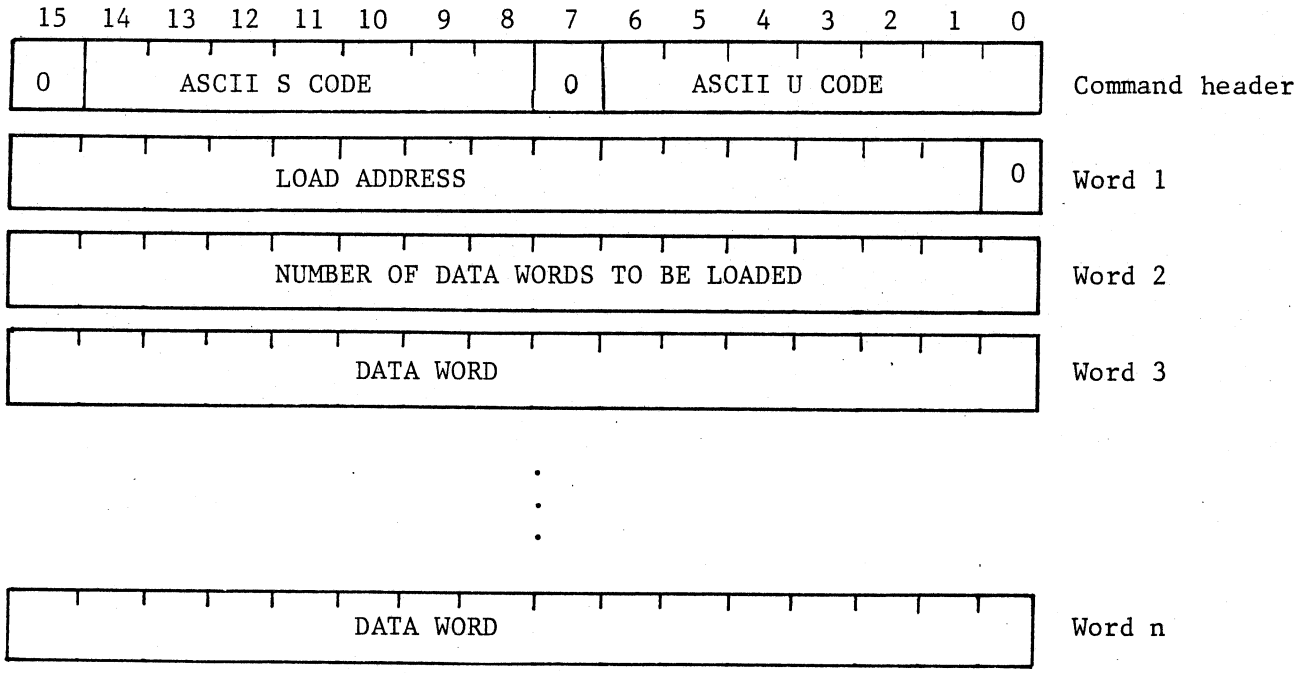


The MU message is a variable-length message used to load data into the read/write memory of the GRAPHIC 7. The load address specified in word 1 tells GCP+ the address at which loading of the data should begin. This address must be the address of an even-numbered byte (an odd address will result in an XX (error status) message being returned by the GRAPHIC 7 and data will not be loaded).

Word 2 specifies the total number of data words that are to be loaded into successive read/write memory locations. This word is then followed by the data words to be loaded.

When a memory update message is sent from the host computer to the GRAPHIC 7, GCP+ halts the graphic controller (this blanks the display indicator). The GRAPHIC 7 then remains halted until a KP (continue picture) or SP (start picture) is sent by the host computer. For lengthy memory update messages, this can result in noticeable blanking on the display indicator.

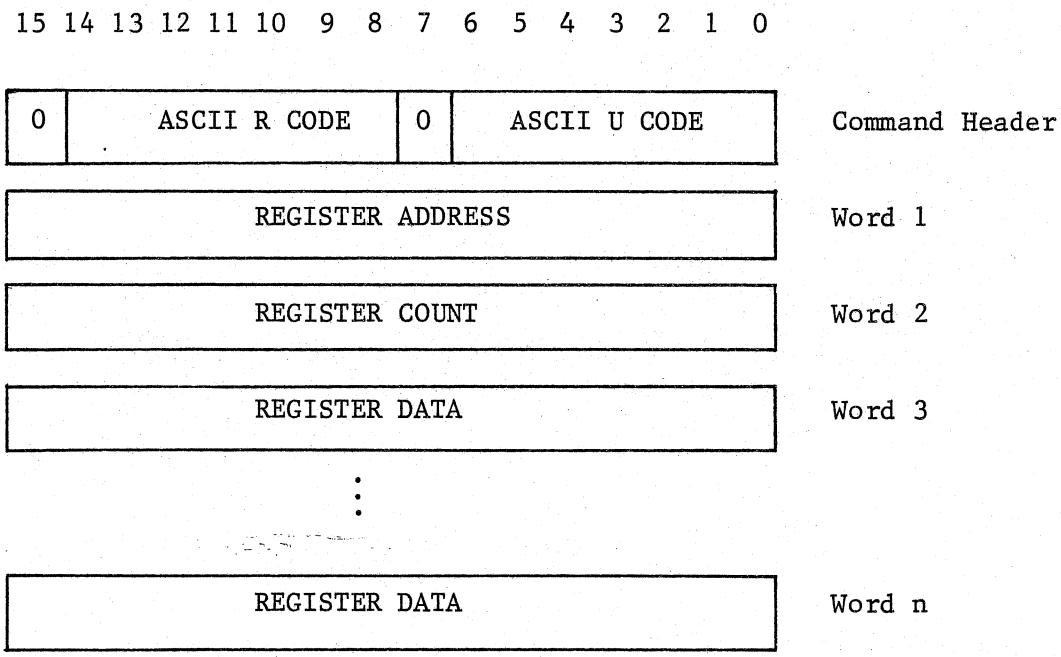
SU (H→G7) SELECTIVE UPDATE Command header code (octal): 051525



The SU message is a variable-length message that operates in exactly the same manner as the MU (memory update) message except that the graphic controller is not halted. Therefore, if an SU message is used to update a refresh file currently being processed by the graphic controller, the file must remain valid as each data word is replaced.

More commonly, an SU message is used to load a new refresh file into a different area of memory while an older file is being processed by the graphic controller. After loading of the new file is complete, an SP (start picture) message from the host computer causes the graphic controller to process the new file. This assures that the display will not be blanked while the data is being transferred as occurs when an MU (memory update) message is used.

RU (H→G7) REGISTER UPDATE Command Header Code (octal): 051125

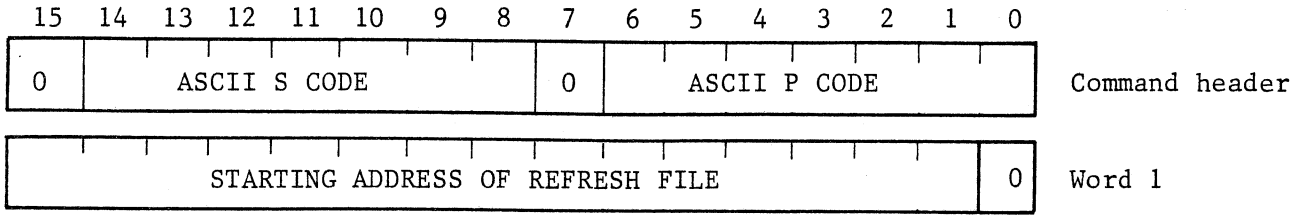


The RU message is a variable length message that is used to update a series of registers in the I/O address area of the hardware. Word 1 contains the address of the first register to be updated. Valid register addresses are in the range of 160000-177777 (octal). Word 2 contains the register count indicating the number of successive registers to be updated. Words 3 through n contain the data values to be loaded into each register.

NOTE

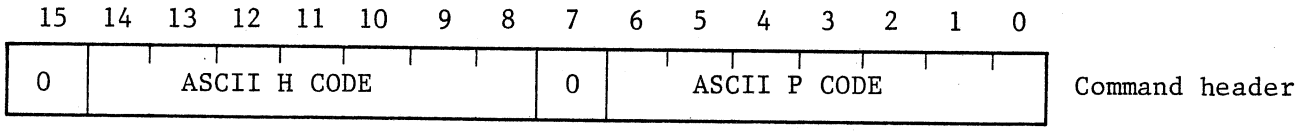
The RU message does not change the current memory bank selection. It is also possible to interpret register address as memory address in the above message. When updating memory address, the user must take into account memory mapping. Memory addresses in the range of 020000 to 077777 are subject to memory mapping.

SP (H→G7) START PICTURE Command header code (octal): 051520



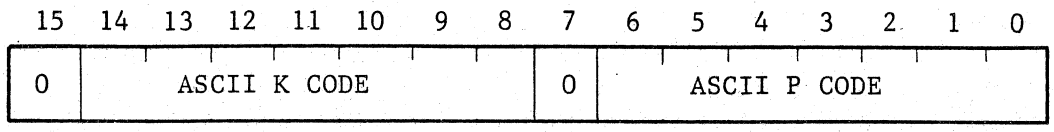
The SP message is a two-word message that causes the graphic controller to begin processing a refresh file starting at the address specified in word 1. The specified starting address should be the address of an even-numbered byte. If, however, an odd address is specified, the low order bit is ignored by GCP+ and no XX (error status) message is generated.

HP (H→G7) HALT PICTURE Command header code (octal): 044120



The HP message is a single-word message that causes GCP+ to halt the graphic controller (this blanks the display indicator). The refresh file is not altered and the graphic controller program counter remains pointing at the location of the next instruction to be processed. Following an HP message, the graphic controller remains halted (and the display blanked) until an SP (start picture) or KP (continue picture) message is sent by the host computer.

KP (H→G7) CONTINUE PICTURE Command header code (octal): 045520



The KP message is a single-word message used to restart the graphic controller at the instruction following the one at which it halted. Conditions causing the graphic controller to halt include:

a. Host computer sends HP (halt picture) message to GRAPHIC 7

b. Host computer sends MU (memory update) message to GRAPHIC 7

c. Display processor sends stop function code (165040) to graphic controller

d. Display processor executes RESET instruction

e. HREF instruction executed by graphic controller

f. LINK instruction executed by graphic controller

g. Invalid instruction executed by graphic controller

h. Bus timeout (memory fails to respond to a fetch command)

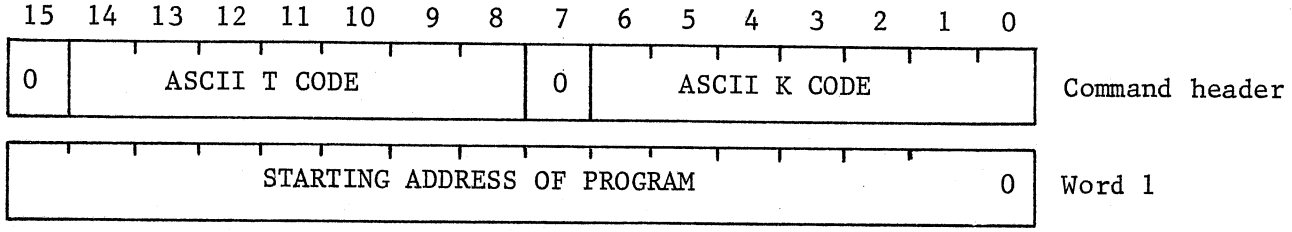
i. PHOTOPEN strike (if interrupt to display processor is enabled)

j. X or Y position overflow (if interrupt to display processor is enabled)

NOTE

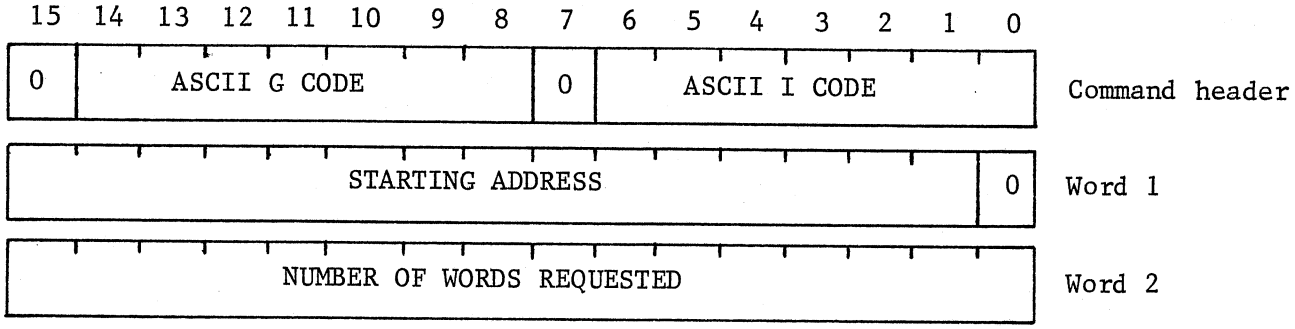
If the graphic controller is running when a KP message is sent by the host computer, GCP+ returns an XX (error status) message to the host computer.

TK (H→G7) TRANSFER CONTROL Command header code (octal): 052113



The TK message is a two-word message that causes the display processor to stop processing GCP+ and begin processing the program that begins at the address specified in word 1. This message is intended for advanced applications to permit a program other than GCP+ to be processed by the display processor. Normally, such a program would be down-loaded from the host computer using an MU (memory update) message and then started by using a TK message. After control has been transferred, no further communications via GCP+ are possible unless the new program deliberately re-enters GCP+.

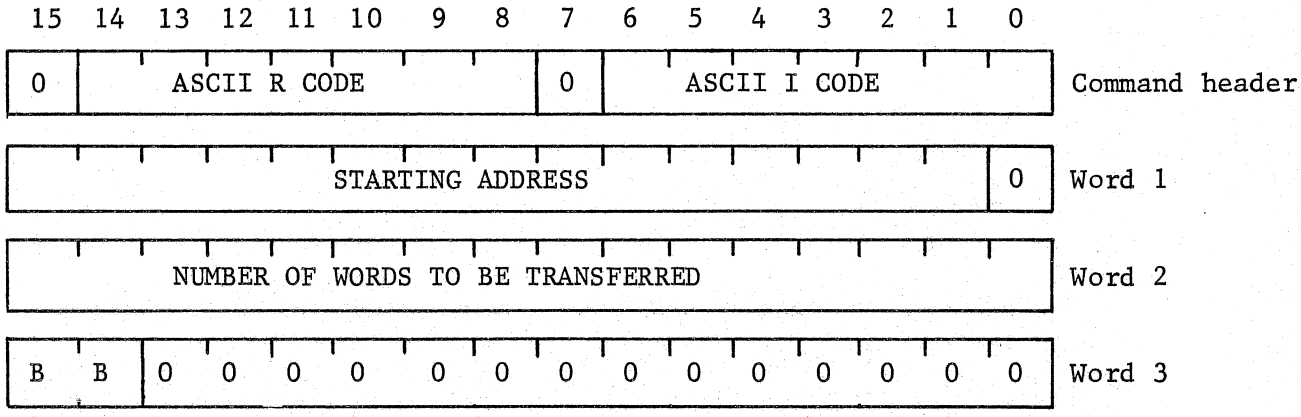
GI (H→G7) GIVE IMAGE Command header code (octal): 043511



The GI message is a three-word message that causes GCP+ to send back to the host computer the contents of the GRAPHIC 7 memory beginning at the specified starting address and ending when the requested number of words have been sent. The specified starting address should be the address of an even-numbered byte. If, however, an odd address is specified, the low order bit is ignored by GCP+ and no XX (error status) message is generated.

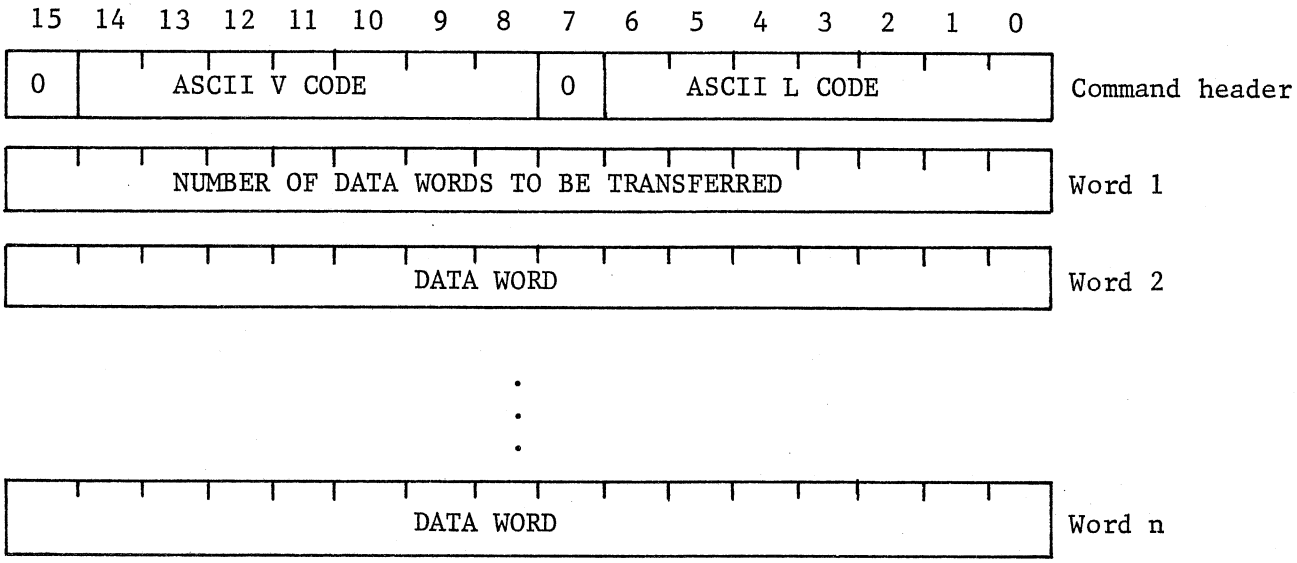
In response to a GI message, GCP+ sends an RI (return image) and a VL (variable length) message to the host computer. The RI message indicates the length of the VL message while the VL message contains the requested data.

RI (G7→H) RETURN MESSAGE Command header code (octal): 051111



GCP+ returns an RI message to the host computer in response to a GI (give image) message from the host computer. Word 1 specifies the starting address of the data to be transferred and word 2 specifies the number of 16-bit words to be transferred. The data in words 1 and 2 are always the same as the data in the corresponding words of the requesting GI message. Bits 14 and 15 contain the bank number that the RI message is related to. Each RI message is immediately followed by a VL (variable length) message that contains the data requested by the host computer.

VL (G7→H) VARIABLE LENGTH Command header code (octal): 053114

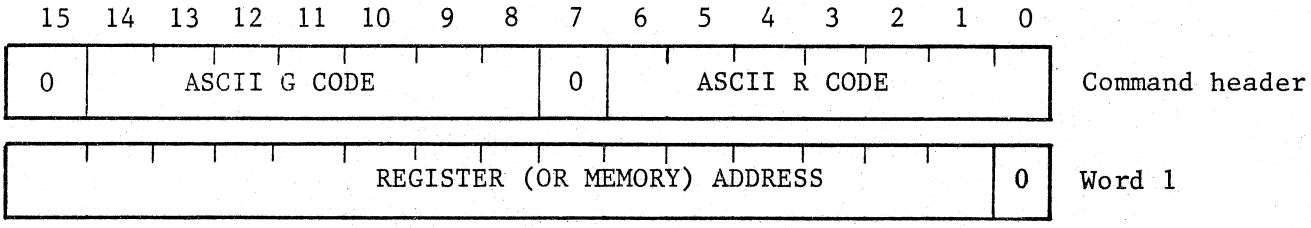


The VL message is the only GRAPHIC 7-to-host message that does not necessarily contain four words. Its length is determined by the number of data words to be transferred. In response to a GI (give image) message from the host computer, GCP+ returns an RI (return image) message that is immediately followed by a VL message. The RI message informs the host computer that a VL message is to follow while the VL message contains the data requested by the host computer. Word 1 of each VL message always contains the same data as word 2 of the preceding RI message. Words 2 through n of each VL message contain the requested data.

NOTE

Normally, after receiving the RI message, the host sets up to read in the VL message. For host dma operations, the word count specified to read in the VL message should be set equal to the number of words to be transferred (i.e., word 2 of the RI message), plus two.

GR (H→G7) GIVE REGISTER Command header code (octal): 043522



The GR message is a two-word message used by the host computer to obtain the contents of the GRAPHIC 7 register specified by the register address in word 1. The contents of any register having an assigned address may be obtained in this manner. If required, GCP+ automatically halts the graphic controller before the data is obtained and then restarts it at the completion of the operation. In response to a GR message, GCP+ sends an RR (return register) message to the host computer.

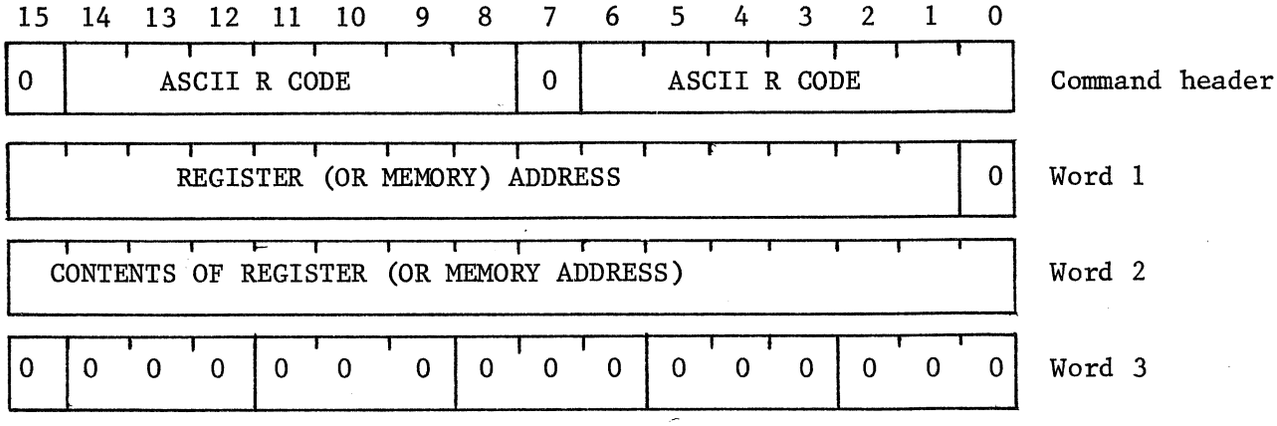
Although the intent of the GR message is to permit the contents of registers to be read, it can also be used to read the contents of GRAPHIC 7 memory addresses. When it is used to read a memory address, the specified address in word 1 must be that of an even-numbered byte. If the address of an odd-numbered byte is specified, GCP+ causes an XX (error status) message to be sent to the host computer.

NOTE

When the GR message is used on a large memory system, the following restrictions must be taken into account.

1. Addresses in the range of 000000-017776 are directly addressable.
2. Addresses in the range of 020000-077776 are subject to memory mapping.
3. Addresses in the range of 100000-117776 are directly addressable.
4. Addresses in the range of 120000-177776 are related to ROM and I/O device registers.

RR (G7→H) RETURN REGISTER Command header code (octal): 051122



An RR message is sent by GCP+ to the host computer in response to a GR (give register) message from the host computer. Word 1 of an RR message is always the same as word 1 of the requesting GR message. The requested data is returned to the host computer in word 2. Word 3 always contains all zeros.

5.3.4 INTERRUPT RELATED MESSAGES

The interrupt related group consist of the following messages:

Host-to-GRAPHIC 7

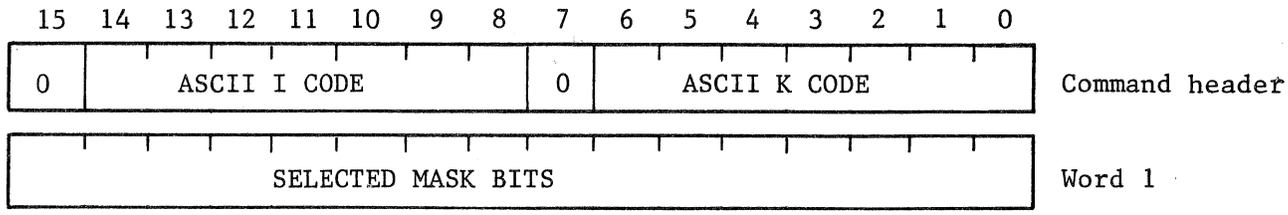
- IK Interrupt control
- IS Enable selected interrupts
- ZI Disable selected interrupts

GRAPHIC 7-to-Host

- HI Halt interrupt
- XI X or Y position overflow interrupt

The following paragraphs discuss these messages and give details concerning the format and application of each.

IK (H→G7) INTERRUPT CONTROL Command header code (octal): 044513



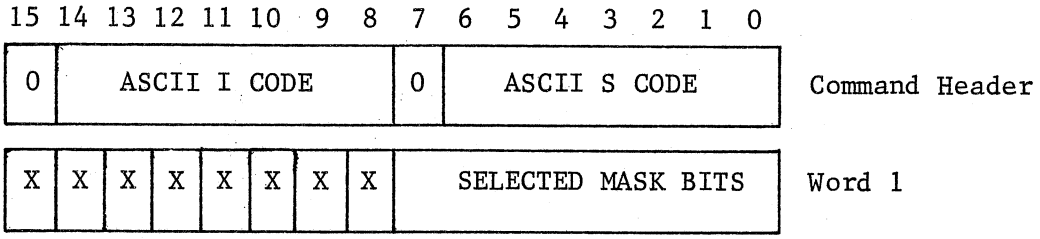
The IK message is a two-word message used to enable or disable certain GCP+ functions and to determine conditions under which the graphic controller can interrupt the display processor. When an IK message is sent, the contents of the low-order byte (bits 0-7) of the word 1 directly replaces the contents of the graphic controller mask register (MKR). The high-order byte (bits 8-15) of the word is decoded and used to enable associated interface ports or set internal software flags as required. The function or interrupt condition associated with each bit is as follows:

Bit	Function or Interrupt Condition
0	Not used
1	Graphic controller halted
2	X or Y position overflow
3	Real time clock
4	PHOTOPEN 1 strike
5	Switch of PHOTOPEN 1 actuated
6	PHOTOPEN 2 strike
7	Switch of PHOTOPEN 2 actuated
8	PHOTOPEN item number enabled
9	Not used
10	PED no. 2 (serial interface port 8)
11	Alphanumeric/function keyboard no. 2 (serial interface port 7)
12	Spare (serial interface port 6)
13	PED no. 1 (serial interface port 4)
14	Alphanumeric/function keyboard no. 1 (serial interface port 3)
15	Spare (serial interface port 2)

NOTE

Bits 12 and 15 are designated as spare ports. It is possible to attach alphanumeric keyboards to these ports but it is not recommended. If a combination alphanumeric/function keyboard is connected to the spare ports, all function key information is ignored.

IS (H→G7) ENABLE SELECTED INTERRUPTS Command Header Code (octal): 044523

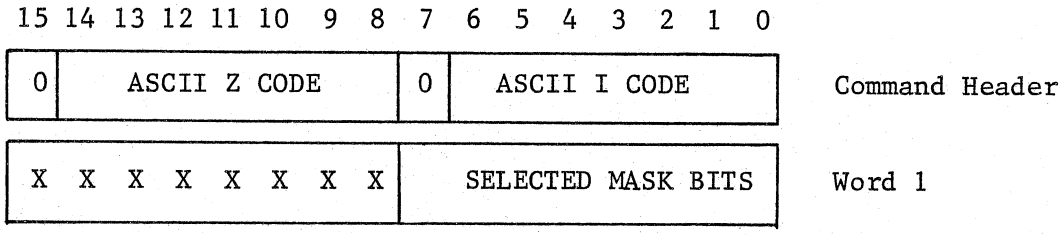


The IS message is a two word message used to selectively enable mask register (MKR) associated interrupts. If a mask bit is set to 1, then the interrupt associated with that bit is enabled. If a mask bit is set to Ø, then the interrupt associated with that bit remains unchanged after the IS message is processed by GCP+.

The function or interrupt associated with each bit is as follows:

BIT	FUNCTION OR INTERRUPT CONDITION
0	Not used
1	Graphic controller halted
2	X or Y position overflow
3	Real time clock
4	PHOTOPEN 1 strike
5	Switch of PHOTOPEN 1 actuated
6	PHOTOPEN 2 strike
7	Switch of PHOTOPEN 2 actuated
8-15	These bits are ignored by GCP+

ZI (H→G7) DISABLE SELECTED INTERRUPTS Command Header Code (octal): 055111



The ZI message is a two word message used to selectively disable mask register (MKR) associated interrupts. If a mask bit is set to 1, then the interrupt associated with that bit is disabled. If a mask bit is set to Ø, then the interrupt associated with that bit remains unchanged after the ZI message is processed by GCP+.

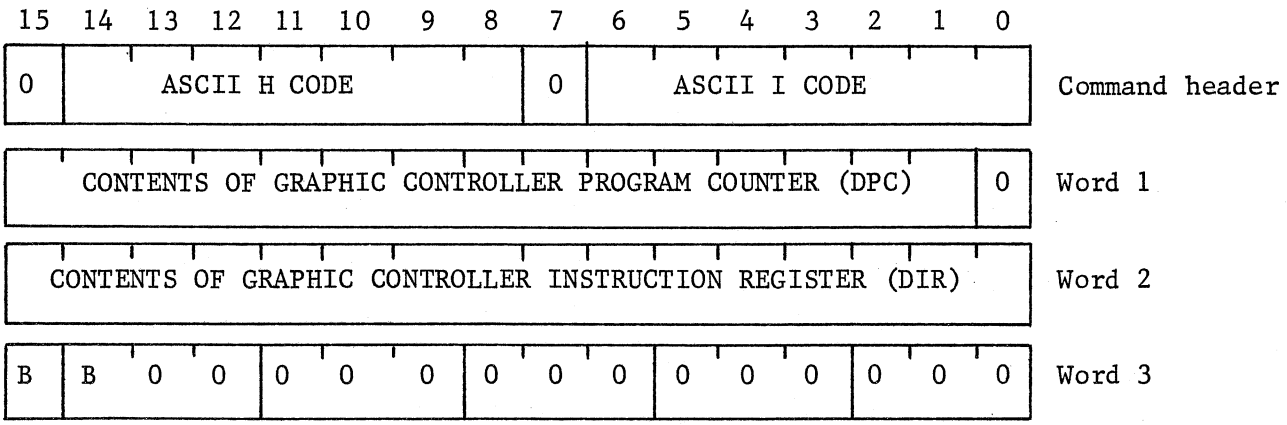
The function or interrupt associated with each bit is as follows:

BIT	FUNCTION OR INTERRUPT CONDITION
0	Not used
1	Graphic controller halted
2	X or Y position overflow
3	Real time clock
4	PHOTOPEN 1 strike
5	Switch of PHOTOPEN 1 actuated
6	PHOTOPEN 2 strike
7	Switch of PHOTOPEN 2 actuated
8-15	These bits are ignored by GCP+

NOTE

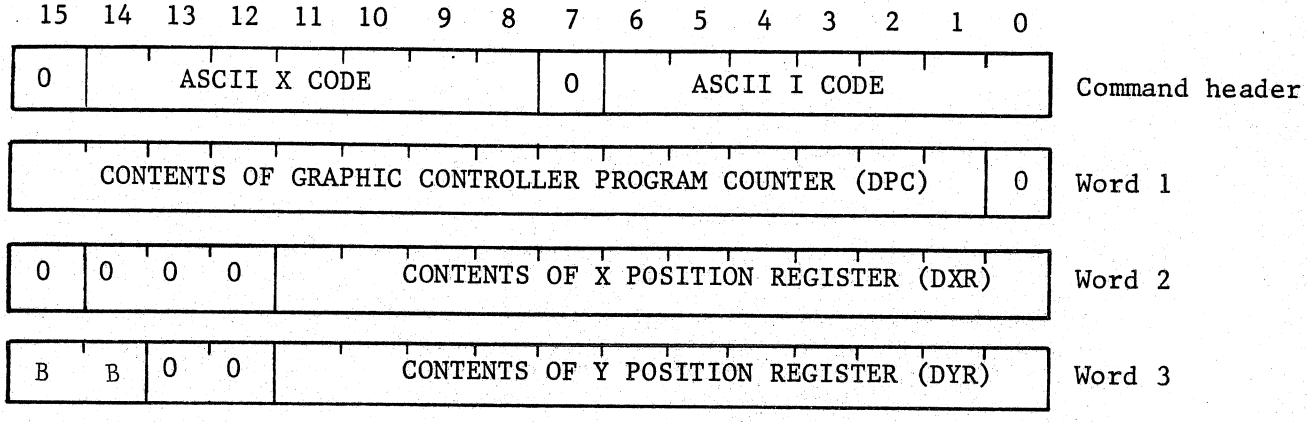
If a data tablet is operating in the auto/tracking mode, the real time clock shouldn't be disabled.

HI (G7→H) HALT INTERRUPT Command header code (octal): 044111



When the graphic controller halt interrupt to the display processor is enabled (by a host-to-GRAPHIC 7 IK or IS message), GCP+ sends an HI message to the host computer each time that a HREF instruction is executed by the graphic controller. Word 1 of the HI message contains the contents of the graphic controller program counter (DPC) which is the address of the instruction following the HREF instruction. Word 2 contains the contents of the graphic controller instruction register (DIR) which, in turn, is the contents of the address pointed to by the program counter. Bits 14 and 15 of word 3 contain the bank number associated with the HI interrupt.

XI (G→H) X OR Y POSITION OVERFLOW INTERRUPT Command header code (octal): 054111



When the graphic controller X or Y position overflow interrupt to the display processor is enabled (by a host-to-GRAPHIC 7 IK or IS message), GCP+ sends an XI message to the host computer whenever the graphic controller determines that an X or Y position overflow condition has been created. An overflow condition exists if the two's complement value in either the X or the Y position register (DXR or DYR) of the graphic controller exceeds 1777_8 (+1023) or 2000_8 (-1024). An overflow condition is detected when bits 11 and 12 of the X position register are not the same. Word 1 of an XI message contains the contents of the graphic controller program counter (DPC). This is the address of the next instruction to be fetched by the graphic controller. Words 2 and 3, respectively, contain the contents of the graphic controller X and Y position registers (following execution of the instruction that caused the interrupt). Also contained in bits 14 and 15 of word 3 is the bank number associated with the XI message.

NOTE

When an X or Y position overflow condition is detected by the graphic controller, an interrupt to the display processor is generated and the graphic controller halts. GCP+ also disables further X, Y overflow interrupts.

5.3.5 KEYBOARD RELATED MESSAGES

The keyboard related group consists of the following messages:

HOST-to-GRAPHIC 7

ZR	Initialize scratchpad for alphanumeric keyboard no. 1
ZT	Initialize scratchpad for alphanumeric keyboard no. 2
ZS	Zero out scratchpad no. 1
ZU	Zero out scratchpad no. 2
LK	Light keys on function keyboard no. 1
LT	Light keys on function keyboard no. 2

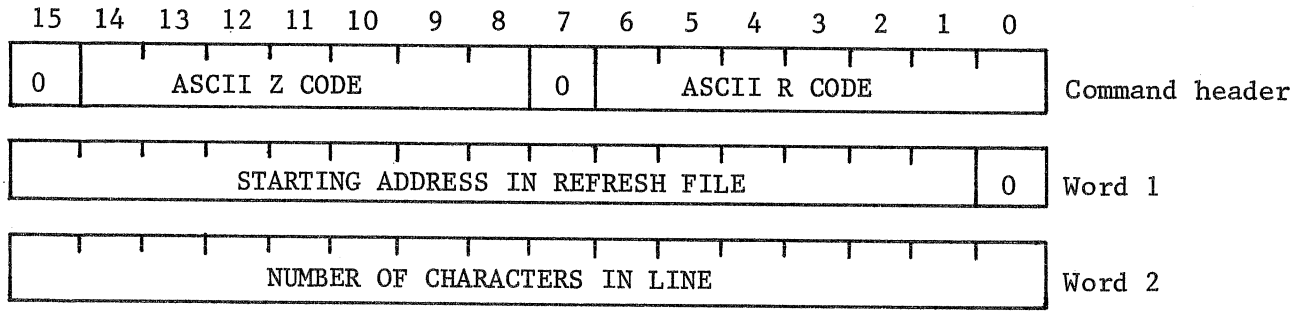
GRAPHIC 7-to-HOST

KY	Alphanumeric keyboard no. 1
KT	Alphanumeric keyboard no. 2
XR	Scratchpad ready for alphanumeric keyboard no. 1
XT	Scratchpad ready for alphanumeric keyboard no. 2
RK	Function keyboard no. 1
RL	Function keyboard no. 2

The following paragraphs discuss these messages and give details concerning the format and application of each.

ZR (H→G7) INITIALIZE SCRATCHPAD FOR ALPHANUMERIC KEYBOARD NO. 1

Command header code (octal): 055122



The ZR message is a three-word message used to establish parameters for handling alphanumeric characters on a line basis from the keyboard connected to serial interface port 3. It is used in conjunction with a refresh file that contains an area set aside for the storage of characters typed in by the operator. Such an area is referred to as a scratchpad. Typically, a scratchpad consists of a series of TXT (draw two tabular characters) instructions containing ASCII codes for spaces. When a ZR message is sent by the host computer, GCP+ is instructed to begin collecting characters from alphanumeric keyboard no. 1 and to store them in the refresh file starting at the address specified in word 1 (this address must be the address of an even-numbered byte). Word 2 specifies the total number of characters that may be collected (this number may be equal to or less than the capacity of the scratchpad as required).

Characters are collected in the scratchpad until the total count specified in word 2 is reached. At that point, GCP+ sends an XR (scratchpad for alphanumeric keyboard no. 1 ready) message to the host computer. RETURN, which may be typed at any time, terminates collection of characters and causes GCP+ to send an XR (scratchpad for alphanumeric keyboard no. 1 ready) message to the host computer. The host computer can then obtain the contents of the scratchpad by sending a GI (give image) message to the GRAPHIC 7. Note that typing RETURN only causes an XR message to be generated and has no effect on the scratchpad itself. Additional inputs from the keyboard are simply added to the scratchpad if space is available or ignored if space is not available.

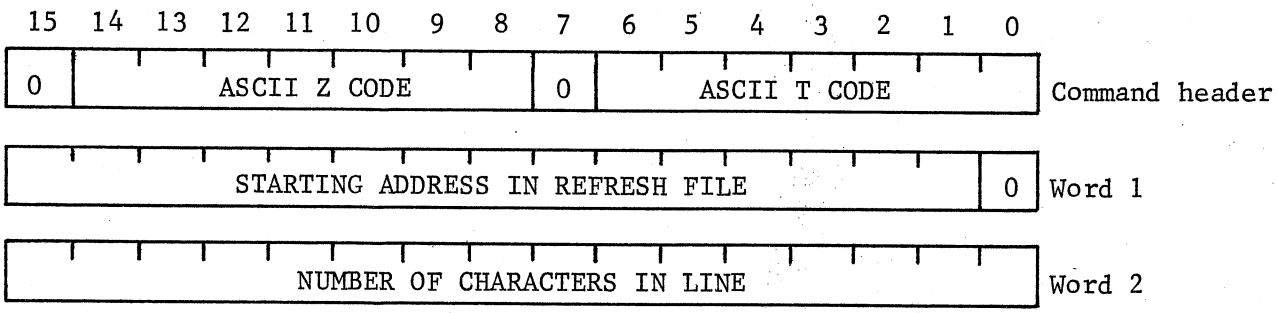
Characters collected in the scratchpad remain there until a) they are cleared by a ZS (zero scratchpad) or a SU (selective update) or MU (memory update) message from the host computer; b) they are replaced when another ZR message from the host computer causes the scratchpad to be reused; or c) RUB OUT is typed. Typing RUB OUT deletes the last character in the scratchpad and permits it to be replaced with a different character. Repeated typing of RUB OUT deletes successive characters in the reverse order of input.

NOTE

When processing on a line basis is no longer required, the host computer can cause keyboard inputs to be handled on a single character basis by sending a ZR message to the GRAPHIC 7 in which words 1 and 2 are all zeros. GCP+ then sends a KY (alphanumeric keyboard no. 1) message to the host computer each time a character is typed.

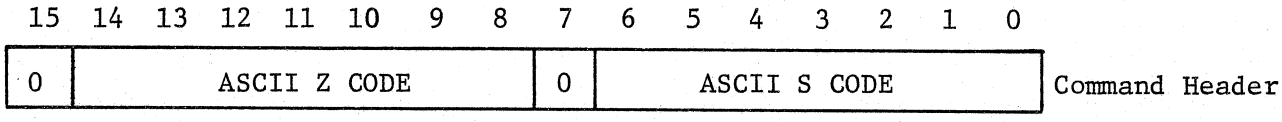
ZT (H→G7) INITIALIZE SCRATCHPAD FOR ALPHANUMERIC KEYBOARD NO. 2

Command header code (octal): 055124



The ZT message is exactly the same as the ZR message except that it applies to an alphanumeric keyboard connected to serial interface port 6 or 7. When the scratchpad for alphanumeric keyboard no. 2 is used, typing RETURN causes GCP+ to send an XT (scratchpad for alphanumeric keyboard no. 2 ready) message to the host computer.

ZS (H→G7) ZERO OUT SCRATCHPAD NO. 1 Command header code (octal): 055123

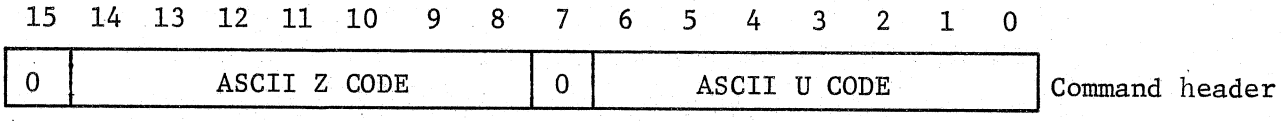


The ZS message is normally sent after an XR scratchpad message has been processed. The ZS message causes GCP+ to replace all characters in scratchpad no. 1 with spaces. After the ZS message is processed by GCP+, the scratchpad input pointer is repositioned to the beginning of the scratchpad area.

NOTE

The ZS message works in conjunction with the last ZR message sent from the host to the GRAPHIC 7. The ZR message establishes the starting address of the scratchpad in refresh memory and the size of the scratchpad.

ZU (H→G7) ZERO OUT SCRATCHPAD NO. 2 Command header code (octal): 055125



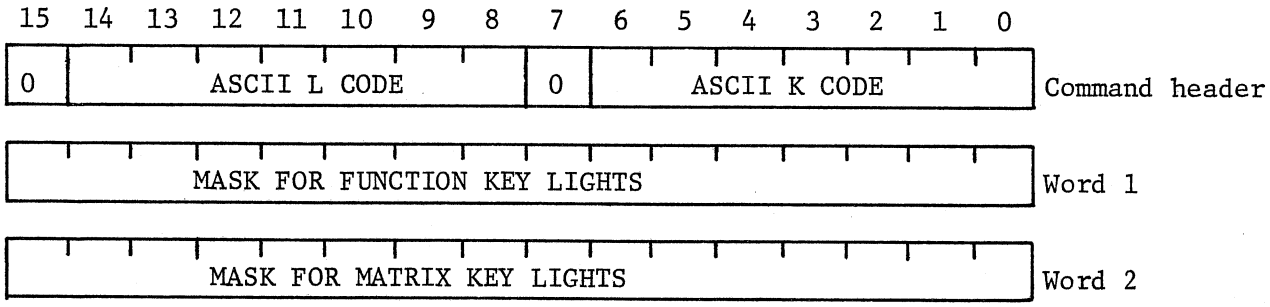
The ZU message is normally sent after an XT scratchpad message has been processed. The ZU message works in the same way as the ZS message except it's used for scratchpad no. 2.

NOTE

The ZU message works in conjunction with the last ZT message sent from the host to the GRAPHIC 7. The ZT message establishes the starting address of the scratchpad in refresh memory and the size of the scratchpad.

LK (H→G7) LIGHT KEYS ON FUNCTION KEYBOARD NO. 1

Command header code (octal): 046113



The LK is a three-word message used to light function and/or matrix keys on a keyboard connected to serial interface port 3. Bits 0 through 15 of word 1 are associated with function keys 0 through 15, respectively. Similarly, bits 0 through 15 of word 2 are associated with matrix keys 0 through 15, respectively. If a bit is set to 1, the corresponding key lights; if a bit is set to 0, the corresponding key does not light. The layout of the function and matrix keys is as follows:

FUNCTION KEYS

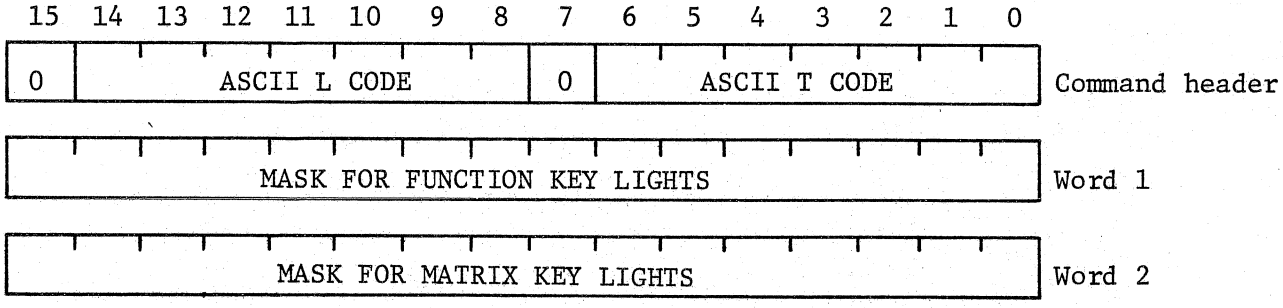
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

MATRIX KEYS

7	8	9	15
4	5	6	14
1	2	3	13
10	0	11	12

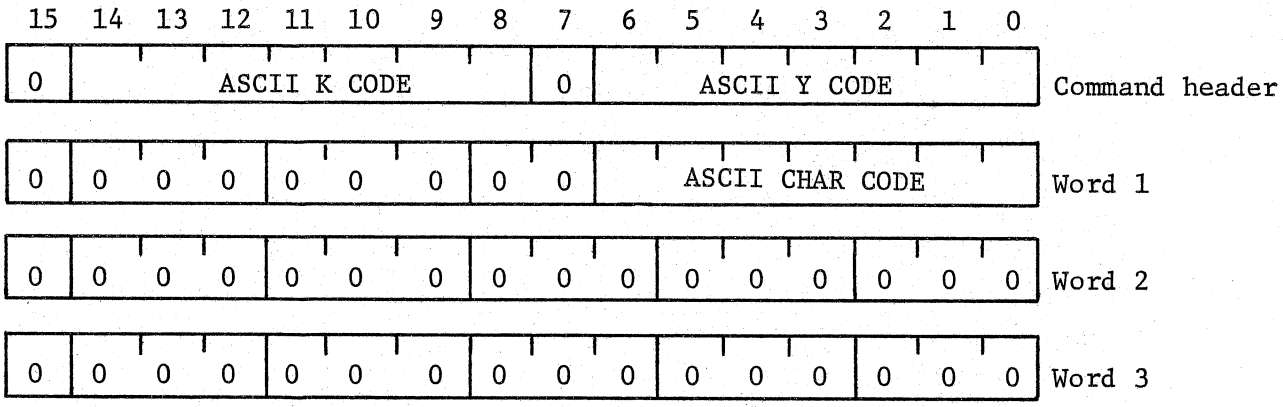
LT (H→G7) LIGHT KEYS ON FUNCTION KEYBOARD NO. 2

Command header code (octal): 046124



The LT message is exactly the same as the LK message except that it is used for a keyboard connected to serial interface port 7.

KY (G7→H) ALPHANUMERIC KEYBOARD NO. 1 Command header code (octal): 045531



KY messages are associated with alphanumeric inputs from a keyboard connected to serial interface port 3. Each time an alphanumeric key is typed, GCP+ sends a KY message to the host computer if the following conditions are met:

a. The keyboard is enabled (refer to host-to-GRAPHIC 7 IK messages).

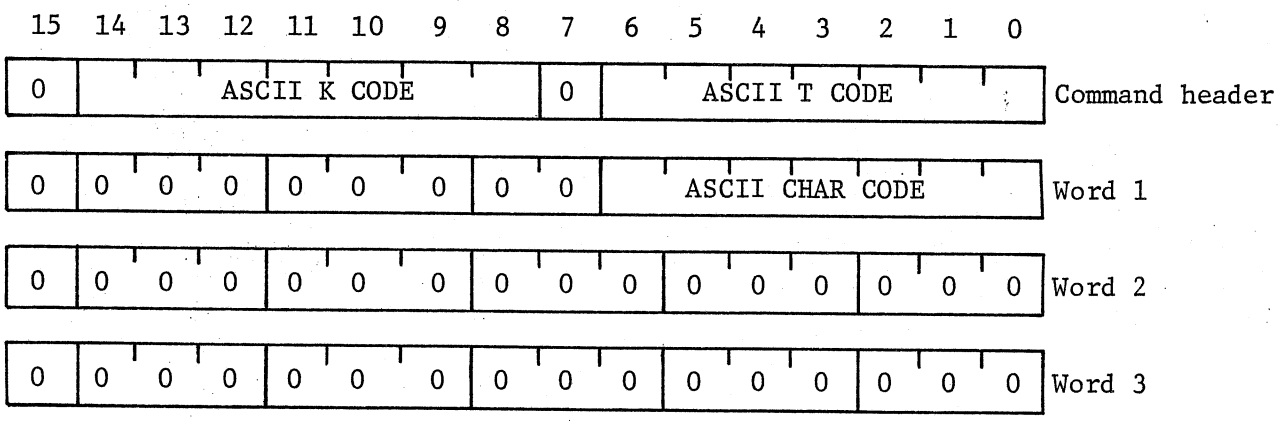
b. The keyboard is not being operated in the scratchpad mode (refer to host-to-GRAPHIC 7 ZR message).

If the keyboard is not enabled, typed inputs are ignored. If the keyboard is being operated in the scratchpad mode, inputs are sent to the host computer in RI (return image) messages. Each KY message sent to the host computer contains the ASCII code for a single alphanumeric character (refer to Appendix A for a summary of keyboard codes). This code is contained in the low order byte (bits 0-7) of word 1. Words 2 and 3 always contain all zeros.

NOTE

Keyboards are automatically enabled by GCP+ when the GRAPHIC 7 is initialized in the system mode.

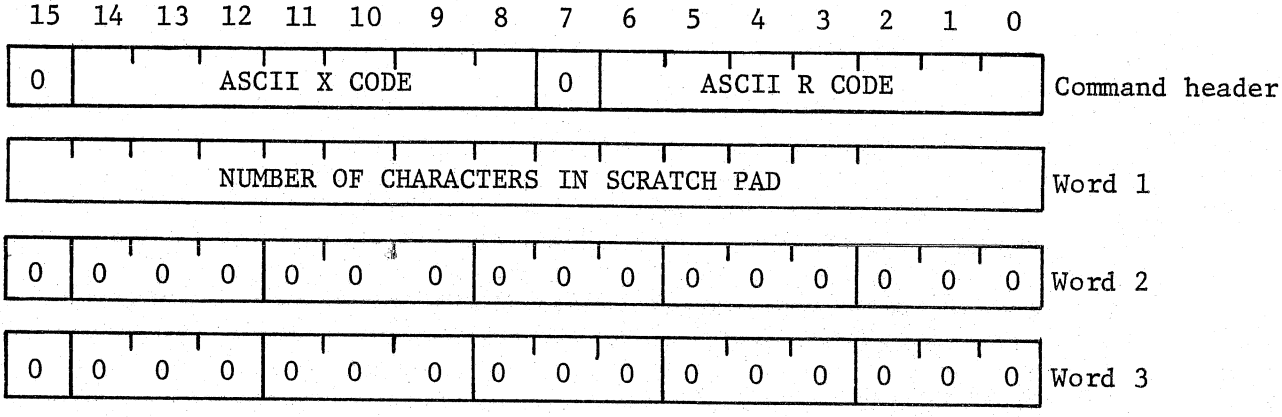
KT (G7→H) ALPHANUMERIC KEYBOARD NO. 2 Command header code (octal): 045524



The KT message is exactly the same as the KY message except that it is used for alphanumeric inputs from a keyboard connected to serial interface port 7 (the host-to-GRAPHIC 7 ZT message enables or disables scratchpad operation for keyboard no. 2).

XR (G7→H) SCRATCHPAD READY FOR ALPHANUMERIC KEYBOARD NO. 1

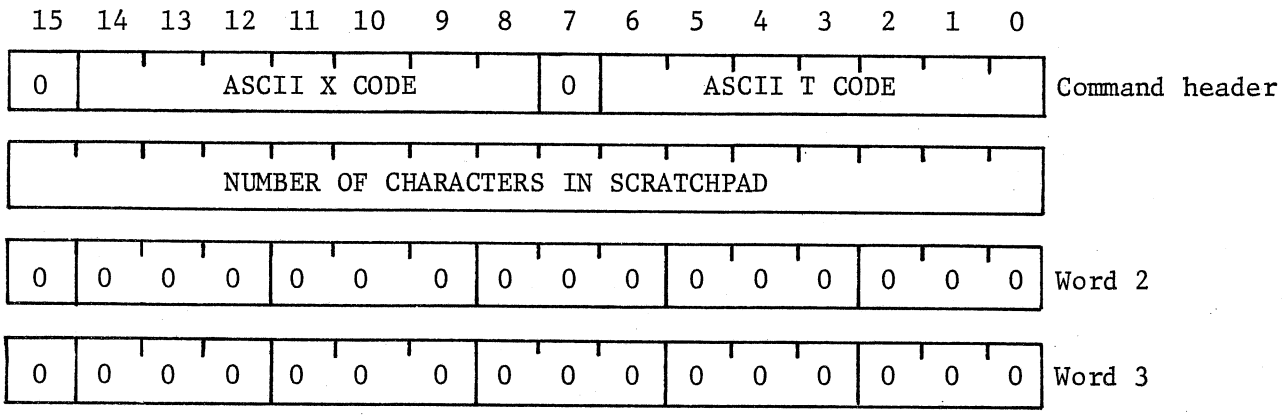
Command header code (octal): 054122



XR messages are generated by GCP+ to inform the host computer that data in the scratchpad for alphanumeric keyboard no. 1 (the alphanumeric keyboard connected to serial interface port 3) is ready to be transferred. GCP+ sends an XR message to the host computer whenever alphanumeric keyboard no. 1 is operated in the scratchpad mode (refer to host-to-GRAPHIC 7 ZR message) and the RETURN key is typed. GCP+ also sends an XR message to the host computer when the scratchpad is full. Word 1 contains the character count indicating the number of characters entered into the scratchpad by the operator. WORDS 2 and 3 always contain all zeros. Normally, the host computer responds with a GI (give image) message to obtain the contents of the scratchpad.

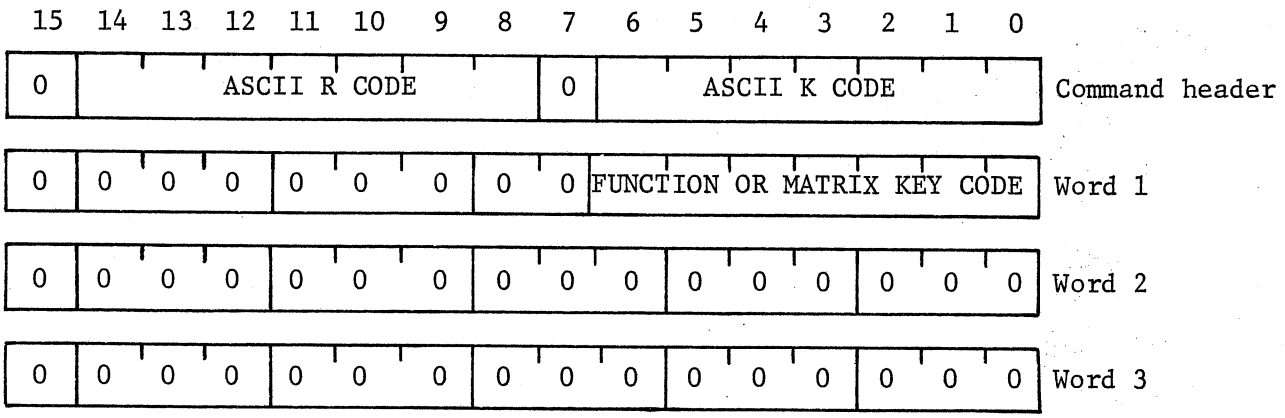
XT (G7→H) SCRATCHPAD READY FOR ALPHANUMERIC KEYBOARD NO. 2

Command header code (octal): 054124



The XT message is exactly the same as the XR message except that it is used for data in the scratchpad for alphanumeric keyboard no. 2 (the alphanumeric keyboard connected to serial interface port 7).

RK (G7→H) FUNCTION KEYBOARD NO. 1 Command header code (octal): 051113



RK messages are associated with the function or matrix keys on the keyboard connected to serial interface port 3. If function keyboard no. 1 has been enabled (refer to host-to-GRAPHIC 7 IK message), GCP+ sends an RK message to the host computer each time a key is typed. Each RK message contains the code for a single function or matrix key (refer to Appendix A for a summary of codes). This code is contained in the low order byte (bits 0-7) of word 1. Words 2 and 3 always contain all zeros.

RL (G7→H) FUNCTION KEYBOARD NO. 2 Command header code (octal): 051114

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	ASCII R CODE							0	ASCII L CODE							Command header
0	0	0	0	0	0	0	0	0	FUNCTION OR MATRIX KEY CODE							Word 1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 3

The RL message is exactly the same as the RK message except that it is used for function or matrix key inputs from the keyboard connected to serial interface port 7.

5.3.6 POSITIONAL ENTRY DEVICE RELATED MESSAGES

The positional entry device related group consist of the following messages:

Host-to-GRAPHIC 7

- TM Assign data tablet as PED no. 1
- TN Assign data tablet as PED no. 2
- IP Initialize PED no. 1
- IT Initialize PED no. 2
- GS Get status of PEDs
- GP Give PED no. 1
- GT Give PED no. 2

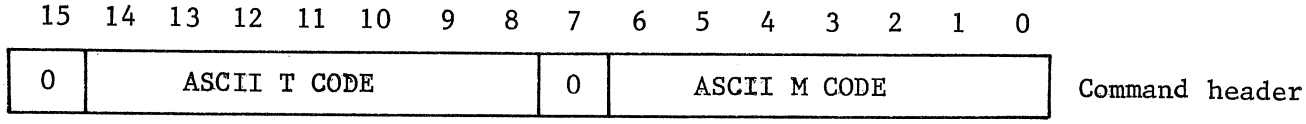
GRAPHIC 7-to-Host

- RT Return PED status
- RP Return PED no. 1
- RW Return PED no. 2

The following paragraphs discuss these messages and give details concerning the format and application of each.

TM (H→G7) ASSIGN DATA TABLET AS PED NO. 1

Command header code (octal): 052115



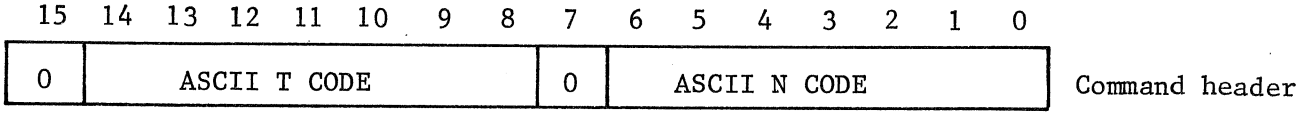
The TM message is used to inform GCP+ that all messages received on port 4 should be interpreted as data tablet type messages. By default, GCP+ is initialized to interpret all messages received on port 4 as trackball/forcestick type messages.

NOTE

Refer to IP message for more information on the data tablet message format.

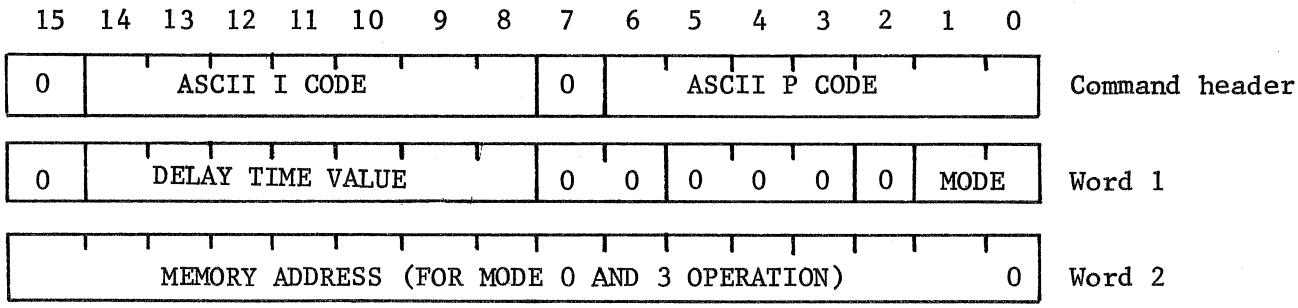
TN (H→G7) ASSIGN DATA TABLET AS PED NO. 2

Command header code (octal): 052116



The TN message is used to inform GCP+ that all messages received on port 8 should be interpreted as data tablet type messages. By default, GCP+ is initialized to interpret all messages received on port 8 as trackball/forcestick type messages.

IP (H→G7) INITIALIZE PED NO. 1 Command header code (octal): 044520



The IP message is a three-word message used to establish the operating mode for the PED (trackball/forcestick or data tablet) that is connected to serial interface port 4. Bits 1 and 0 of word one specify the operating mode in binary form as follows: automatic tracking mode (mode Ø) = 00; automatic mode (mode 1) = 01; request mode (mode 2) = 10; tracking mode (mode 3) = 11. Bits 8 through 14 of word 1 select the delay time when mode Ø is selected. For all other modes bits 8 through 14 of word 1 are set to zeroes. All remaining bits in word 1 are always set to zeroes. When mode Ø or 3 is specified, word 2 contains a memory address to be used for the storage of associated data. The address should be that of an even-numbered byte. If an odd address is specified, the low-order bit is ignored by GCP+ and no XX (error status) message is generated.

NOTE

Mode Ø, 2, and 3 are applicable to data tablet type PEDs and modes 1, 2, and 3 are applicable to trackball/forcestick type PEDs.

In the automatic tracking mode (mode Ø), absolute displacement data received from the data tablet is used to update the memory address specified in word 2 each time a data tablet message is received, to reflect the last position of the data tablet pen entry. This updating, which permits the CRT beam to track movements of the data tablet pen, is done by generating an LDXA (load X absolute) and an MVYA (move Y absolute) instruction to replace the ones already in the refresh file. When mode Ø operation for PED no. 1 is specified, word 2 of the IP message from the host computer contains the address of the LDXA instruction to be replaced. The new MVYA instruction then replaces the old MVYA instruction at the next higher address.

NOTE (Cont.)

Associated with mode Ø is the delay time value stored in word 1. The delay time values are given below:

BIT	DELAY TIME VALUE (seconds)
8	1/60
9	2/60
10	4/60
11	8/60
12	16/60
13	32/60
14	64/60

When the bit is set to 1, the associated delay factor is activated. When the bit is set to Ø, a zero delay factor is associated with the bit.

Each time the data tablet pen switch is pressed, the data tablet sends coordinate information to GCP+ at the rate of 100 messages per second. As each message is received, GCP+ does a data tablet-to-display coordinate system conversion and updates the memory address specified in word 2.

As soon as the pen switch is released, the delay time mechanism is activated. GCP+ then waits for whatever time the delay is set for and then sends an RP message to the host to reflect the latest position of the last data tablet entry. If the delay time is set for Ø seconds, then each time a data tablet message is received, an RP message is sent to the host computer. With a delay time of Ø seconds, the host computer could be overloaded with a series of identical RP messages. (E.g., if the data tablet pen switch remains pressed for 2 seconds, then 200 RP messages would have to be processed by the host computer.) The recommended delay time should be approximately ¼ second.

Each time the data tablet pen switch is pressed, the delay time mechanism is restarted. If the data tablet pen switch is re-pressed before the delay time expires, then no RP message is sent to the host computer until the new delay time expires.

NOTE (Cont.)

When the automatic tracking mode is selected for PED no. 1, the GP (give PED no. 1) message must not be sent from the host computer to the GRAPHIC 7. If a GP message is sent to the GRAPHIC 7 when PED no. 1 is operating in the automatic tracking mode, GCP+ responds by sending an XX message back to the host computer.

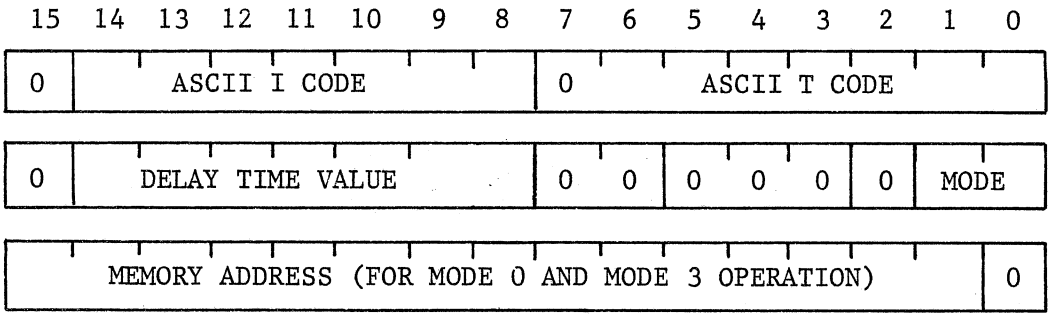
In the automatic mode (mode 1), relative displacement data received from the trackball/forcestick is sent to the host computer in an RP (return PED no. 1) message. An RP message is sent each time the display processor is interrupted by the PED. When the automatic mode is selected for PED No. 1, the GP (give PED no. 1) message must not be sent from the host computer to the GRAPHIC 7. If a GP message is sent to the GRAPHIC 7 when PED no. 1 is operating in the automatic mode, GCP+ responds by sending an XX message back to the host computer.

In the request mode (mode 2), GCP+ maintains the absolute coordinates of the PED position internally. Then, when a GP message is sent by the host computer, GCP+ returns the latest absolute position data to the host computer in an RP message.

In the tracking mode (mode 3), GCP+ maintains absolute PED position data and sends it to the host computer in the same manner as for mode 2 operation. In addition, GCP+ continuously updates the refresh file to reflect the latest position of the PED at all times. This updating, which permits the CRT beam to track movements of the PED, is done by generating an LDXA (load X absolute) and an MVYA (move Y absolute) instruction to replace the ones already in the refresh file. When mode 3 operation for PED no. 1 is specified, word 2 of the IP message from the host computer contains the address of the LDXA instruction to be replaced. The new MVYA instruction then replaces the old MVYA instruction at the next higher address.

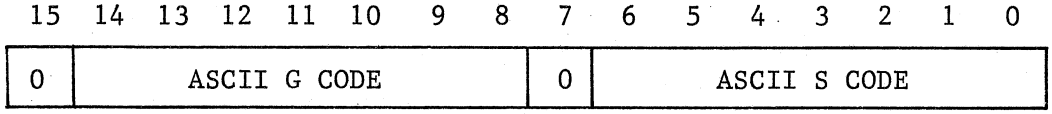
Note that relative position data is returned to the host computer in mode 1 operation while absolute position data is returned in mode Ø, mode 2, and mode 3 operation.

IT (H→G7) INITIALIZE PED NO. 2 Command header code (octal): 044524



The IT message is exactly the same as the IP message except that is applies to a PED connected to serial interface port 8. PED no. 2 data is requested by the host computer using GT (give PED no. 2) messages and sent to the host computer using RW (return PED no. 2) messages.

GS (H→G7) GET STATUS OF PEDS Command header code (octal): 043523

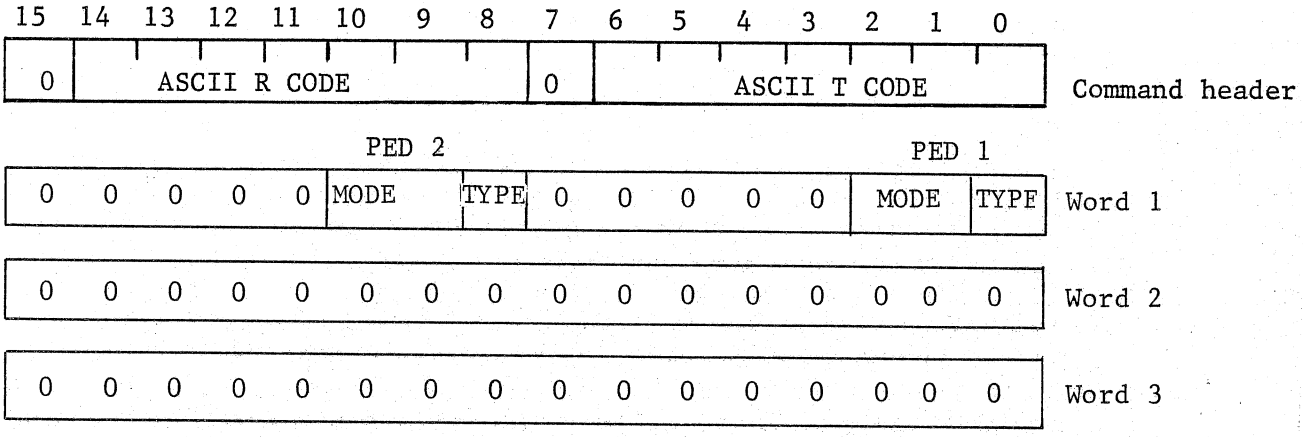


The GS message is used to request the current status of each PED. An RT message is sent by GCP+ to the host computer in response to the GS message.

NOTE

The GS and RT messages are maintenance type messages. Normally the GCP+ application programmer won't process the GS and RT messages but they can be used to validate the modes and PED types established by the IP, IT, TM and TN messages.

RT (G7→H) RETURN PED STATUS Command header code (octal): 051124

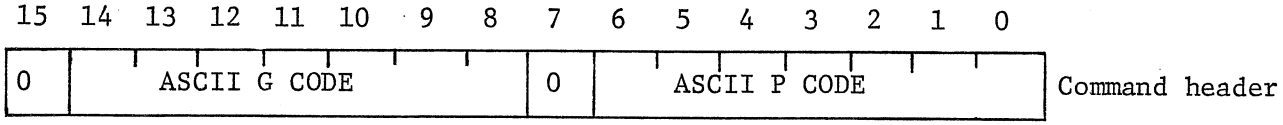


The RT message is sent by GCP+ to the host computer in response to a GS message. Word 1 contains the software status of each PED.

Bits 0, 1 and 2 are associated with PED 1 (PED connected to port 4 on serial multiport interface 1). Bits 8, 9 and 10 are associated with PED 2 (PED connected to port 8 on serial multiport interface 2). The meaning of the TYPE and MODE bits are given below:

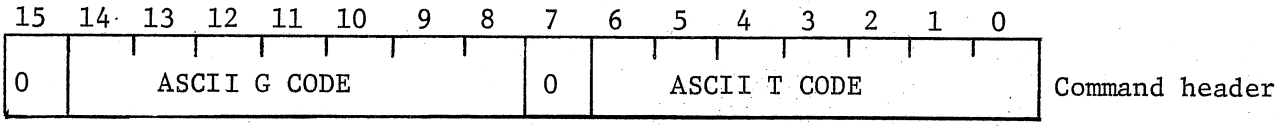
Bits	Value	Type PED
0 or 8	0	Trackball/forcestick
	1	Data tablet
1,2 or 9,10	00	Automatic tracking mode (data tablet only)
	01	Automatic mode
	10	Request mode
	11	Tracking mode

GP (H→G7) GIVE PED NO. 1 Command header code (octal): 043520



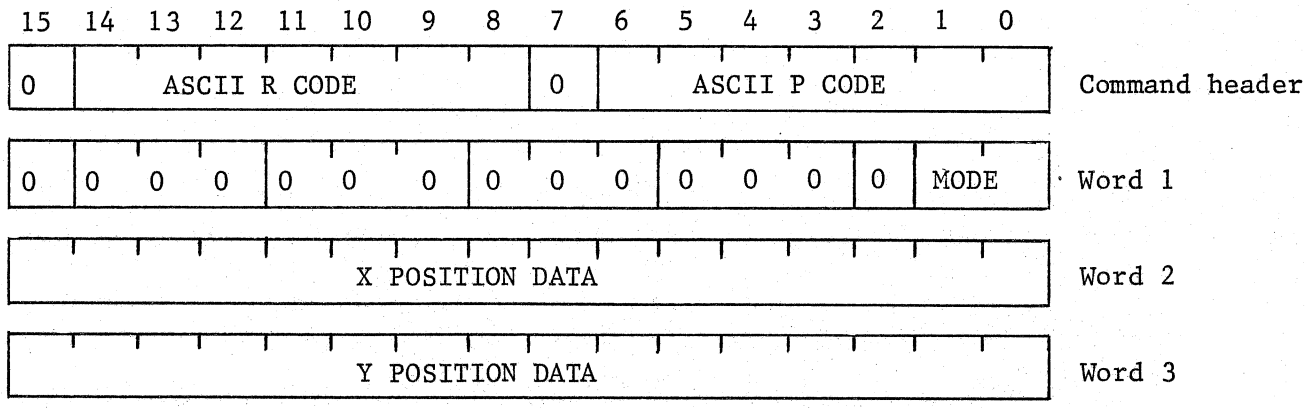
The GP message is a single-word message used to request the current absolute coordinate data for the PED (trackball/forcestick or data tablet) that is connected to serial interface port 4. Requested data is returned by GCP+ to the host computer using an RP (return PED no. 1) message. A GP message can be used only when PED no. 1 is operating in mode 2 (request mode) or mode 3 (tracking mode). If a GP message is sent when PED no. 1 is operating in mode Ø (automatic tracking mode) or in mode 1 (automatic mode), GCP+ responds with an XX (error status) message.

GT (H→G7) GIVE PED NO. 2 Command header code (octal): 043524



The GT message is exactly the same as the GP message except that it applies to a PED connected to serial interface port 8. Data requested by a GT message is returned to the host computer using RW (return PED no. 2) messages.

RP (G7→H) RETURN PED NO. 1 Command header code (octal): 051120



RP messages are associated with the PED (trackball/forcestick or data tablet) that is connected to serial interface port 4. When PED no. 1 is operating in the automatic tracking mode (mode Ø) or in the automatic mode (mode 1), RP messages are sent automatically by GCP+. When PED no. 1 is operating in the request or tracking mode (modes 2 and 3, respectively), RP messages are sent in response to GP (give PED no. 1) messages from the host computer. The operating mode for PED no. 1 is established by an IP (initialize PED no. 1) message from the host computer. For all RP messages, the operating mode of the PED is identified by bits 1 and 0 of word 1 (00, 01, 10, and 11 indicate modes 0, 1, 2, and 3, respectively). Bits 2 through 15 of word 1 are always zeros. When PED no. 1 is operating in mode Ø (data tablet only), GCP+ sends an RP message to the host computer every time the data tablet pen switch is pressed. In this mode words 2 and 3 contain absolute X and Y position data, respectively, for the PED.

When PED no. 1 is operating in mode 1, GCP+ sends an RP message to the host computer every time PED no. 1 generates an interrupt to the display processor (PED interrupts are enabled or inhibited by host-to-GRAPHIC 7 IK messages). In this mode, words 2 and 3 contain relative X and Y position data, respectively, for the PED (direction and distance moved since last RP message was sent). The relative data in each word consists of eight bits in two's complement form with the sign bit (bit 7) extended to fill the complete 16-bit word.

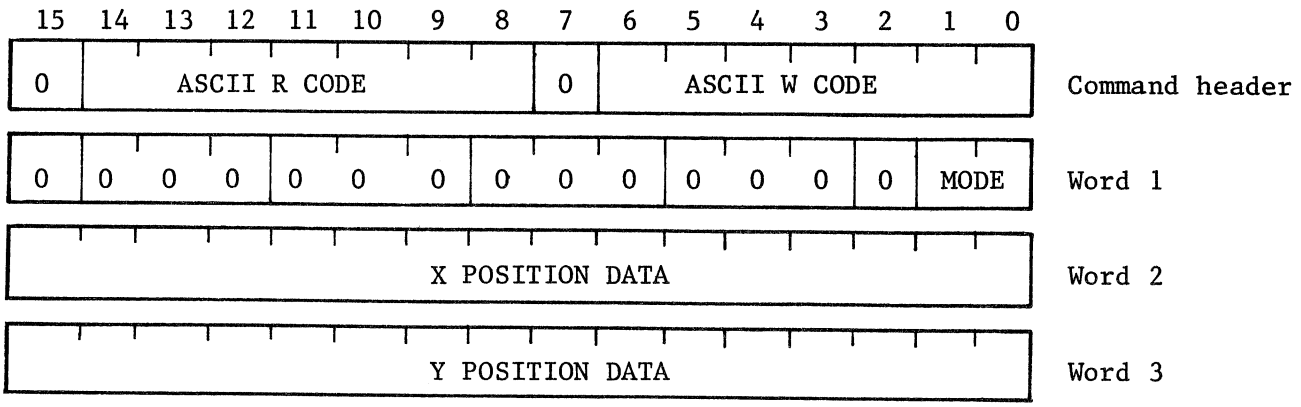
When PED no. 1 is operating in mode 2 or mode 3, GCP+ sends RP messages to the host computer in response to GP (give PED no. 1) messages from the host computer. In these modes, words 2 and 3 contain absolute X and Y position data for the PED.

The absolute data in each word consists of 12 bits in two's complement form with the sign bit (bit 11) extended to fill the complete 16-bit word. Note that PED interrupts are not used to initiate RP messages in mode 2 or mode 3.

NOTE

PED's are automatically enabled by GCP+ when the GRAPHIC 7 is initialized in the system mode.

RW (G7→H) RETURN PED NO. 2 Command header code (octal): 051127



The RW message is exactly the same as the RP message except that it is used for the PED that is connected to serial interface port 8. For mode 2 or mode 3 operation, RW messages are sent in response to GT (give PED no. 2) messages from the host computer.

5.3.7 PHOTOPEN RELATED MESSAGES

The PHOTOPEN related group consist of the following messages:

Host-to-GRAPHIC 7

PM	Change PHOTOPEN no. 1 mode
PP	Change PHOTOPEN no. 2 mode
LS	Link application program to scan routine

GRAPHIC 7-to-Host

PN	Return PHOTOPEN no. 1 strike/scan
PT	Return PHOTOPEN no. 2 strike/scan
SW	PHOTOPEN no. 1 switch activated
ST	PHOTOPEN no. 2 switch activated

The following paragraphs discuss these messages and give details concerning the format and application of each.

PM (H→G7) CHANGE PHOTOPEN NO. 1 MODE Command header code (octal): 050115

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	ASCII P CODE							0	ASCII M CODE							Command header
X	X	X	X	X	X	X	X	X	X	X	INDICATOR				MODE	Word 1

The PM message is used to establish the operating mode for PHOTOPEN no. 1. The PHOTOPEN can be operated in either a strike mode or a scan mode. By default, GCP+ is initialized to strike mode. In the strike mode, when GCP+ detects PHOTOPEN strikes, a PN message is returned to the host computer. The PN message contains coordinate information related to the display object that light was detected from.

The scan mode is used to identify the X, Y coordinates for a blank area on the screen. In the scan mode, when the PHOTOPEN switch is pressed, a grid pattern is flashed on the screen, and when light is detected by the PHOTOPEN, the grid pattern is removed and a PN message is returned to the host computer. The PN message contains the X, Y coordinates of the blank area selected by the PHOTOPEN.

NOTE

When the scan mode is used, the PHOTOPEN switch must be pressed in a blank area on the screen. If the PHOTOPEN switch is pressed in an area that already has an image displayed, then an incorrect PN message may be returned to the host computer.

The PHOTOPEN works on a principle of detecting light. If the PHOTOPEN grid pattern is flashed in an area that already has an object displayed, then it won't be able to differentiate between the object displayed and the grid pattern. (I.e., light is detected on either the object displayed or the grid pattern, depending on which instruction is currently being executed by the graphic controller at the time the grid pattern is flashed.)

NOTE (Cont.)

If the scan mode is to be operated in areas that have objects displayed, then all objects in these areas should be made insensitive to PHOTOPEN strikes. By blocking the refresh code in these areas with LDDP instructions, these areas can be desensitized, by disabling PHOTOPENS during the time these objects are being drawn and re-enabling PHOTOPENS after these objects are drawn (refer to paragraph 3.3.4 for a description of the LDDP instruction).

The mode selected in word 1 is defined as follows:

BIT	VALUE	DESCRIPTION
0	0	Select strike mode
	1	Select scan mode

The indicators selected in word 1 are defined as follows:

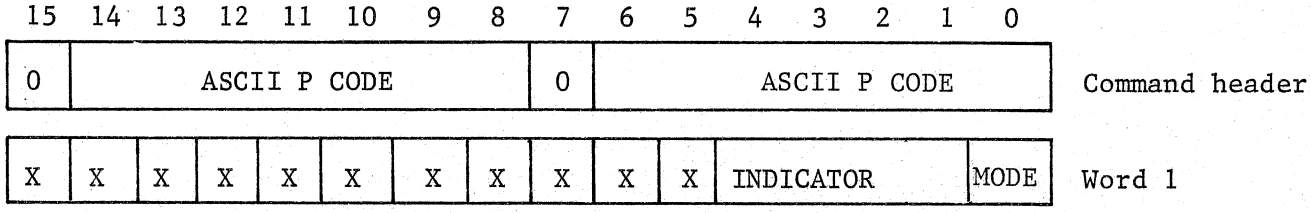
BITS	DESCRIPTION
4 3 2 1	
1 0 0 0	Enable grid pattern to be flashed on indicator 1
0 1 0 0	Enable grid pattern to be flashed on indicator 2
0 0 1 0	Enable grid pattern to be flashed on indicator 3
0 0 0 1	Enable grid pattern to be flashed on indicator 4

When strike mode is selected, GCP+ ignores the values assigned to the indicator bits.

When the scan mode is selected in the GCP+ environment, the user must also issue the SL message to link the scan routine to the application refresh program. When the scan mode is selected in the FSP environment, the scan routine is automatically linked to the application refresh program and there is no need to use the SL message.

Refer to paragraph 5.3.9 for more information on the FSP environment.

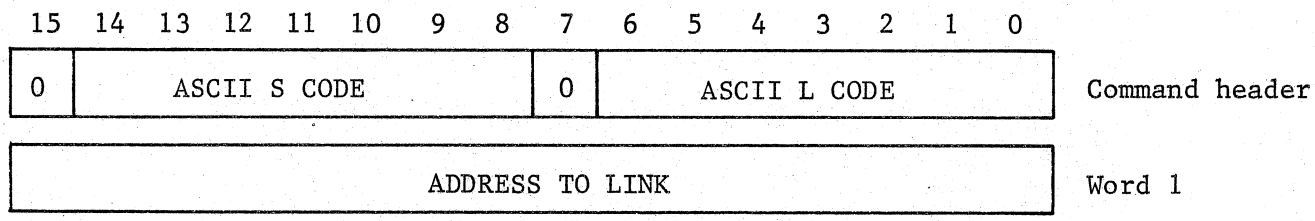
PP (H→G7) CHANGE PHOTOPEN NO. 2 MODE Command header code (octal): 050120



The PN message is used to establish the operating mode for PHOTOPEN no. 2. This message is similar to the PM message except it relates to PHOTOPEN no. 2.

SL (H→G7) LINK APPLICATION PROGRAM TO SCAN ROUTINE

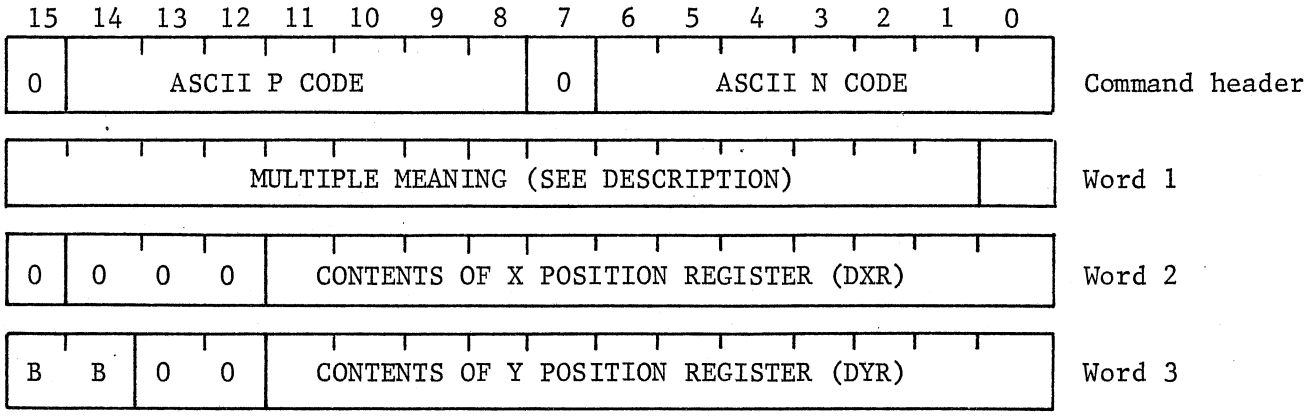
Command header code (octal): 046123



The SL message is used to link the scan routine to the GCP+ application refresh program. Word 1 contains the address in the user's refresh to link the scan routine. When GCP+ executes the SL message, it stores an absolute subroutine call at the address specified in word 1. It stores the address of the scan routine in the next word following the absolute subroutine call. The absolute address of the scan routine is kept track of internally by GCP+. The scan routine is stored in bank Ø. This requires that the address to link the scan subroutine to must always be in bank Ø.

PN (G7→H) RETURN PHOTOPEN NO. 1 STRIKE/SCAN

Command header code (octal): 050116



GCP+ returns PN messages to the host computer in response to PHOTOPEN activity. PN messages can be generated in the strike or scan modes. The meanings for PN messages in the GCP+ environment and the FSP environment are slightly different.

In the GCP+ environment, PN messages have the following meaning for strike mode. Strike interrupts are enabled by a host-to-GRAPHIC 7 IK or IS message. GCP+ sends a PN message to the host computer when light is detected by PHOTOPEN no. 1. The content of word 1 of a PN message is determined by the state of the PHOTOPEN item number flag which is controlled by an IK message from the host computer. If the PHOTOPEN item number flag is set to 1, word 1 contains the contents of the graphic controller general purpose register 1 (DR1) at the time the strike occurred. If the PHOTOPEN item number flag is cleared, word 1 contains the contents of the graphic controller program counter (DPC) which equals the address of the instruction being executed by the graphic controller at the time of the PHOTOPEN strike, plus 4 bytes. In all PN messages, words 2 and 3 respectively contain the contents of the graphic controller X and Y position registers (DXR and DYR). Also contained in bits 14, 15 of word 3 is the bank number that the strike occurred in. The values for bits 14 and 15 are given below:

BITS		
15	14	
0	0	Bank Ø
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

NOTE

When a PN message is returned in response to a strike interrupt, GCP+ disables further PHOTOPEN strike interrupts. Another IK or IS message must be sent to GCP+ to re-enable strike interrupts. If this is not done, no further PN messages will be generated.

In the GCP+ environment, a PN message has the following meaning for scan mode. Word 1 has a value of zero to indicate that the PN message relates to the scan mode. Words 2 and 3 contain respectively the X and Y coordinates that were selected by the operator when the scan pattern was flashed on the display indicator. Also contained in bits 14 and 15 of word 3 is the bank number that the scan pattern was flashed in. These bits should always be zero because the scan routine is located in bank Ø.

NOTE

In the scan mode all PHOTOPEN interrupts are controlled by GCP+ so the user need not be concerned with sending IK or IS messages to enable PHOTOPEN interrupts.

The FSP environment is entered when an IG message is sent to GCP+. When operating in the FSP environment, a PN message has the following meaning for strike mode. Word 1 contains the address of the FSP PHOTOPEN table. This table contains information related to PHOTOPEN strikes in the FSP environment. The information in the table is retrieved by sending a GI message to GCP+ with a word count of 6. The information contained in the FSP PHOTOPEN table is given below:

ADDRESS OR PHOTOPEN STRIKE	Word 1
TYPE OF OBJECT	Word 2
ADDRESS OF CALLING PAGE	Word 3
BANK OF CALLING PAGE	Word 4
TEXT BYTE	Word 5
ITEM NUMBER	Word 6

The types of OBJECTS that can be identified are given below:

WORD 2 VALUE	OBJECT
1	Text
2	Vector
3	Conic
4	Point
5	Short vector

Word 5, the text byte, has a value of 0 to indicate that the right byte of a text instruction was detected and has a value of 1 to indicate that the left byte of a text instruction was detected.

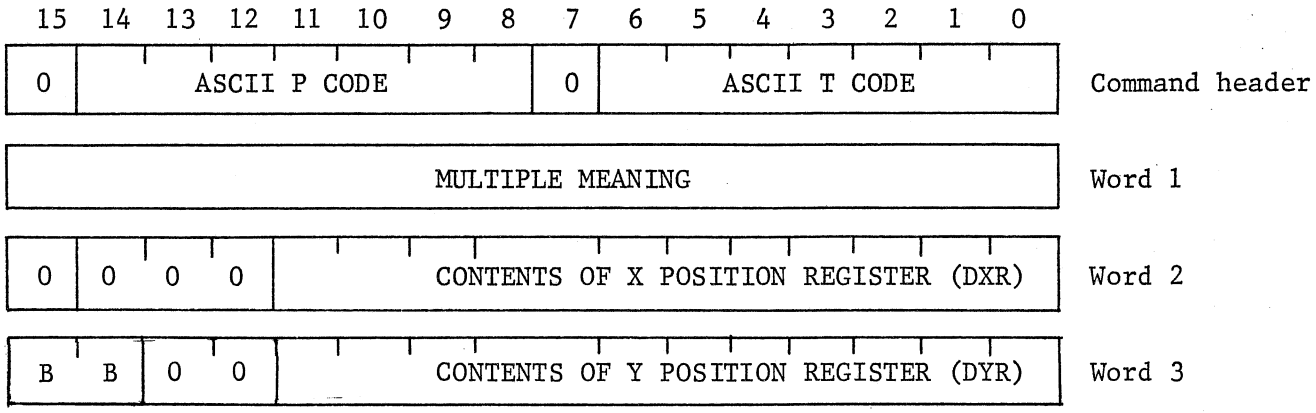
In the FSP environment, a PN message returned while in the SCAN mode has the same meaning as for the GCP+ environment.

NOTE

PHOTOPENs are not automatically enabled when the GRAPHIC 7 is initialized in the system mode.

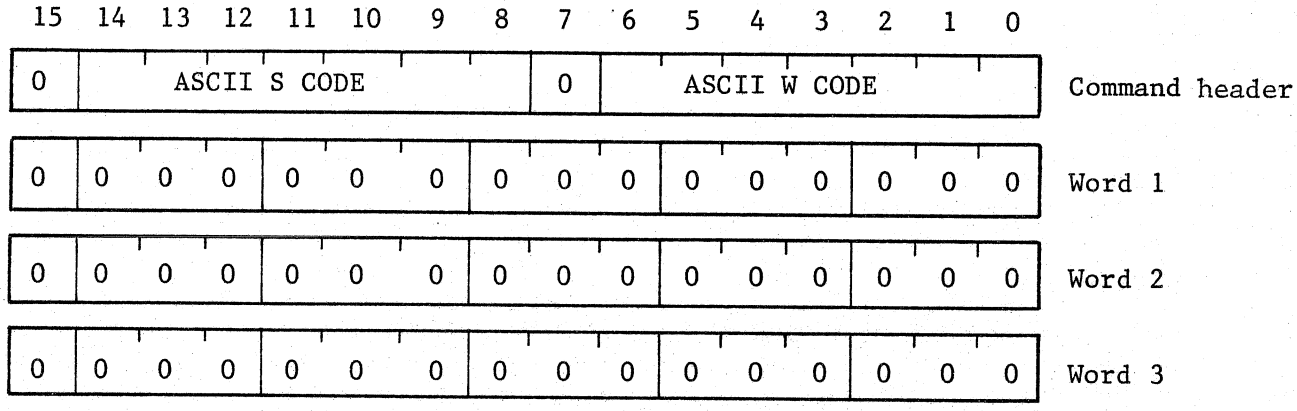
PT (G7→H) RETURN PHOTOPEN NO. 2 STRIKE/SCAN

Command header code (octal): 050124



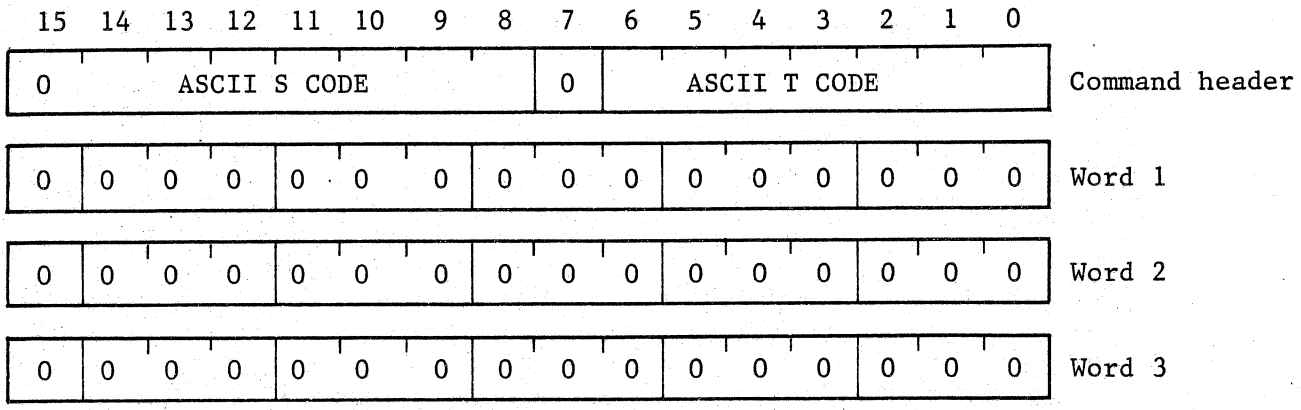
The PT message is exactly the same as the PN message except that it is used for PHOTOPEN no. 2 strikes.

SW (G7→H) PHOTOPEN NO. 1 SWITCH ACTUATED Command header code (octal): 051527



When PHOTOPEN no. 1 switch interrupts are enabled (by a host-to-GRAPHIC 7 IK or IS message), GCP+ sends an SW message to the host computer each time that the switch on PHOTOPEN no. 1 is pressed. Words 1 through 3 always contain all zeros. Note that PHOTOPEN switch interrupts are not enabled when the GRAPHIC 7 is initialized in the system mode.

ST (G7→H) PHOTOPEN NO. 2 SWITCH ACTUATED Command header code (octal): 051524



The ST message is exactly the same as the SW message except that it is used for PHOTOPEN no. 2 switch interrupts.

5.3.8 HARDCOPY MESSAGES

The hardcopy group consist of the following messages:

Host-to-GRAPHIC 7

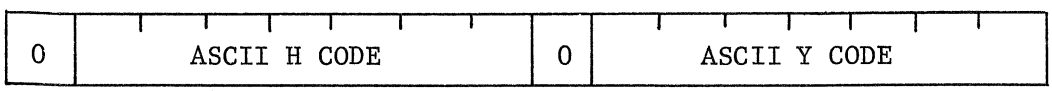
HY Initiate hardcopy

GRAPHIC 7-to-host

HK Hardcopy complete

The following paragraphs discuss these messages and give details concerning the format and application of each.

HY (H→G7) INITIATE HARD COPY Command header code (octal): 044131

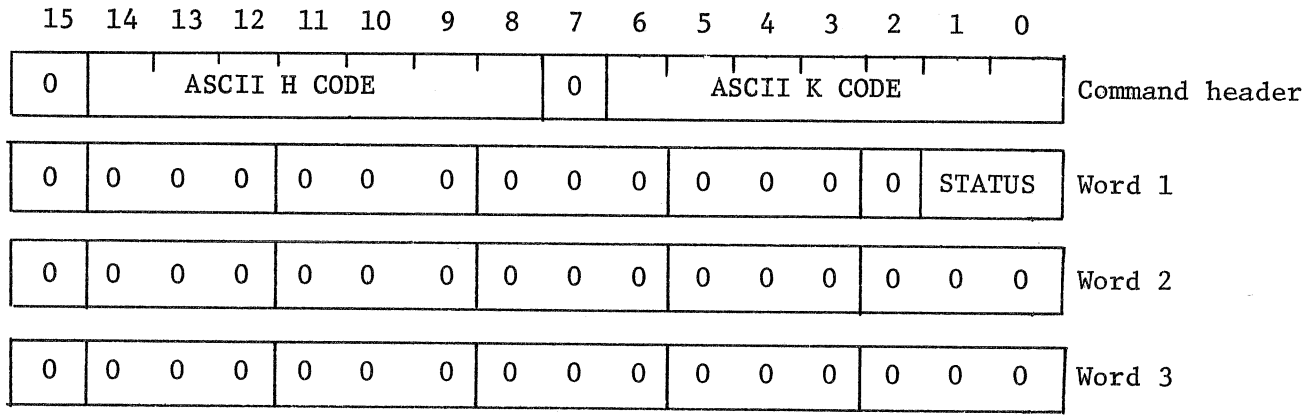


The HY message is used to request a hard copy of the image currently being directed to display #4. The hard copy unit is connected to serial port 5.

When the host requests a hard copy operation, GCP+ writes into the transmit data buffer (TBD5) to initiate the hard copy operation. The refresh file must contain an LDDZ instruction which selects the display #4.

An HK message is returned to the host computer when the hard copy operation is complete.

HK (G7→H) HARD COPY COMPLETED Command header (octal): 044113



The HK message is returned to the host computer upon completion of the requested hard copy operation.

Word 1 contains the completion status as follows in bits 0 and 1.

Value	Meaning
0	Successful
2	Hard copy unit connected but off-line or in manual mode, request rejected.

If an unexpected hard copy complete interrupt occurs, as when the button is pushed, it is ignored with no hard copy complete interrupt message sent to the host computer.

5.3.9 FORTRAN SUPPORT (FSP) MESSAGES

The Fortran support (FSP) group consists of the following messages:

Host-to-GRAPHIC 7

IG Initialize GCP+ to support FSP

GU Graphic update

MI Move image

NP Enable box display

ZP Disable box display

NN Enable error number

ZN Disable error number

LM Large memory in use for FSP

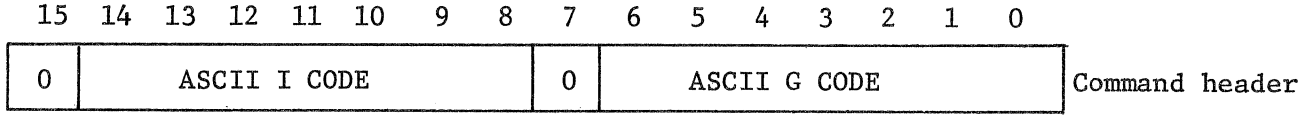
PV Packed vector

GRAPHIC 7-to-Host

RG Return FSP table address

The following paragraphs discuss these messages and give details concerning the format and application of each.

IG (H→G7) INITIALIZE GCP+ TO SUPPORT FSP Command header (octal): 044507



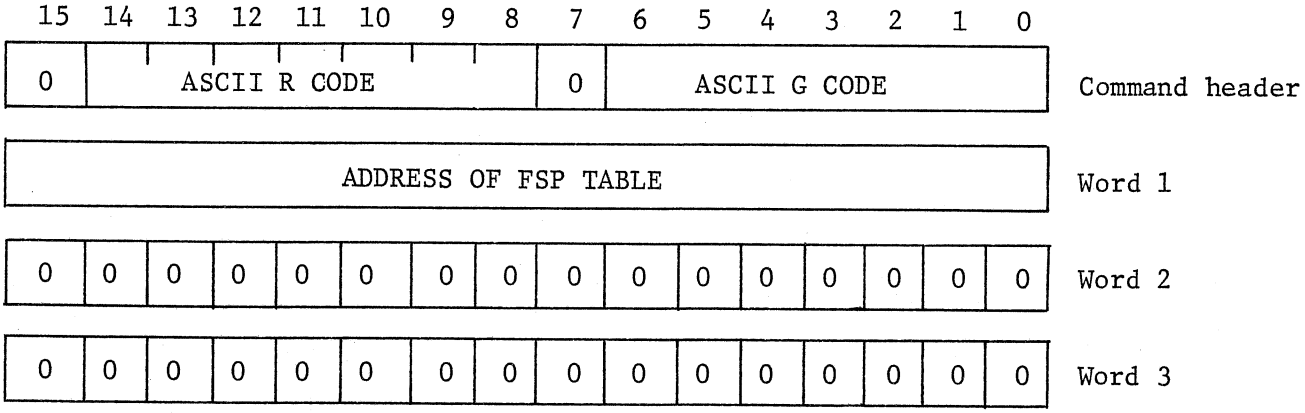
The IG message is used to initialize GCP+ to operate in the Fortran support program (FSP) environment. Associated with this environment is a Sanders-developed Fortran Graphic Support program. This program is host resident and consists of a collection of Fortran callable subroutines. This program simplifies the task of generating a GCP+ graphic program by enabling the application programmer to write all application programs in Fortran. The task of formatting GCP+ messages is performed by the FSP. (I.e., FSP converts Fortran subroutine calls into the equivalent GCP+ messages to generate the desired display image.)

The execution of the IG message results in a full screen box and an error code being displayed on all display indicators. This gives the application programmer a visual indication that GCP+ is now operating in an FSP environment. In response to the IG message, GCP+ sends an RG message to the host computer to indicate where all key addresses are located in the GRAPHIC 7. FSP uses these addresses to manage the refresh program associated with the FSP environment.

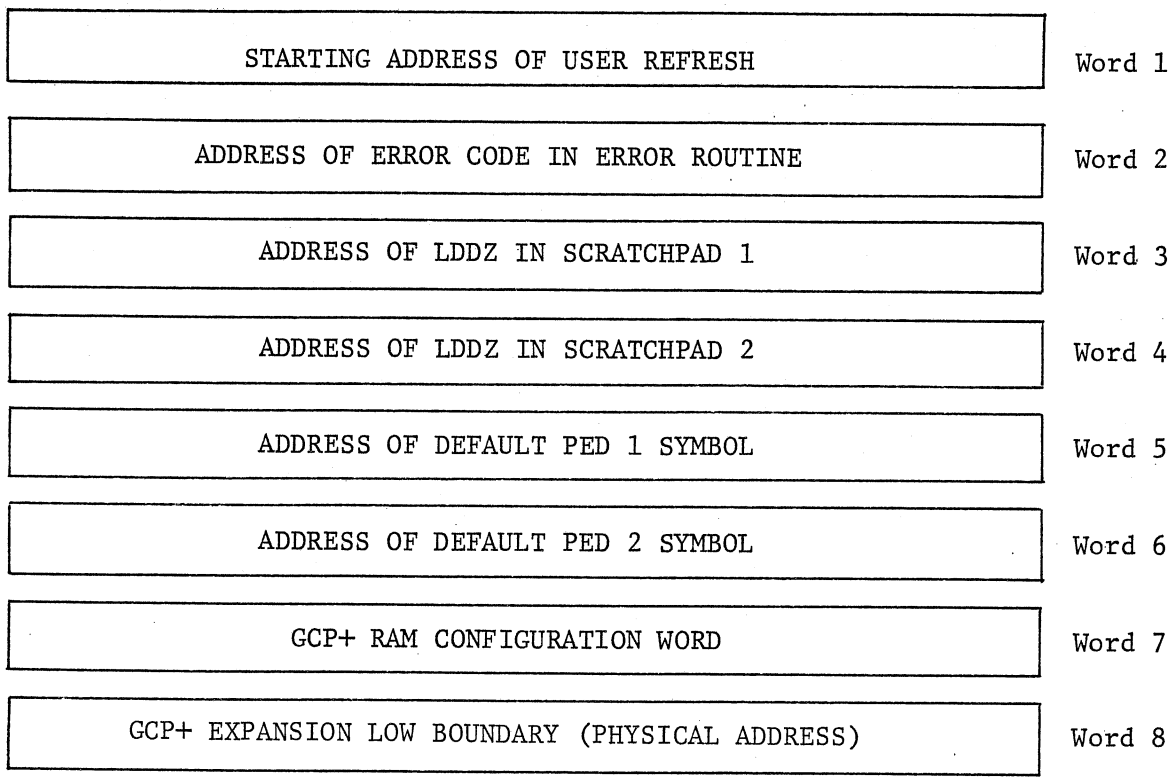
NOTE

Although the IG message is primarily intended for use in the FSP environment, the user has the option of developing his own host package to communicate with GCP+ in the FSP environment.

RG (G7→H) RETURN FSP TABLE ADDRESS Command header (octal): 051107

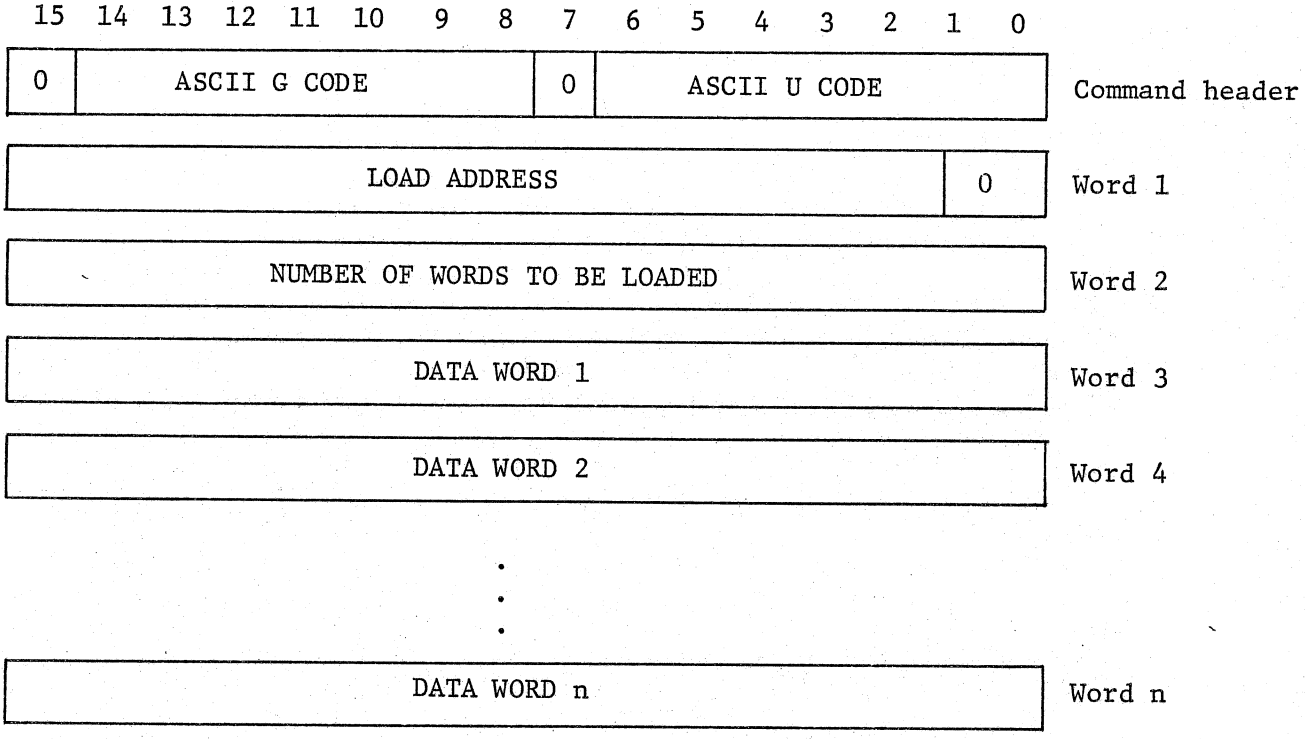


The RG message is returned to the host computer in response to the IG message. Word 1 contains the starting address of the FSP table in GRAPHIC 7 memory. This table contains all key addresses associated with the FSP refresh program that is started by the IG message. The addresses contained in this table are retrieved by sending a GI message with a word count of 10 (octal). The FSP table always resides in bank Ø and contains the following information:



Words 2 and 3 always contain zeroes.

GU (H→G7) GRAPHIC UPDATE Command header code (octal): 043525



The GU message is a variable-length message used to load data into read/write memory of the GRAPHIC 7. This message is primarily intended for use in the FSP environment but it can also be used by the GCP+ application programmer. The GU message is a special form of the MU and SU messages.

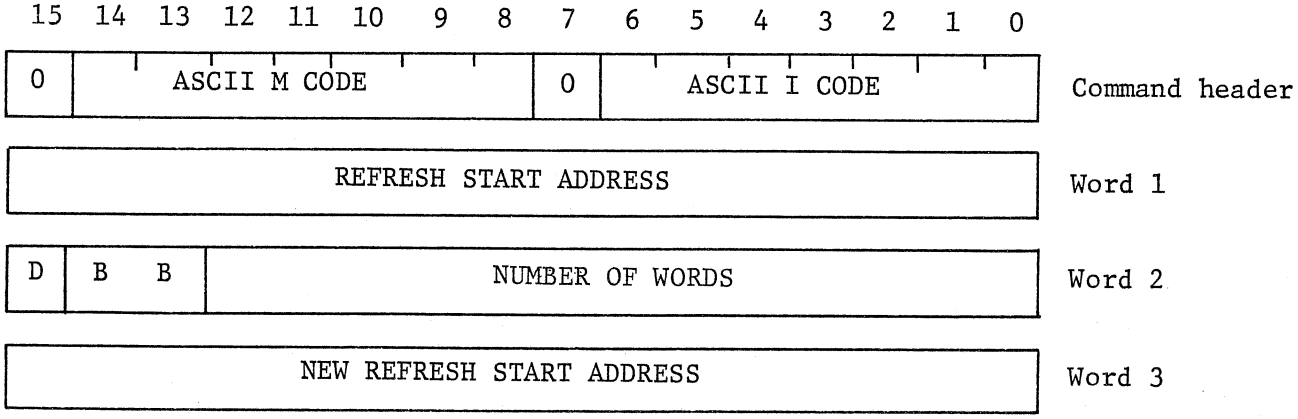
This message has been designed to maintain the validity of the refresh file during updates. This is done by loading words 4 through word n into read/write memory first. Then a return instruction is added following word n. Then word 3 is loaded into read/write memory.

The use of the GU message assumes that the user is operating in a subroutine environment (i.e., the first data word to be replaced contains a return instruction and that the end of the subroutine is identified by a return instruction).

NOTE

If no message has been sent to GCP+, the standard return (octal code 2300) is added after word n of the GU message. If an LM message has been previously sent to GCP+, an FSP coded return (octal code 0) is added after word n of the GU message. The coded return instruction is used by the FSP to permit subroutine calls between different banks.

MI (H→G7) MOVE IMAGE Command header code (octal): 046511



The MI message is a four word message that is primarily intended for use in the FSP environment. This message permits the copying of sections of refresh files to other areas of memory. Word 1 specifies the starting address of the refresh data to be copied to another area of memory. Bits 0 through 12 in word 2 define the number of successive words that should be transferred beginning with the refresh start address specified in word 1. Bits 13, 14, and 15 define the bank where the new refresh start address is located. These bits are defined as follows:

BIT 15

VALUE

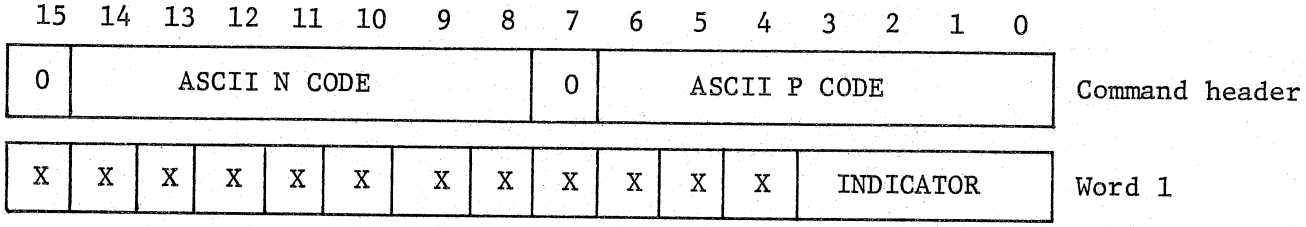
0	Transfer refresh data to current bank (ignore bits 13 and 14)
1	Transfer refresh data to bank specified in bits 13 and 14

BITS

14 13	DESTINATION BANK
0 0	0
0 1	1
1 0	2
1 1	3

When the last data word has been copied into the new refresh area, a return instruction is appended to the refresh data moved. If no previous LM message has been sent, the standard return (RTRN) instruction is added. If an LM message has been sent, the coded return (octal 0) instruction is added.

NP (H→G7) ENABLE BOX DISPLAY Command header (octal): 047120

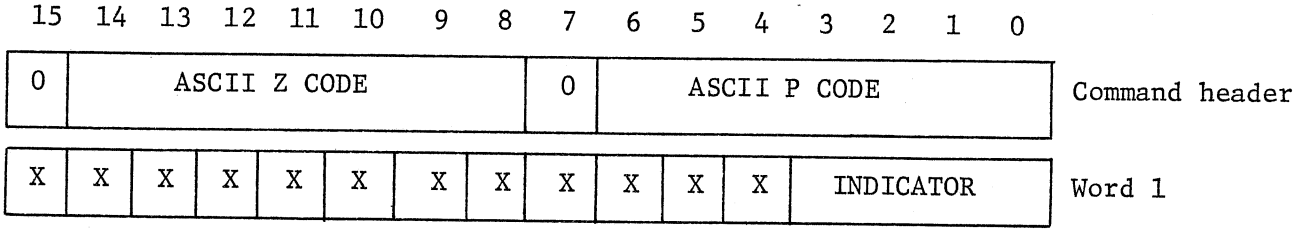


The NP message is used to enable the box display on selected indicators when operating in the FSP environment. Bits 4 through 15 in word 1 are ignored by GCP+. Bits 0 through 3 specify which indicators the box should be displayed on. These bits are defined as follows:

BITS 3 2 1 0	
1 0 0 0	Display box on indicator 1
0 1 0 0	Display box on indicator 2
0 0 1 0	Display box on indicator 3
0 0 0 1	Display box on indicator 4

If all bits are set to 1, then the box is displayed on all four indicators. Any combination of indicators is permitted.

ZP (H→G7) DISABLE BOX DISPLAY Command header (octal): 055120

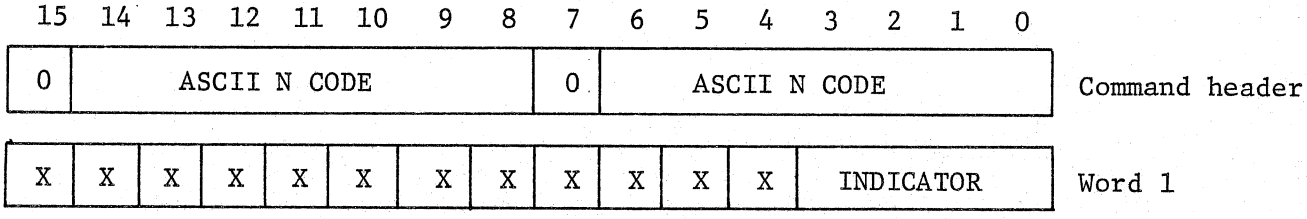


The ZP message is used to disable the box display on selected indicators when operating in the FSP environment. Bits 4 through 15 in word 1 are ignored by GCP+. Bits 0 through 3 specify which indicators the box should be removed from. These bits are defined as follows.

BITS	
3 2 1 0	
1 0 0 0	Remove box from indicator 1
0 1 0 0	Remove box from indicator 2
0 0 1 0	Remove box from indicator 3
0 0 0 1	Remove box from indicator 4

If all bits are set to 1, then the box will be removed from all four indicators.

NN ENABLE ERROR NUMBER Command header (octal): 047116

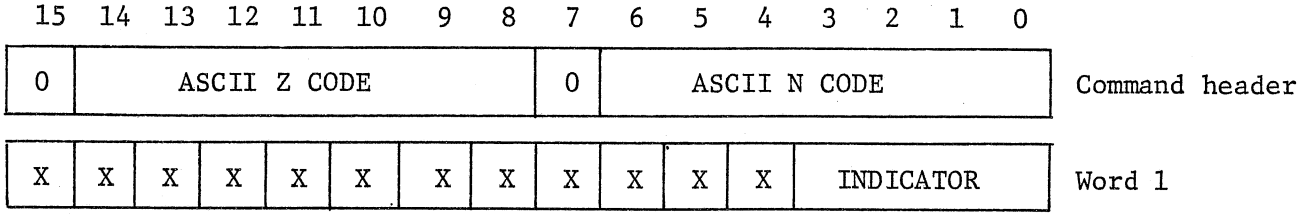


The NN message is used to enable the error number display on selected indicators when operating in the FSP environment. These error numbers are updated by FSP to give the user a visual indication that an error has occurred. Bits 4 through 15 are ignored by GCP+. Bits 0 through 3 specify which indicators the error number should be displayed on. These bits are defined as follows:

BITS 3 2 1 0	
1 0 0 0	Display error number on indicator 1
0 1 0 0	Display error number on indicator 2
0 0 1 0	Display error number on indicator 3
0 0 0 1	Display error number on indicator 4

If all bits are set to 1, then the error number is displayed on all four indicators.

ZN (H→G7) DISABLE ERROR NUMBER Command header (octal): 055116

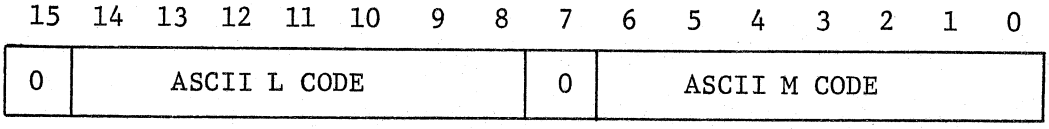


The ZN message is used to remove the error number display from selected indicators when operating in the FSP environment. Bits 4 through 15 are ignored by GCP+. Bits 0 through 3 specify which indicators the error number should be removed from. These bits are defined as follows:

BITS 3 2 1 0	
1 0 0 0	Remove error number from indicator 1
0 1 0 0	Remove error number from indicator 2
0 0 1 0	Remove error number from indicator 3
0 0 0 1	Remove error number from indicator 4

If all bits are set to 1, then the error number is removed from all four indicators.

LM (H→G7) LARGE MEMORY IN USE FOR FSP Command header code (octal): 046115



The LM message is primarily intended for use by the FSP. This message informs GCP+ that refresh data may reside in several banks.

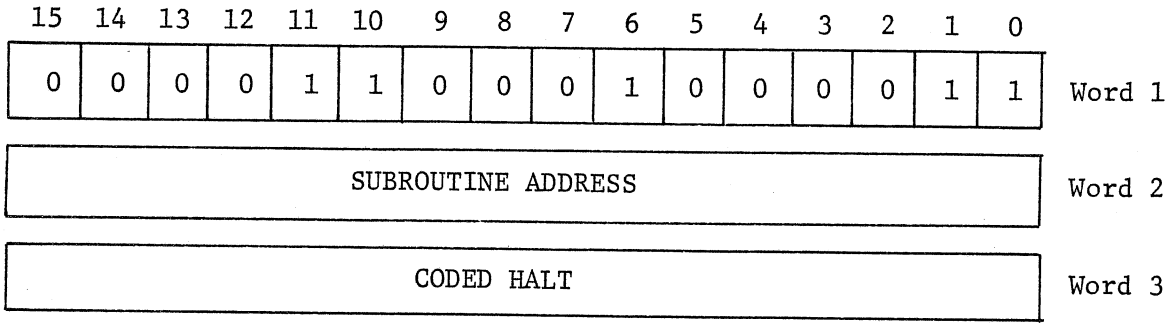
The graphic controller presently is unable to call subroutines between different banks. When the LM message is issued, a special halt interrupt vector is established. Coded halts with an associated DR3 value (address) pass the required arguments so the display processor can allow graphic subroutines to be executed from bank to bank. The display processor 'Q' register is used for a private graphics stack pointer. In all display processor 'SYSTEM' software codes, the following must be observed:

1) 'EXCQ's' shall only be used for this halt interrupt routine (OP code 767DD)

2) 'MUL', 'DIV' and 'ASHC' instructions should not be used. If it is absolutely essential, then this sequence can be used:

```
MOV @#PSW,-(SP)          ;save PSW (if not already done)
EXCQ - (SP)              ;save Q (.WORD 76746)
SPL 7                    ;disable interrupts
MUL #10.,R0              ;extended instruction
EXCQ (SP) +              ;restore Q (.WORD 76726)
MOV (SP)+,@#PSW          ;restore PSW
```

The following instructions are used to make subroutine calls between different banks in the FSP environment.



Word 1 contains an LDDI instruction to load the subroutine address into general purpose register 3 (DR3). Word 2 contains the starting address of the subroutine, relative to the beginning of the bank that the subroutine is stored at. Word 3 contains the coded halt. These are defined as follows:

Word 3 Value	DESCRIPTION
1	Call subroutine located in Bank Ø
2	Call subroutine located in Bank 1
3	Call subroutine located in Bank 2
4	Call subroutine located in Bank 3

NOTE

It is highly unlikely that a GCP+ application programmer will need to make subroutine calls between different banks; but if the need arises, the above mechanism could be used to make subroutine calls between different banks.

The LM message also informs GCP+ to process the coded return instructions. The standard return instruction, RTRN (octal code 2300) isn't used in the FSP large memory environment. All standard RTRN instructions are replaced with a coded RTRN instruction (octal code Ø). When processing GU and MI messages, GCP+ adds the coded RTRN instead of the standard RTRN instruction to the end of the refresh file.

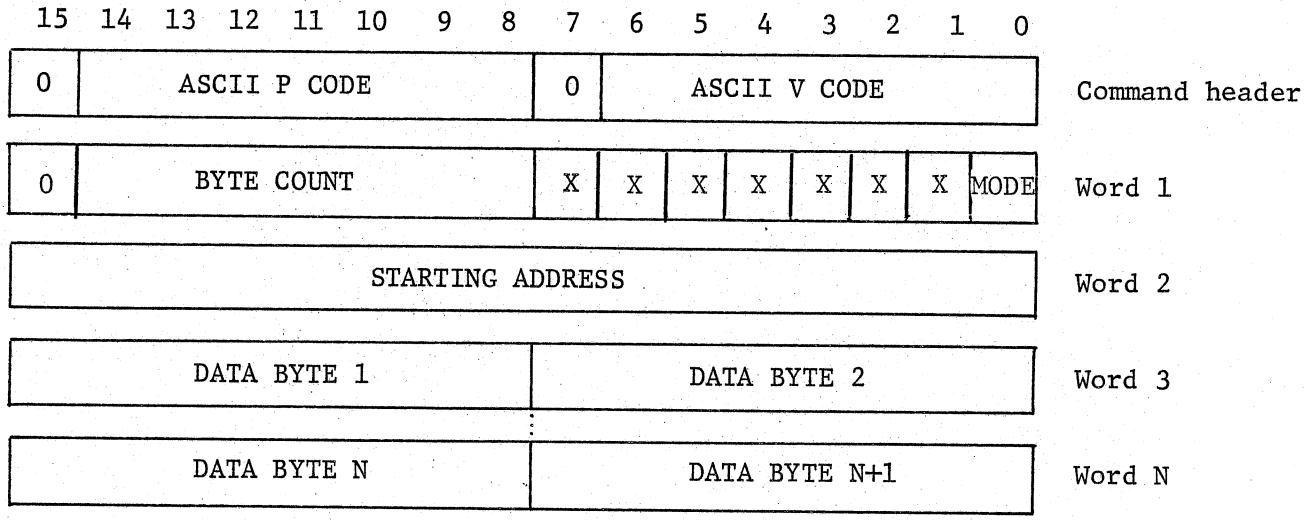
5.3.9.1 Packed Vector Mode

Packed vector mode is primarily intended for serial users running in the FSP environment. Using packed vector mode can result in a 4 to 1 speed increase when inserting absolute move (LDXA, MVYA) and absolute draw (LDXA, DRYA) instructions into refresh. Normal vectors are generated by sending an appropriate GU (or MU or SU) message. The GU message contains all of the LDXA, MVYA, and DRYA instructions needed to generate the desired image. These LDXA, MVYA, and DRYA instructions are created at the host computer. This method requires that large amounts of data be transmitted between the host computer and the GRAPHIC 7 to get these instructions stored in refresh.

When packed vector mode is used, a coded PV message is sent to the GRAPHIC 7. The PV message contains a series of ASCII characters that reflect the moves and draws that should be stored in refresh. The GRAPHIC 7 decodes the PV message and generates the equivalent LDXA, MVYA, and DRYA instructions and stores them in refresh.

The PV message is given below:

PV (H→G7) PACKED VECTOR Command header code (octal): 050126



Bits 8 through 14 in word 1 contain the byte count indicating the number of data bytes contained in the PV message. Bit Ø in word 1 selects the mode. When bit Ø is set to Ø, add mode is selected. For add mode, an appropriate return instruction is added to the end of the refresh code created from the data bytes contained in the PV message. If no LM message has been sent, the standard return (octal code 2300) is added to refresh. If an LM message has been sent, the coded return (octal code Ø) is added to refresh.

When bit Ø is set to 1, edit mode is selected. For edit mode, no return instruction is added to refresh. Bits 1 through 7 in word 1 are ignored by GCP+. Word 2 contains the starting address of where the first LDXA instruction should be stored.

NOTE

When word 1 and word 2 are sent from the host computer to the GRAPHIC 7, they must be sent according to the algorithm described in paragraph 5.2.1 for serial transmission of binary words (i.e., 4 characters for each binary word).

Words 3 through n contain the data bytes for the PV message. The format of the data bytes are given below:

BITS 15 THROUGH 8 AND 7 THROUGH 0	DESCRIPTION
X 0 0 1 1 1 1 1	Create move instead of draw
X 0 1 n n n n n	HI 5 bits of Y value or HI 5 bits of X value
X 1 0 n n n n n	LO 5 bits of X value
X 1 1 n n n n n	LO 5 bits of Y value

Table 5-2 relates the number of bytes that change at the host computer to the number of bytes required for transmission to generate the appropriate LDXA, MVYA, or DRYA instructions in the refresh file.

TABLE 5-2

BYTE TRANSMISSION REQUIREMENTS

BYTES WHICH CHANGE				BYTE TRANSMISSION REQ.				# OF BYTES SENT*
HI Y	LO Y	HI X	LO X	HI Y	LO Y	HI X	LO X	
1	1	1	1	1	1	1	1	4
1	1	1	0	1	1	1	1	4
1	1	0	1	1	1	0	1	3
1	1	0	0	1	1	0	1	3
1	0	1	1	1	0	1	1	3
1	0	1	0	1	0	1	1	3
1	0	0	1	1	0	0	1	2
1	0	0	0	1	0	0	1	2
0	1	1	1	0	1	1	1	3
0	1	1	0	0	1	1	1	3
0	1	0	1	0	1	0	1	2
0	1	0	0	0	1	0	1	2
0	1	1	1	0	1	1	1	3
0	0	1	0	0	1	1	1	3
0	0	0	1	0	0	0	1	1
0	0	0	0	0	0	0	1	1

** 0 = no transmission.
1 = transmit the byte containing that field.

* 1 extra byte will be sent on a MOVE to set to move mode.

** HI Y defined as bits 5-9 of user Y on a scale from 0 - 1023

LO Y defined as bits 0-4 of user Y on a scale from 0 - 1023

To change a HI X, you must send at least one LO Y.

NOTE

The host coordinate system is from 0,0 (lower left) to 1023, 1023 (upper right) and the display coordinate system is from -512, -512 (lower left) to +511, +511 (upper right). GCP+ maps 0,0 into -512, -512 and 1023, 1023 into +511, +511.

Below is a brief description of how the PV message functions:

1. The normal case (also initial value is):

X01AAAAA X11BBBBB
X01CCCCC X10DDDDD

GCP+ compares each byte as sent with old value (except on initial value). If same, do nothing until LO X value is sent, then create LDXA and MVYA or DRYA commands with the 10 bits of X and Y data. In the case shown above, since 4 bytes were sent, all 4 data values changed so old X, Y values are all replaced with new values.

	BEFORE (OLD VALUES)	AFTER	
HI Y	000KKKKK	000AAAAA	As soon as LO X byte is
LO Y	000LLLLL	000BBBBB	received, create LDXA with
HI X	000MMMMM	000CCCCC	A//B data and MVYA with
LO X	000NNNNN	000DDDDD	C//D data.

2. The concatenated data (A//B or C//D) is in the displayable range of the screen (with values from 0-1023). The data is converted to screen coordinates -512 to +511 before creating LDXA, MVYA, or DRYA instructions.

3. After commands have been created, they are added to the user refresh file. GCP+ checks the mode specified in the PV message to see whether a return must be added in addition to the MOVE and DRAW command sequence.

4. A LO X byte (bits 6 and 7 = 10) initiates creation of graphic instructions.

5. Review table 5-2 for further clarification of meaning of bytes sent. A Ø in the byte transmission column implies byte is not sent. (Ex. If only the lower 5 bits of Y value change, the following bytes are sent LO Y and LO X).

NOTE

All data bytes are valid ASCII characters (i.e., range is between 037 and 177 in octal). When the data bytes are transmitted from the host computer to the GRAPHIC 7, there is no need to code these bytes according to the algorithm previously described for serial transmission of binary words (i.e., words 3 through n are transmitted directly without any conversion performed at the host computer). PV messages can also be used on parallel interface systems but it is strongly recommended that PV messages not be used on parallel systems. No ASCII code conversion is required for parallel transmissions and the use of PV messages on such systems will probably result in a decrease of speed. For serial GCP+ users who are using applications that require the generation of large amounts of absolute moves and draws, the PV mode feature can be very useful. The routines needed at the host computer for PV mode are quite involved and as such are not included in this manual. On request, Sanders will provide additional information on the host routines needed to perform pack vector mode functions.

5.3.10 OPTION SUPPORT

Software options allow the GRAPHIC 7 to expand into a more specialized system while maintaining a common firmware program (i.e., GCP+). GCP+ includes a method for the user to load, test, initialize, and link several options together to enhance system requirements. There are a variety of option types that can be supported by GCP+. Some general types of options are listed below:

1. Sanders-developed software to support a present or future option (e.g., additional GCP+ messages to provide sophisticated 3-D coordinate converter support at the GRAPHIC 7 end).

2. Customer-developed software to meet a unique requirement (e.g., additional GCP+ messages to permit local editing of text at the GRAPHIC 7 end).

3. Sanders-developed control program (e.g., GET-2 emulator control program to effectively replace the GCP+ program).

4. Customer-developed control program (e.g., a special control program that effectively replaces the GCP+ program).

Normally the option software is stored on the expansion module. GCP+ also can support the downloading of options from a host computer.

NOTE

The option support provided by GCP+ is quite extensive and as such is not included in this manual. Refer to Sanders Publication H-79-0357 for a detailed description of all option support. This publication also contains information on writing customer-developed options.

5.3.10.1 Option Messages

The option group consist of the following messages:

Host-to-GRAPHIC 7

IY Initialize 2

GO Give option status

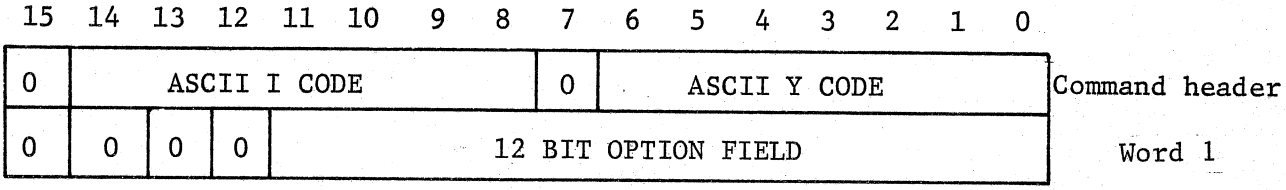
OU Option update (host downloading - described in option manual, H-79-0357)

GRAPHIC 7-to-Host

RO Return option

The following paragraphs discuss these messages and give details concerning the format and application of each.

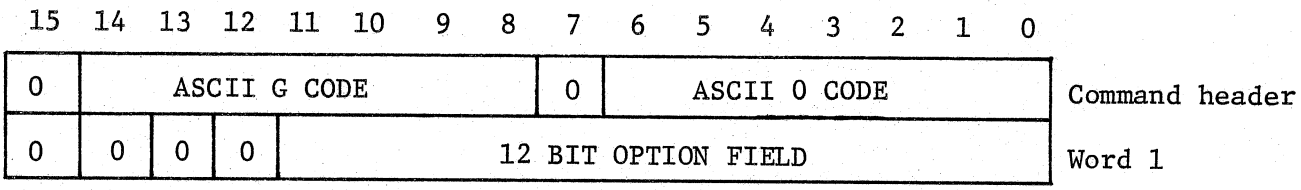
IY (H→G7) INITIALIZE 2 Command header code (octal): 044531



The initialize 2 message is a two word message that performs one of the following actions:

(OCTAL) OPTION FIELD	ACTION
Ø	Load all system automatic load options
7777	Unload all options
1 to 3777	Load specified option (if unloaded), initialize option, and update option status
4001 to 7776	

GO (H→G7) GIVE OPTION STATUS Command Header Code (octal): 043517



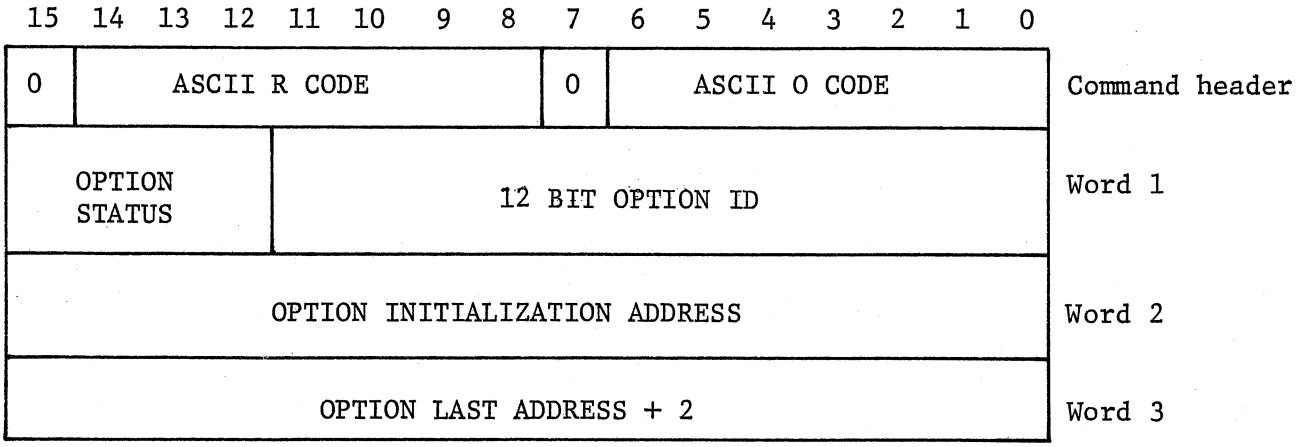
The give option status message is a two word message which allows the host to verify an option's status. One or two messages will be returned to the host as specified below.

OPTION FIELD	G7 ACTION
Ø	Return status of all options via RO, VL
Non-zero	Return status of specified option via RO

Two styles of RO option messages are returned to the host in response to the GO (give option) message: the single option status message and multiple option status message.

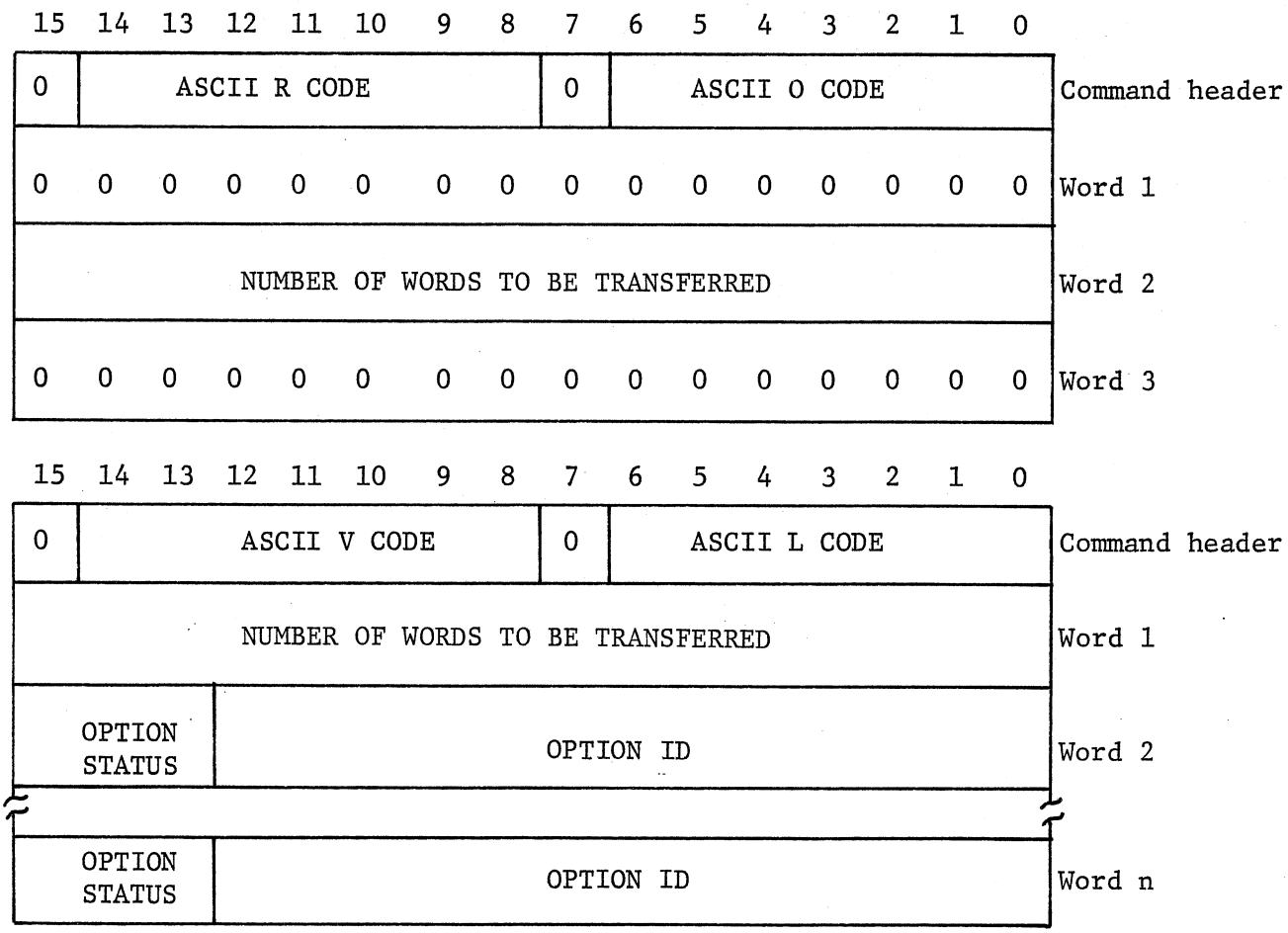
RO (G7→H) RETURN OPTION Command header code (octal): 051117

Single option status return.



RO (G7→H) RETURN OPTION Command header code (octal): 051117

Multiple option status return.



The number of words to be transmitted equal the option limit. A limit of zero prevents any VL message being returned. The option ID is returned in bits Ø through 11 of words 2 through n. Option ID values of zero shall be interpreted to mean that no option is loaded for that reserved area. The option status code is returned in bits 12 through 15 words of 2 through n. The meaning associated with the option status code are given in the following table:

(OCTAL) BIT 15 14 13 12	
0 0 0 0	Local detected option ,unloaded
0 0 0 1	Local checksum error ,unloaded
0 0 1 0	Local hardware not present ,unloaded
0 0 1 1	Local self test = NOGO ,unloaded
0 1 0 0	Local Self Test = GO ,loaded
0 1 0 1	Unfound option (for single RO message only)

5.4 PROGRAMMING COLOR DISPLAY INDICATORS

No special GCP+ messages are needed to program color display indicators. The GCP+ application programmer can control the color functions by using the standard LDRI graphic instruction. The LDRI instruction can be loaded into user refresh via the MU, SU, or GU messages.

The LDRI instruction for color is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	Octal 006010

x	x	x	x	DISPLAY SELECT	COLOR SELECT	Word 1

The display select bits in word 1 are defined as follows:

BIT	FUNCTION
11	Change color for display #1
10	Change color for display #2
9	Change color for display #3
8	Change color for display #4

The color select bits in word 1 are defined as follows:

BITS	COLOR SELECTED
7 6 5 4 3 2 1 0	
0 0 0 0 0 0 0 0	RED
0 0 0 0 0 0 1 0	ORANGE
0 0 0 0 1 0 1 0	YELLOW
0 0 1 0 1 0 1 0	GREEN

All other combinations of bits 0 through 7 are reserved for future color development.

NOTE

When programming color display indicators, a maximum of four color changes should be performed per refresh frame. GCP+ application programmers should organize refresh programs so that codes for each color are grouped together so that only four color changes are necessary. If more than four color changes per refresh frame are executed, there is no guarantee that the color change will actually take place. Hardwarewise, the color display indicator is protected against programming errors so the GCP+ application programmer need not be overly concerned if an error is made and the refresh program contains five color changes.

5.5 PROGRAMMING THE 3-D COORDINATE CONVERTER

By using the register update (RU) and the give register (GR) commands, the GCP+ programmer may read and write all registers associated with the 3-D coordinate converter.

This allows complete host control to perform such functions as:

- Set matrix parameters
- Set viewbox parameters
- Set perspective parameters
- Set various control parameters
    - Depth cueing select
    - Scale select
    - Refresh limits select
    - Source/destination of conversion process
    - Homogeneous/non-homogeneous select
    - 2D/3D select
    - Perspective/no-perspective select
- Start 3-D coordinate converter
- Activate 3-D coordinate converter for a PHOTOPEN search
- Selectively establish the desired interrupt control

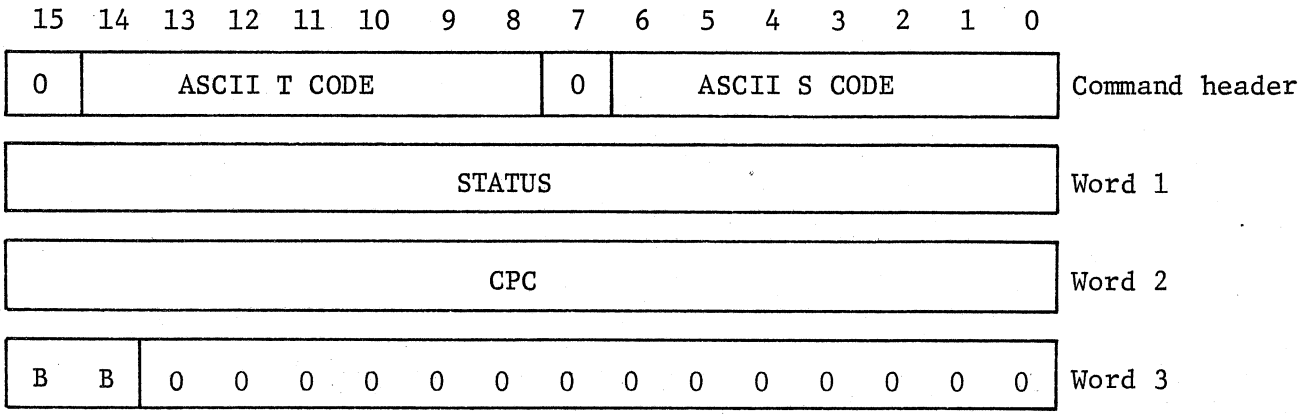
When 3-d interrupts are generated, an appropriate TS message is returned to the host computer.

NOTE

Please refer to Sanders Publication H79-0350 for more information on the 3-D coordinate converter.

TS (G7→H) 2-D/3-D Coordinate Converter Status

Command header code (octal): 052123



The 2-D/3-D coordinate converter can generate 16 interrupt conditions, provided that the corresponding mask bits are enabled. The TS message is returned to the host computer when a 3D coordinate converter interrupt condition occurs.

Word 1 contains the contents of the 2-D/3-D coordinate converter status register. Each bit in this register corresponds to an interrupt condition. One or more of these bits sets to indicate the type of interrupt condition detected.

Word 2 is the value of 2-D/3-D coordinate converter program counter.

Word 3 contains two bits of the 2-D/3-D coordinate converter block register corresponding to the bank in which the coordinate converter was executing at the time of the interrupt condition. Bits 14 and 15 are defined as followed:

Bits 15 14	Bank Number
0 0	0
0 1	1
1 0	2
1 1	3

SECTION 6
GRAPHIC CONTROL PROGRAM USAGE

6.1 GENERAL

This section contains information concerning basic usage of the graphic control program (GCP+). Included are startup procedures, procedures for generating and manipulating a refresh file, and information for using optional GRAPHIC 7 equipment.

6.2 STARTUP PROCEDURES

Startup procedures consist of initializing the GRAPHIC 7 in the system mode and ensuring that an XX (error status) message indicating zero errors is sent by GCP+ to the host computer. In certain cases, these operations are performed automatically. In other cases, action must be initiated by the host computer. The following paragraphs describe startup procedures for typical operating conditions.

6.2.1 GRAPHIC 7 TURNED ON AFTER HOST COMPUTER

When power is applied to the GRAPHIC 7, it is automatically initialized in the system mode and an XX message is sent to the host computer. If the host computer application program is running at the time and no errors are indicated by the XX message (bit 15 should be one and bits 0 through 14 should be zeros), no further action is required and startup is complete.

NOTE

On systems that do not have a 2-D/3-D coordinate converter option installed, bit 4 of the XX message is set to 1 to indicate failure of the 3-D self test.

6.2.2 GRAPHIC 7 TURNED ON BEFORE HOST COMPUTER

If the GRAPHIC 7 is turned on before the host computer, any XX message sent to the host computer is lost. In this case, action must be initiated by the host computer to obtain another XX message from the GRAPHIC 7. This is done by sending

an IZ (initialize) message to the GRAPHIC 7 which causes GCP+ to return an initialization XX message to the host computer.

6.2.3 POWER FAILURE STARTUP

To ensure proper startup following power failure, it should be assumed that the power failure affected both the GRAPHIC 7 and the host computer and that power is first restored to the GRAPHIC 7. This condition is similar to that described in paragraph 6.2.2 in that the XX message automatically generated by the GRAPHIC 7 is lost. To ensure proper startup, therefore, the power recovery routine in the host computer should cause an IZ message to be sent to the GRAPHIC 7 causing GCP+ to respond by sending an initialization XX message to the host computer. In this way, an initialization XX message is guaranteed to be received by the host computer regardless of the order in which power is restored to the various equipments.

6.2.4 STARTUP WITH GRAPHIC 7 IN TELETYPEWRITER EMULATION MODE

A special startup procedure is required when the teletypewriter emulation capability of the GRAPHIC 7 is used for communications with the host computer. This capability is used when the host computer is a time-sharing system or when loading and running the host application program must be accomplished from a console-type device. Refer to Section 2 for the procedure used to establish the teletypewriter emulation mode.

When all procedures requiring the teletypewriter emulation capability have been completed, the host computer should initiate the startup procedure by sending the single ASCII character GS (group separator; octal code 035) to the GRAPHIC 7. This character causes the GRAPHIC 7 to exit from the teletypewriter emulation mode and respond as if an IZ message had been sent from the host computer (an IZ message would not be recognized when the GRAPHIC 7 is in the teletypewriter emulation mode). The resulting initialization XX message to the host computer then completes the startup procedure.

NOTE

Exit from the teletypewriter emulation mode, initialization in the system mode, and sending of the XX message can also be accomplished by typing function key F13 (this causes octal code 035 to be generated). However, this operation would not be synchronized with normal host computer operations and might result in an improper startup sequence.

6.3 REFRESH FILES

The following paragraphs describe the generation, transmission, and alteration of refresh files to be processed by the graphic controller. Table 6-1 is an example of a simple refresh file that is used to illustrate various parts of the discussions. Figure 6-1 shows the display that results when the refresh file in table 6-1 is processed.

6.3.1 REFRESH FILE GENERATION

After startup procedures have been completed, the host computer application program must generate a refresh file to send to the GRAPHIC 7 so that the desired image can be displayed. The refresh file may be included in the application program itself or may be generated dynamically by the application program.

Table 6-1 is the listing for a simple refresh file that could be generated for display by the GRAPHIC 7. The first part of the file (before the label RLOOP) initializes parameters as required to ensure the proper interpretation of the instructions that follow. Then, at the beginning of the refresh loop, the CRT beam is moved to the center of the screen to wait for the frame sync pulse. Centering the beam in this manner is strongly recommended to improve the reliability of the display indicator by reducing power consumption (no deflection currents are supplied to the yoke during the wait period).

Following the WATE instruction is the sequence of instructions used to draw the two large squares and the four vectors that intersect in the center of the screen. The next instructions establish a scratchpad area by inserting ten spaces following the word "INPUT:" at the lower left of the screen. Note that the size and spacing of these characters was established by the initialization instructions at the beginning of the listing. The actual scratchpad is defined by the memory locations in which the spaces are located.

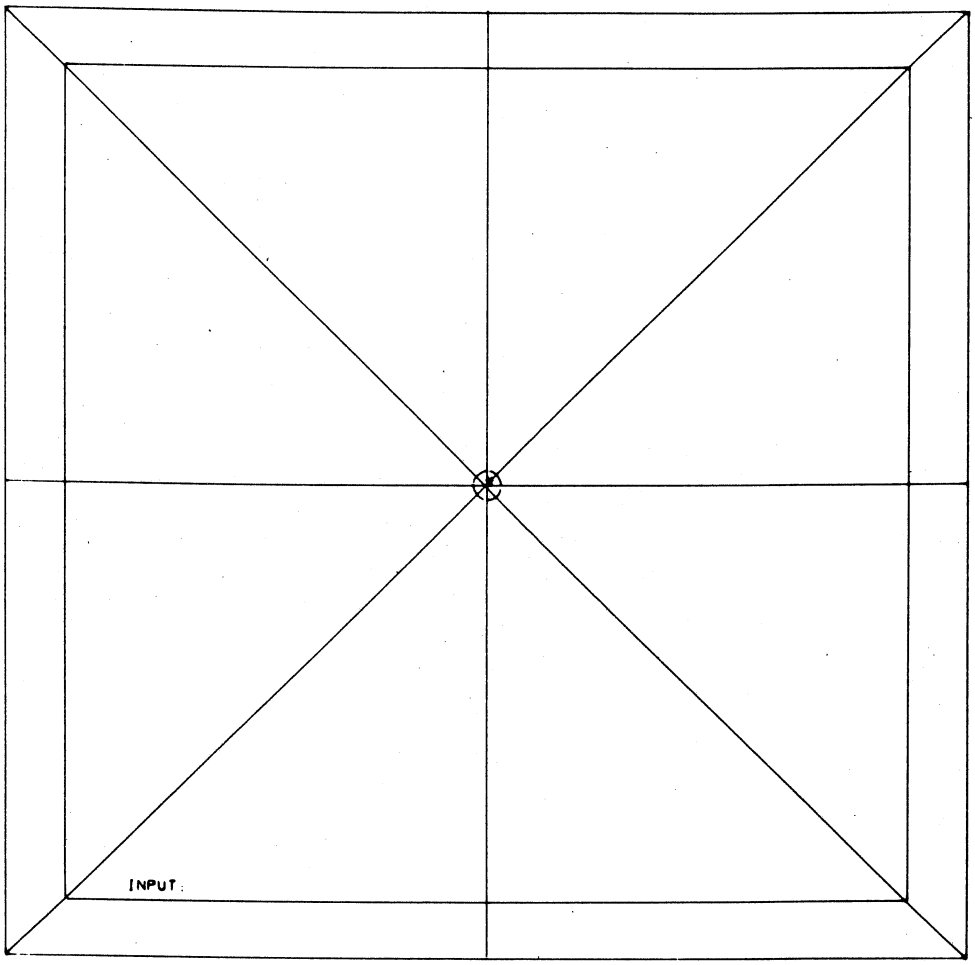
The four instructions following the scratchpad instructions define the small circle that is drawn at the center of the display. This circle is used to illustrate PED operation. Note that, if a conic generator card is not installed in the terminal controller, a small square is drawn in place of the small circle.

Finally, an asterisk (which is used to illustrate PHOTOPEN operation) is placed at the center of the display and the refresh file is terminated with a JUMP instruction. The JUMP instruction causes the graphic controller to loop back to the point

TABLE 6-1

SAMPLE REFRESH FILE NO. 1

```
00100                          ;
00200                          ; SET DISPLAY PARAMETERS
00300                          ;
00400    002000 013007                 LDDZ <CRT1,BLOFF,LINE,BR7 >
00500    002002 016330                 LDDP <F60, PP1, NOROTATE, CS0, FAST>
00600    002004 140112                 LDTI 12
00700    002006                RLOOP:
00800    002006 003000                 IZPR                    ;INITIALIZE
00900    002010 020000                 LDXA 0                  ;WAIT WITH BEAM AT 0,0
01000    002012 060000                 MVYA 0
01100    002014 007000                 WATE
01200                          ;
01300                          ; DRAW DIAGONALS
01400                          ;
01500    002016 020777                 LDXA 777                ;MOVE TO
01600    002020 060777                 MVYA 777                ; UPPER RIGHT
01700    002022 023000                 LDXA -1000              ;DRAW DIAGONAL
01800    002024 043000                 DRYA -1000              ; TO LOWER LEFT
01900    002026 050777                 MVXA 777                ;MOVE TO LOWER RIGHT
02000    002030 023000                 LDXA -1000              ;DRAW DIAGONAL
02100    002032 040777                 DRYA 777                ; TO UPPER LEFT
02200    002034 050000                 MVXA 0                  ;MOVE TO TOP CENTER
02300    002036 043000                 DRYA -1000              ;DRAW STRAIGHT DOWN
02400    002040 020777                 LDXA 777                ;MOVE TO FAR
02500    002042 060000                 MVYA 0                  ; RIGHT CENTER
02600    002044 033000                 DRXA -1000              ;DRAW HORIZONTAL TO LEFT
02700                          ;
02800                          ; DRAW OUTSIDE SQUARE
02900                          ;
03000    002046 063000                 MVYA -1000              ;MOVE TO BOTTOM LEFT
03100    002050 030777                 DRXA 777                ;DRAW STRAIGHT UP
03200    002052 040777                 DRYA 777                ;DRAW TOP EDGE TO RIGHT
03300    002054 033000                 DRXA -1000              ;DRAW RIGHT EDGE DOWN
03400    002056 043000                 DRYA -1000              ;DRAW BOTTOM EDGE TO LEFT
03500                          ;
03600                          ; DRAW INNER SQUARE
03700                          ;
03800    002060 023076                 LDXA -702               ;SET INSIDE POINT
03900    002062 063076                 MVYA -702               ; AT LOWER LEFT
04000    002064 045604                 DRYR 1604               ;DRAW STRAIGHT UP
04100    002066 035604                 DRXR 1604               ;DRAW TOP EDGE TO RIGHT
04200    002070 046174                 DRYR -1604              ;DRAW STRAIGHT DOWN
04300    002072 036174                 DRXR -1604              ;DRAW BOTTOM EDGE TO LEFT
04400                          ;
04500                          ; SCRATCHPAD PROMPTER
04600                          ;
04700    002074 023200                 LDXA -600               ;POSITION INSIDE INNER
04800    002076 063110                 MVYA -670               ; SQUARE AT LOWER LEFT
04900    002100 147311                 TXT I,N                 ;INSERT 'INPUT:'
05000    002102 152720                 TXT P,U
05100    002104 135324                 TXT T,:
05200                          ;
05300                          ; SCRATCHPAD
05400                          ;
05500    002106                SCRPAD:
05600    002106 120240                 TXT < >,< >             ;TEN
05700    002110 120240                 TXT < >,< >             ; SPACES
05800    002112 120240                 TXT < >,< >             ;  FOR
05900    002114 120240                 TXT < >,< >             ;   SCRATCHPAD
06000    002116 120240                 TXT < >,< >             ;    AREA
06100                          ;
06200                          ; PED CONTROLLED MOVING CIRCLE
06300                          ;
06400    002120                MCRCLE:
06500    002120 020000                 LDXA 0                  ;PED CONTROLLED
06600    002122 060000                 MVYA 0                  ; POSITION OF
06700    002124 073020                 LDKX 3,20               ;  SMALL
06800    002126 077020                 DRKY 3,20               ;   CIRCLE
06900                          ;
07000                          ; PHOTOPEN CONTROLLED ASTERISK
07100                          ;
07200    002130                PHPEN:
07300    002130 020000                 LDXA 0                  ;COORDINATES FOR
07400    002132 060000                 MVYA 0                  ; LAST PHOTOPEN STRIKE
07500    002134 117652                 CHAR *                  ;PUT AN ASTERISK THERE
07600    002136 001000                 JUMP RLOOP              ;REPEAT
07700    002140 002006
```



GA-77-419-07

Figure 6-1 Display Created by Sample Refresh File No. 1.

in the file labeled RLOOP and reprocess the entire file except for the parameter initialization instructions.

Thus table 6-1 represents a complete refresh file that can be processed by the graphic controller. The listing specifies that 49 words are required and that they are to be loaded into read/write memory beginning at octal location 2000 and ending at octal location 2140. Figure 6-1 illustrates the display that is created when this refresh file is processed by the graphic controller.

6.3.2 REFRESH FILE TRANSMISSION

After a refresh file has been generated, it must be transmitted from the host computer to the GRAPHIC 7 using a GCP+ message. Following transmission of the file, another GCP+ message must be sent to the host computer to start processing of the file. For the example refresh file shown in table 6-1, the following sequence of GCP+ messages would be used (it is assumed that an IZ message from the host computer has previously been sent to initialize the GRAPHIC 7):

a. Host-to-GRAPHIC 7 MU (memory update) message:

046524 - MU command header
002000 - load address for first data word
000061 - number of data words to be loaded
013007 - first data word to be loaded
016330 - second data word to be loaded
.
.
.
002006 - last data word to be loaded

b. Host-to-GRAPHIC 7 SP (start picture) message:

051520 - SP command header
002000 - starting address of refresh file

After this message is sent, a display similar to that illustrated in figure 6-1 appears on display indicator no. 1.

NOTE

For purposes of development or debugging, refresh files may also be loaded into read/write memory manually or from paper tape. These methods employ local mode commands for the GRAPHIC 7 as described in Section 2.

6.3.3 REFRESH FILE ALTERATION

After a refresh file has been loaded into read/write memory, it can be altered by the application program of the host computer using various GCP+ messages. Suppose for example, it is desired that the word "READY" be displayed in the scratchpad area for a specific period of time, after which spaces will be reinserted into the display. This can be accomplished by using GCP+ messages as follows:

a. Host-to-GRAPHIC 7 IS (enabled selected interrupts) message:

   044523 - IS command header
   000002 - enable halt interrupt

   This message enables the graphic controller to interrupt the display processor whenever the graphic controller executes a HALT instruction.

b. Host-to-GRAPHIC 7 SU (selective update) message:

   051525 - SU command header
   002106 - load address (beginning of scratchpad)
   000004 - number of data words to be loaded
   142722 - first data word (text "RE")
   142301 - second data word (text "AD")
   120331 - third data word (text "Y")
   000000 - fourth data word (HREF)

   After the refresh file has been altered in accordance with this message, "READY" is displayed in the scratchpad area each time the file is processed. Immediately after displaying "READY", the graphic controller halts and interrupts the display processor. This causes GCP+ to send a HI message to the host computer.

c. GRAPHIC 7-to-host HI (halt interrupt) message:

044111 - HI command header
002116 - contents of graphic controller program counter
120240 - contents of graphic controller instruction register
000000 - filler

This message is sent to the host computer each time the HREF instruction is executed by the graphic controller. Knowing that the refresh file is processed at the rate of 60 Hz (determined by the LDDP instruction at address 2002), the host application program can count the number of HI messages received to determine how long "READY" has been displayed in the scratchpad area. After the desired time has elapsed, a new MU message can be sent to restore spaces to the scratchpad addresses.

d. Host-to-GRAPHIC 7 KP (continue picture) message:

045520 - KP command header

Each time an HI message is received from the GRAPHIC 7, the host computer must respond with a KP message to restart the graphic controller or the CRT screen will remain blanked.

e. Host-to-GRAPHIC 7 MU (memory update) messages:

046524 - MU command header
002106 - load address (beginning of scratchpad)
000004 - number of data words to be loaded
120240 - first data word (text " ")
120240 - second data word (text " ")
120240 - third data word (text " ")
120240 - fourth data word (text " ")

This message is sent after "READY" has been displayed in the scratchpad area for the desired period of time. Altering the refresh file in accordance with this message restores the file to its original content and format.

f. Host-to-GRAPHIC 7 KP (continue picture) message:

045520 - KP command header

Following restoration of the refresh file to its original content, the host computer should send a KP message to the GRAPHIC 7.

6.4 OPTIONAL EQUIPMENT USAGE

Optional GRAPHIC 7 equipment includes keyboards, PEDs, PHOTOPENs, and a hard copy unit. At the time the GRAPHIC 7 is initialized in the system mode, keyboards and PEDs are automatically enabled. Following initialization, these devices and any PHOTOPENs that are used are enabled and disabled as required by IK (interrupt control) messages from the host computer to the GRAPHIC 7. The hard copy unit is controlled by sending an HY message from the host computer to the GRAPHIC 7. The following paragraphs provide examples of how GCP+ is used to control each type of optional equipment (unless otherwise stated, the examples are assumed to refer to alphanumeric keyboard no. 1, PED no. 1, and PHOTOPEN no. 1).

6.4.1 KEYBOARDS

After a keyboard has been enabled, GCP+ sends each character to the host computer in a KY (alphanumeric keyboard no. 1) message. For the refresh file listed in table 6-1, this feature might be used by the host computer to collect characters for the scratchpad area on an individual basis. The host computer could then use SU (selective update) messages to echo each character as it is typed by the operator.

Such operations have the advantage that the host computer can maintain complete control over what is displayed in the scratchpad area and can perform any editing or character conversion routines that may be required (e.g., lower case to upper case). For simple applications, however, the scratchpad feature available in GCP+ can be used to relieve the host computer of many processing tasks. As an example, assume that the scratchpad feature is to be used for the scratchpad area defined in the table 6-1 refresh file. The scratchpad mode of operation would be established by the following host-to-GRAPHIC 7 ZR (initialize scratchpad for alphanumeric keyboard no. 1) message:

055122 - ZR command header
002106 - starting address (first address of scratchpad)
000012 - number of characters in line (ten)

Once this message has been received by the GRAPHIC 7, GCP+ enables the keyboard and enters the scratchpad mode of processing keyboard inputs. The host computer is then free to proceed to other tasks as necessary while GCP+ collects keyboard inputs in the scratchpad area. GCP+ collects characters in the scratchpad area and echos them on the display completely independently of any operations being performed by the host computer. The RUB OUT key can also be used, if required, to

delete any erroneous entries that may be made. When the operator is through typing characters into the scratchpad, he types RETURN at which time communications are reestablished with the host computer by means of the following GRAPHIC 7-to-host XR (scratchpad ready for alphanumeric keyboard no. 1) message:

054122 - XR command header
XXXXXX - number of characters in the scratchpad
000000 - filler
000000 - filler

After receiving the XR message, the host computer responds with the following host-to-GRAPHIC 7 message:

043511 - GI command header
002106 - starting address (first address of scratchpad)
000005 - number of words requested

The GI message would, in turn, cause GCP+ to respond with the following two messages to the host computer:

NOTE

For large scratchpad sizes, the number of integer words requested for the GI message could be calculated as follows:

number of words = (number of characters in the scratchpad +1)/2

a. GRAPHIC 7-to-host RI (return image) message:

051111 - RI command header
002106 - starting address (first address of scratchpad)
000005 - number of words to be transferred
000000 - filler

b. GRAPHIC 7-to-host VL (variable length) message:

053114 - VL command header
000005 - number of words to be transferred
147723 - first data word (text "SO")
140640 - second data word (text " A")
120315 - third data word (text "M ")
120311 - fourth data word (text "I ")
120240 - fifth data word (text " ")

This message indicates that the operator typed "SO AM I" into the scratchpad and then typed RETURN.

After the requested data has been returned to the host computer in a VL message, the host computer sends the following message to the GRAPHIC 7 to clear the scratchpad area:

Host-to-GRAPHIC 7 ZS message:

055123 - ZS command header

This message causes GCP+ to space fill the whole scratchpad area and reposition the scratchpad pointer to the beginning of the scratchpad area.

6.4.2 PEDs

There are four modes of operation that can be established for PEDs. These are the automatic track mode (mode Ø), the automatic mode (mode 1), the request mode (mode 2), and the tracking mode (mode 3). The desired operating mode is established by a host-to-GRAPHIC 7 IP (initialize PED no. 1) message, a detailed discussion of which is contained in paragraph 5.3.6. In the following paragraphs, the small circle in figure 6-1 is used as an example of a PED-controlled display element.

Mode Ø is applicable only to data tablet type PEDs and is not discussed in this example. Mode 1 is applicable only to trackball/forcestick type PEDs. Modes 2 and 3 are applicable to all types of PEDs. Uses of modes 1, 2, and 3 are described in the following paragraphs.

When mode 1 is used, RP (return PED no. 1) messages are sent automatically from the GRAPHIC 7 to the host computer to indicate changes in the relative position of the PED. The host computer then processes this data and, whenever required by the application program, sends an SU (selective update) message to update the instructions that define the center of the circle (these are the LDXA and MVYA instructions at addresses 2120 and 2122 respectively).

When mode 2 is used, absolute PED position coordinates are maintained at all times by GCP+ but the refresh file is not altered and the data is not sent to the host computer. In this mode, if the host computer application desires to know the position of the PED, a GP (give PED no. 1) message must be sent to the GRAPHIC 7. GCP+ responds by returning the latest absolute PED position data to the host computer in an RP (return PED no. 1) message. If desired, the host computer can then send the data back to the GRAPHIC 7 in an SU message to update the instructions that define the center of the circle.

When mode 3 is used, the position of the circle can be controlled by manipulating the PED completely independently of the host computer. This mode would be established for the circle by the following host-to-GRAPHIC 7 IP message:

044520 - IP command header
000003 - establish PED operating mode 3
002120 - address of LDXA instruction (that defines circle center)

After mode 3 operation has been established for the PED, GCP+ automatically updates the instructions in the refresh file that define the center of the circle and the circle follows PED motions without any further action on the part of the host computer.

NOTE

Whenever an IP message is sent from the host computer to the GRAPHIC 7, GCP+ automatically enables the interrupt associated with the PED.

6.4.3 PHOTOPENs

To illustrate PHOTOPEN operation, it is assumed that the host computer application program is written to move the asterisk in figure 6-1 to the location of the latest PHOTOPEN strike. Then, if the PHOTOPEN switch is pressed, the position of the asterisk will no longer change. When the PHOTOPEN switch is pressed again,

moving the asterisk to the location of the latest PHOTOPEN strike will resume. The process begins with the following host-to-GRAPHIC 7 IK (interrupt control) message to enable the PHOTOPEN interrupts:

044513 - IK command header

XXX06X - enable interrupts caused by PHOTOPEN no. 1 strikes or switch closures, disable PHOTOPEN item number interrupts (status of other interrupts is not specified)

NOTE

When the IK message is sent, all interrupts are reestablished. If 0's are stored in the X bits of the interrupt word, all these types of interrupts are disabled.

After this message has been sent to the GRAPHIC 7, GCP+ causes a PN (PHOTOPEN no. 1 strike) message to be sent to the host computer when light is detected by the PHOTOPEN. The PN message contains the contents of the graphic controller program counter (the address of the instruction being executed at the time of the strike plus four bytes) and the contents of the X and Y position registers (when a PHOTOPEN strike is caused by a vector, the X and Y position registers contain the coordinates of the end point of the vector). For instance, if the PHOTOPEN is pointed at any point along the right hand vertical vector of the inner square, GCP+ sends the following PN message to the host computer:

050116 - PN command header

002074 - contents of graphic controller program counter

000702 - contents of X position register

007076 - contents of Y position register

This message indicates that a PHOTOPEN strike occurred when the graphic controller was executing the instruction at memory address 2070 and, at that time, the contents of the X and Y position registers were $+702_8$ and -702_8, respectively (the bottom point of the vector pointed to by the PHOTOPEN).

When GCP+ sends the PN message to the host computer, it also disables PHOTOPEN strikes. This is done to prevent GCP+ from overloading the host computer by sending 60 PN messages to host computer every second. (If the refresh rate were 60 Hz and the operator pointed the PHOTOPEN at the selected vector for 3 seconds, then 180 identical PN messages would be send to the host computer if PHOTOPEN strikes were not disabled after the first PHOTOPEN strike is detected.)

TABLE 6-2

SAMPLE REFRESH FILE NO. 2

```
00100                    ;
00200                    ; SET DISPLAY PARAMETERS
00300                    ;
00400   002000 016370             LDDP <F60, ALLPP, NOROTATE, CSO, FAST>
00500   002002 140112             LDTI 12
00600   002004           RLOOP:
00700   002004 003000             IZPR                        ;INITIALIZE
00800   002006 013407             LDDZ <CRT1, CRT2, BLOFF, LINE, BR7>
00900   002010 020000             LDXA 0                      ;WAIT WITH BEAM AT 0,0
01000   002012 060000             MVYA 0
01100   002014 007000             WATE
01200                    ;
01300                    ; DRAW DIAGONALS
01400                    ;
01500   002016 020777             LDXA 777                    ;MOVE TO
01600   002020 060777             MVYA 777                    ; UPPER RIGHT
01700   002022 023000             LDXA -1000                  ;DRAW DIAGONAL
01800   002024 043000             DRYA -1000                  ; TO LOWER LEFT
01900   002026 050777             MVXA 777                    ;MOVE TO LOWER RIGHT
02000   002030 023000             LDXA -1000                  ;DRAW DIAGONAL
02100   002032 040777             DRYA 777                    ; TO UPPER LEFT
02200   002034 050000             MVXA 0                      ;MOVE TO TOP CENTER
02300   002036 043000             DRYA -1000                  ;DRAW STRAIGHT DOWN
02400   002040 020777             LDXA 777                    ;MOVE TO FAR
02500   002042 060000             MVYA 0                      ; RIGHT CENTER
02600   002044 033000             DRXA -1000                  ;DRAW HORIZONTAL TO LEFT
02700                    ;
02800                    ; DRAW INSIDE SQUARE
02900                    ;
03000   002046 063000             MVYA -1000                  ;MOVE TO BOTTOM LEFT
03100   002050 030777             DRXA 777                    ;DRAW STRAIGHT UP
03200   002052 040777             DRYA 777                    ;DRAW TOP EDGE TO RIGHT
03300   002054 033000             DRXA -1000                  ;DRAW RIGHT EDGE DOWN
03400   002056 043000             DRYA -1000                  ;DRAW BOTTOM EDGE TO LEFT
03500                    ;
03600                    ; DRAW INNER SQUARE
03700                    ;
03800   002060 023076             LDXA -702                   ;SET INSIDE POINT
03900   002062 063076             MVYA -702                   ; AT LOWER LEFT
04000   002064 045604             DRYR 1604                   ;DRAW STRAIGHT UP
04100   002066 035604             DRXR 1604                   ;DRAW TOP EDGE TO RIGHT
04200   002070 046174             DRYR -1604                  ;DRAW STRAIGHT DOWN
04300   002072 036174             DRXR -1604                  ;DRAW BOTTOM EDGE TO LEFT
04400                    ;
04500                    ; SCRATCHPAD PROMPTER #1
04600                    ;
04700   002074 013007             LDDZ <CRT1>                 ;SELECT DISPLAY #1
04800   002076 023200             LDXA -600                   ;POSITION INSIDE INNER
04900   002100 063110             MVYA -670                   ; SQUARE AT LOWER LEFT
05000   002102 147311             TXT I,N                     ;INSERT "INPUT:"
05100   002104 152720             TXT P,U
05200   002106 135324             TXT T,:
05300                    ;
05400                    ; SCRATCHPAD FOR DISPLAY #1
05500                    ;
05600   002110           SCRPD1:
05700   002110 120240             TXT < >,< >                 ;TEN
05800   002112 120240             TXT < >,< >                 ; SPACES
05900   002114 120240             TXT < >,< >                 ;  FOR
06000   002116 120240             TXT < >,< >                 ;   SCRATCHPAD
06100   002120 120240             TXT < >,< >                 ;    AREA
06200                    ;
06300                    ; PHOTOPEN CONTROLLED ASTERISK #1
06400                    ;
06500   002122           PHPN1:
06600   002122 020000             LDXA 0                      ;COORDINATES FOR
06700   002124 060000             MVYA 0                      ; LAST PHOTOPEN #1 STRIKE
06800   002126 117652             CHAR *                      ;PUT AN ASTERISK THERE
```

TABLE 6-2

SAMPLE REFRESH FILE NO. 2 (Cont)

```
06900                    ;
07000                    ; MOVING CIRCLE #1
07100                    ;
07200   002130           MCRCL1:
07300   002130 020000            LDXA 0                  ;PED CONTROLLED
07400   002132 060000            MVYA 0                  ; POSITION OF
07500   002134 073020            LDKX 3,20               ;  SMALL
07600   002136 077020            DRKY 3,20               ;   CIRCLE
07700                    ;
07800                    ; SCRATCHPAD PROMPTER #2
07900                    ;
08000   002140 012407            LDDZ <CRT2>             ;SELECT DISPLAY #2
08100   002142 023200            LDXA -600               ;POSITION INSIDE INNER
08200   002144 063110            MVYA -670               ; SQUARE AT LOWER LEFT
08300   002146 147311            TXT I,N                 ;INSERT "INPUT:"
08400   002150 152720            TXT P,U
08500   002152 135324            TXT T,:
08600                    ;
08700                    ; SCRATCHPAD FOR DISPLAY #2
08800                    ;
08900   002154           SCRPD2:
09000   002154 120240            TXT < >,< >             ;TEN
09100   002156 120240            TXT < >,< >             ; SPACES
09200   002160 120240            TXT < >,< >             ;  FOR
09300   002162 120240            TXT < >,< >             ;   SCRATCHPAD
09400   002164 120240            TXT < >,< >             ;    AREA
09500                    ;
09600                    ; PHOTOPEN CONTROLLED ASTERISK #2
09700                    ;
09800   002166           PHPN2:
09900   002166 020000            LDXA 0                  ;COORDINATES FOR
10000   002170 060000            MVYA 0                  ; LAST PHOTOPEN #2 STRIKE
10100   002172 117652            CHAR *                  ;PUT AN ASTERISK THERE
10200                    ;
10300                    ; PED CONTROLLED MOVING CIRCLE #2
10400                    ;
10500   002174           MCRCL2:
10600   002174 020000            LDXA 0                  ;PED CONTROLLED
10700   002176 060000            MVYA 0                  ; POSITION OF
10800   002200 073020            LDKX 3,20               ;  SMALL
10900   002202 077020            DRKY 3,20               ;   CIRCLE
11000   002204 001000            JUMP RLOOP              ;REPEAT
11100   002206 002004
```

It is also possible for a refresh file to contain entirely different images for each display indicator. Normally, such a file would contain a main program loop that calls two subroutines, each subroutine being the instructions associated with a particular display indicator. The following is an example of how such a refresh file might be structured:

```
MAIN:
    LDXA 0                      ;CENTER BEAMS OF ALL
    MVYA 0                      ;  DISPLAY INDICATORS
    WATE                        ;WAIT FOR FRAME SYNC
    CALL RFRS1                  ;DRAW IMAGE ON DISPLAY INDICATOR NO. 1
    CALL RFRS2                  ;DRAW IMAGE ON DISPLAY INDICATOR NO. 2
    JUMP MAIN                   ;LOOP BACK TO BEGINNING

RFRS1:
    IZPR                        ;INITIALIZE
    LDDZ <CRT1,.....ARGn>       ;SELECT DISPLAY INDICATOR NO. 1
    LDDP <ARG1,.....ARGn>       ;  AND ESTABLISH
    LDTI nn                     ;    DESIRED PARAMETERS
     .                          ;INSTRUCTIONS
     .                          ;  FOR
     .                          ;    DRAWING
     .                          ;      IMAGE ON
     .                          ;        DISPLAY
     .                          ;          INDICATOR NO. 1
    RTRN                        ;RETURN TO MAIN LOOP

RFRS2:
    IZPR                        ;INITIALIZE
    LDDZ <CRT2,.....ARGn>       ;SELECT DISPLAY INDICATOR NO. 2
    LDDP <ARG1,.....ARGn>       ;  AND ESTABLISH
    LDTI nn                     ;    DESIRED PARAMETERS
     .                          ;INSTRUCTIONS
     .                          ;  FOR
     .                          ;    DRAWING
     .                          ;      IMAGE ON
     .                          ;        DISPLAY
     .                          ;          INDICATOR NO. 2
    RTRN                        ;RETURN TO MAIN LOOP
```

NOTE

In the example presented, some additional software considerations should be taken into account if it is desired to generate hard copies of each display presentation. The LDDZ instruction performs several functions in addition to enabling the Z-axes for each display indicator. (Refer to paragraph 2.3.4 for a description of the LDDZ instruction.) In the example, the refresh instructions contained in the body of code designated as instructions for drawing image on display indicator no. 1 or no. 2 could contain several LDDZ instructions. To simplify hardcopy generation, all LDDZ instructions, contained in the body of code mentioned previously, should be set up so that the display change enable bit (bit 10) is set to Ø. When these LDDZ instructions are executed, the display selected remains unchanged (i.e., the display selected defaults to the display selected via the first LDDZ instruction contained in the refresh files allocated to each display indicator).

In the example, a hardcopy of display indicator no. 1 could be generated as follows:

1. Send an SU message to modify the first LDDZ in RFRS2 to select display indicator no. 2.

2. Send an SU message to modify the first LDDZ in RFRS1 to select display indicators no. 1 and no. 4.

3. Send an HY message to generate a hardcopy for display indicator no. 1.

A hardcopy of display indicator no. 2 could be generated as follows:

1. Send an SU message to modify the first LDDZ in RFRS1 to select display indicator no. 1.

2. Send an SU message to modify the first LLDZ in RFRS2 to select display indicators no. 2 and no. 4.

3. Send an HY message to generate a hardcopy for display indicator no. 2.

SECTION 7
ADVANCED GRAPHIC CONTROL PROGRAM USAGE

7.1 INTRODUCTION

For certain applications, the GRAPHIC 7 may be required to operate in a stand alone mode or to provide processing capabilities beyond those available in the standard graphic control program. As previously discussed, the display processor and the graphic controller can operate independently, each executing a separate program located in the GRAPHIC 7 memory. It is also possible for the two microprocessors to interact by using the LINK instruction to synchronize the operations. Programs to be executed by the display processor are loaded into the read/write memory of the GRAPHIC 7 in the same manner that refresh files to be processed by the graphic controller are loaded (refer to the descriptions of the host-to-GRAPHIC 7 MU and TK messages in paragraph 5.3.3).

This section describes the manner in which special instructions and programming techniques can be used to expand the processing capabilities of the GRAPHIC 7. The discussions assume that the reader is thoroughly familiar with the display processor instruction set (paragraph 3.2). It is also recommended that, before using any of the techniques described in this section, the user study a listing of the graphic control program enhanced (Sanders publication H-79-0356) and familiarize himself with the details of its operation.

7.2 RAM LINKAGES

There are three different linkages by which GCP+ can exit to a user-defined program in the GRAPHIC 7 memory. Each linkage can be enabled separately by placing an address, to which GCP+ should transfer control, into a specified memory location. These memory locations and the associated GCP+ exit points are as follows:

Memory Location	Associated GCP+ Exit Point
000710	Unknown command header sent by host computer
000712	Beginning of GCP+ executive loop
000714	Message ready to send to host computer

Normally, the contents of the linkage locations are zero. If, however, the user places a non-zero value in any of the locations, its contents will be interpreted by GCP+ as a subroutine address to which control of the display processor should be transferred. That is, a non-zero value in any linkage location will cause the following instruction to be performed by GCP+:

```
JSR   PC,address
```

7.2.1 UNKNOWN COMMAND HEADER SENT BY HOST COMPUTER

When GCP+ does not recognize a command header sent by the host computer, it checks location 710 for a possible linkage to a user program. If the contents of the location is non-zero, GCP+ loads the command header into register R0 of the display processor and performs a JSR PC instruction to the specified address.

At this point, GCP+ is operating within an interrupt handling process if communications with the host computer are being handled over a parallel interface. If communications with the host computer are being handled over a serial interface, GCP+ operates under simulated interrupt conditions. The basic difference is that the true interrupt process is non-interruptable, whereas the simulated interrupt process can be interrupted.

Regardless of the interface used, additional processing required by a user program should be completed as quickly as possible. GCP+ expects the user program either to recognize the command header and process it or to make an error return. If the command header is recognized and/or processed, the normal return is simply:

```
RTS   PC
```

If the command header is not recognized, the error return is:

```
ADD   #2,(SP)
RTS   PC
```

which eventually causes GCP+ to send an XX message to the host computer with bit 14 of word 1 set to one. Bit 14 indicates that the command header was not recognized by GCP+.

During the processing of a user-defined command header sent by the host computer, the following subroutines of GCP+ may be useful:

READ - reads additional data from the host computer over a parallel or serial interface (serial data is coded, 4 bytes per word, as described in paragraph 5.2.1)

READH - reads data in command header sent by the host computer

REQUST - requests a send buffer for returning data to the host computer

FULL - this subroutine should be called if REQUST indicates that no send buffer is available

Methods of employing these subroutines are as follows:

a. READ

1. Single word read:

```
CLR  R5          ;SET R5 = 0
JSR  PC,@READ    ;READ ONE WORD
                 ;WORD IS IN R0
```

NOTE

For a word read over the parallel interface, the word is placed directly in R0. For a word read over the serial interface, the word is decoded and then placed in R0.

2. DMA mode:

```
Set R5 = word count
Set R4 = start address
JSR  PC,@READ    ;READ R5 WORDS
                 ;INTO ADDRESS STARTING AT R4
```

NOTE

On return, the DMA is done.

b. READH

READH is always called by:

```
CLR  R5          ;SET R5 = 0
JSR  PC,@READH   ;GET COMMAND HEADER
                 ;COMMAND HEADER IS IN R0
```

NOTE

For a word read over the parallel interface, READH is the same as READ. For a word read over the serial interface, READH places the word, undecoded, in R0.

c. REQUST

The following sequence requests a send buffer:

```
JSR  PC,REQUST   ;BUFFER ADDRESS
TST  R3          ;  RETURNED IN R3
BNE  GOTIT       ;    UNLESS
 .               ;      R3 = 0 (NONE AVAILABLE)
 .
GOTIT:
```

NOTE

The send buffer is 4 words (8 bytes) long. The first word should consist of 2 alphanumeric (A-Z or 0-9) ASCII characters with the MSB (8th bit) of each set to one. Any desired information may be placed in the remaining three words. The action of REQUST is to queue the selected buffer so that GCP+ can eventually send its contents to the host computer.

d. FULL

If REQUST returns R3 = 0, FULL should be called as follows:

```
Set R3 = first word intended for send buffer
 JSR  PC,FULL
```

NOTE

Calling FULL causes GCP+ to send a buffer XX message to the host computer as described in paragraph 5.3.1.

7.2.2 BEGINNING OF GCP+ EXECUTIVE LOOP

GCP+ operates constantly in an interruptable executive loop that performs the following steps as shown in figure 7-1,

Any additional processes that the user may want to include in the GCP+ executive loop can be included by writing an appropriate subroutine and placing the starting address of the subroutine in linkage location 712. GCP+ then performs a JSR PC to the specified address as the first step in its executive routine. Note that the contents of registers R0 through R5 are meaningless at this point in the execution of the program.

The GCP+ subroutines listed in paragraph 7.2.1 may also be useful in user-defined programs to which GCP+ exists via linkage location 712. Two additional GCP+ subroutines that may be useful are:

ENBINT - enables interrupts on interface to host computer
DISINT - disables interrupts on interface to host computer

These subroutines are called by JSR PC,@ENBINT and JSR PC,@DISINT, respectively.

7.2.3 MESSAGE READY TO SEND TO HOST COMPUTER

As described in paragraph 7.2.2, GCP+ automatically checks linkage location 714 whenever a message is ready to be sent from the GRAPHIC 7 to the host computer. If the content of location 714 is zero, the message is sent to the host computer. If the content of location 714 is non-zero, GCP+ performs a JSR PC to the user-defined program at the specified address. One purpose of this linkage is to permit standard messages to be intercepted and, if necessary, modified before being sent to the host computer. A second purpose is to permit the data in certain messages to be processed locally by user-defined routines within the GRAPHIC 7 and thereby relieve the host computer of many of its processing tasks.

7.3 LINK INSTRUCTION

The graphic controller LINK instruction provides an efficient means of time-sharing the capabilities of the graphic controller and the display processor. Use of the LINK instruction permits:

- Peripheral and optional equipment to be slaved to requirements of the refresh file.

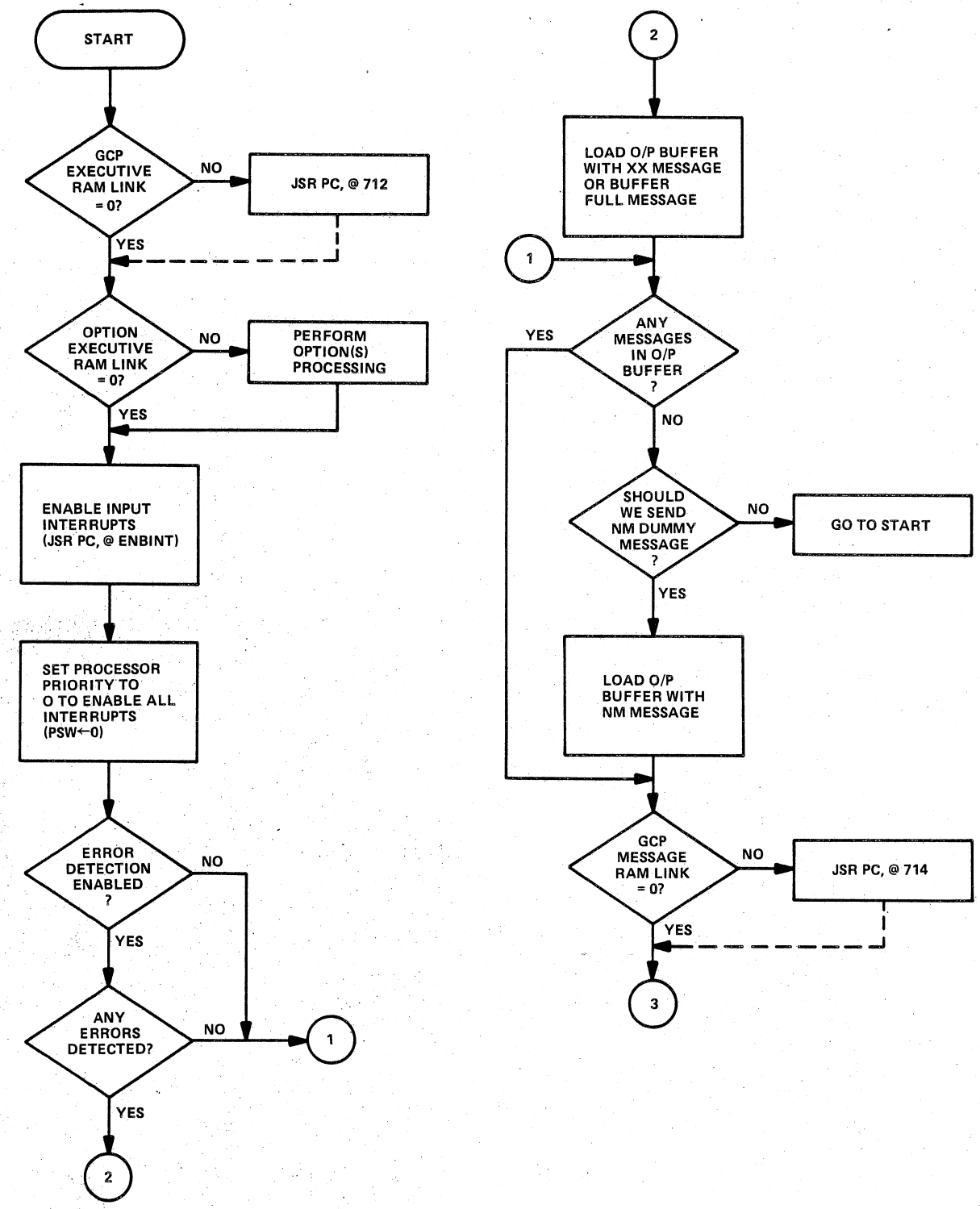


Figure 7-1 GCP+ Executive Loop Flowchart. (Sheet 1)

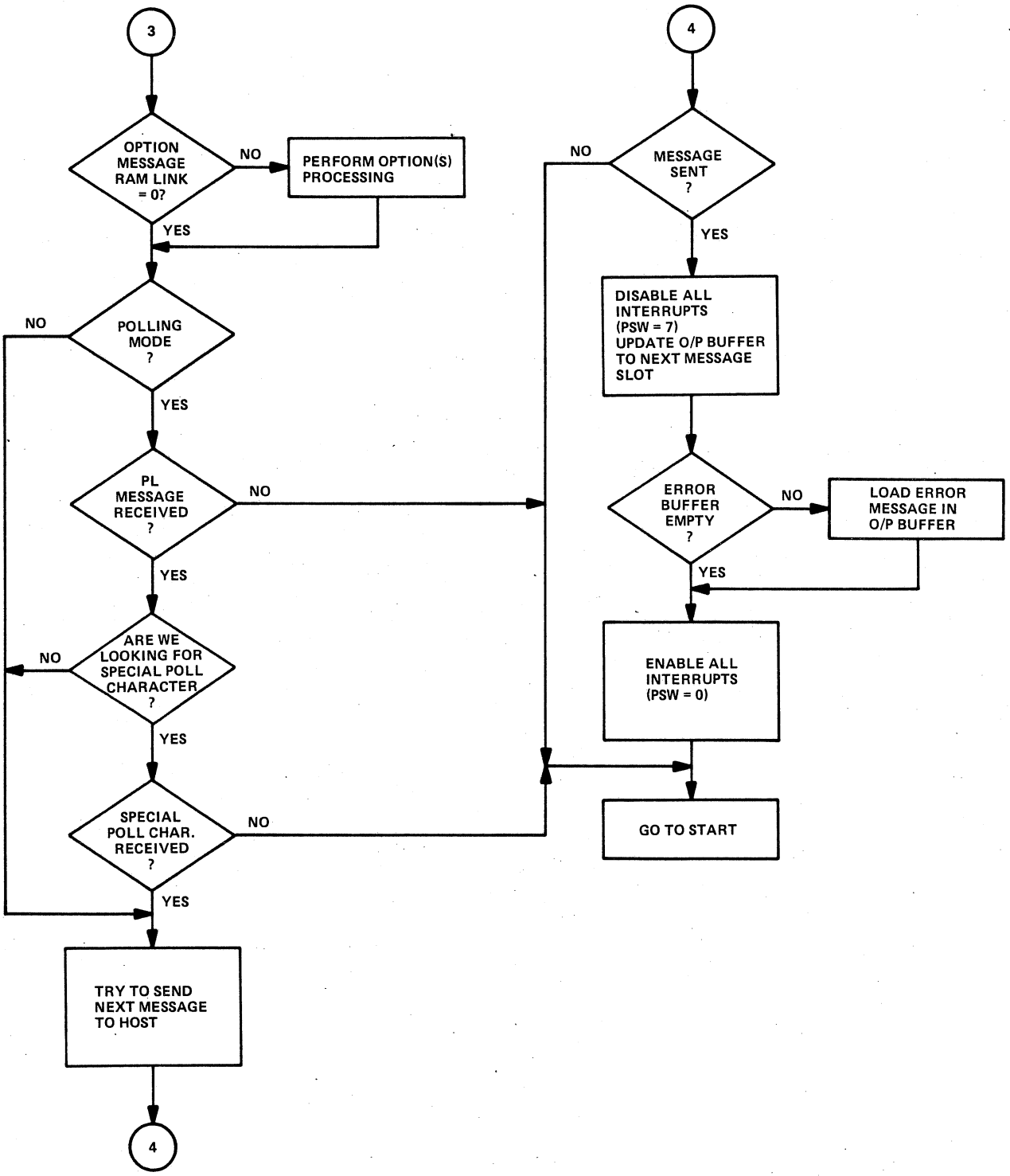


Figure 7-1 GCP+ Executive Loop Flowchart. (Sheet 2)

- Parallel processing to be accomplished by the graphic controller and the display processor without having to maintain a separate work copy of the refresh file and without harmful interference to the displayed image.

7.3.1 BASIC INSTRUCTION OPERATION

When the graphic controller encounters a LINK instruction in the refresh file, the following operations occur:

a. The graphic controller fetches the address portion of the LINK instruction and then performs a jump and mark to that address.

b. The graphic controller stops fetching words from the refresh file, halts, and interrupts the display processor.

c. The contents of all graphic controller registers are made available to the program being run by the display processor.

Several features of the display processor contribute to the efficiency of operations using the LINK instruction:

a. The display processor allows a unique trap vector to be assigned to the LINK interrupt (the hardwired trap address is 170). Therefore, a lengthy and time-consuming interrupt handler is not required to sort out the LINK interrupt from all other possible types of interrupts.

b. The display processor hardware automatically stores its program parameters when an interrupt occurs.

c. Almost all display processor instructions can be performed using references to memory locations instead of registers. This means that, for many routines, the contents of the general purpose registers need not be stored.

d. If the LINK interrupt routine requires the use of one or more general purpose registers, their contents can easily be saved on the interrupt push down stack and recalled upon completion of the interrupt routine.

e. Upon completion of the LINK interrupt routine, a single instruction (RTI) returns the display processor to the task it was performing at the time of the interrupt.

f. The interrupt nesting feature of the display processor (using the stack pointer) allows the LINK interrupt routine to be interrupted, if necessary, and returned to in an entirely transparent manner.

To restart the graphic controller after a LINK instruction has been executed, it is only necessary to write the desired starting address into the graphic controller program counter (DPC). The graphic controller automatically starts fetching refresh file instructions from that location. It is possible for a given LINK interrupt routine to have several exits, each specifying a different restart address as determined by decisions made in the interrupt routine.

There are three basic methods of using the LINK instruction. They are referred to as synchronized linkage, sync link, and super sync. Paragraphs 7.3.2 through 7.3.4 describe these methods in detail and give a coding example and a flow chart for each.

7.3.2 SYNCHRONIZED LINKAGE

Synchronized linkage is the straightforward method of using the LINK instruction. Figure 7-2 is an example of program coding using the synchronized linkage method. Figure 7-3 is a flow chart for the coding example. The following features of the synchronized linkage method of using the LINK instruction should be noted:

a. The interrupt routine can be used several times in a refresh file. This is possible because each calling routine automatically writes a unique LINK return address into memory to provide the required steering back to the routine that called it.

b. If two or more different image problems must be solved in the same refresh file, the synchronized linkage method usually requires the beginning portion of the LINK interrupt routine to contain an interrupt handler. This handler is used to identify which of the image problems is to be solved. Such identification can be based on the LINK return address, the contents of the graphic controller program counter, or the contents of other graphic controller registers.

c. If the refresh file and image problem-solving sequence is well ordered, an interrupt handler may not be required. Instead, the routine used to solve one image problem can load the trap address (location 170) with the starting address of the next LINK interrupt routine in the refresh file.

7.3.3 SYNC LINK

The sync link method improves the efficiency and ease of using the LINK instruction by means of a simple technique. When a sync link operation is performed, the LINK instruction causes the LINK return address to be placed in the LINK

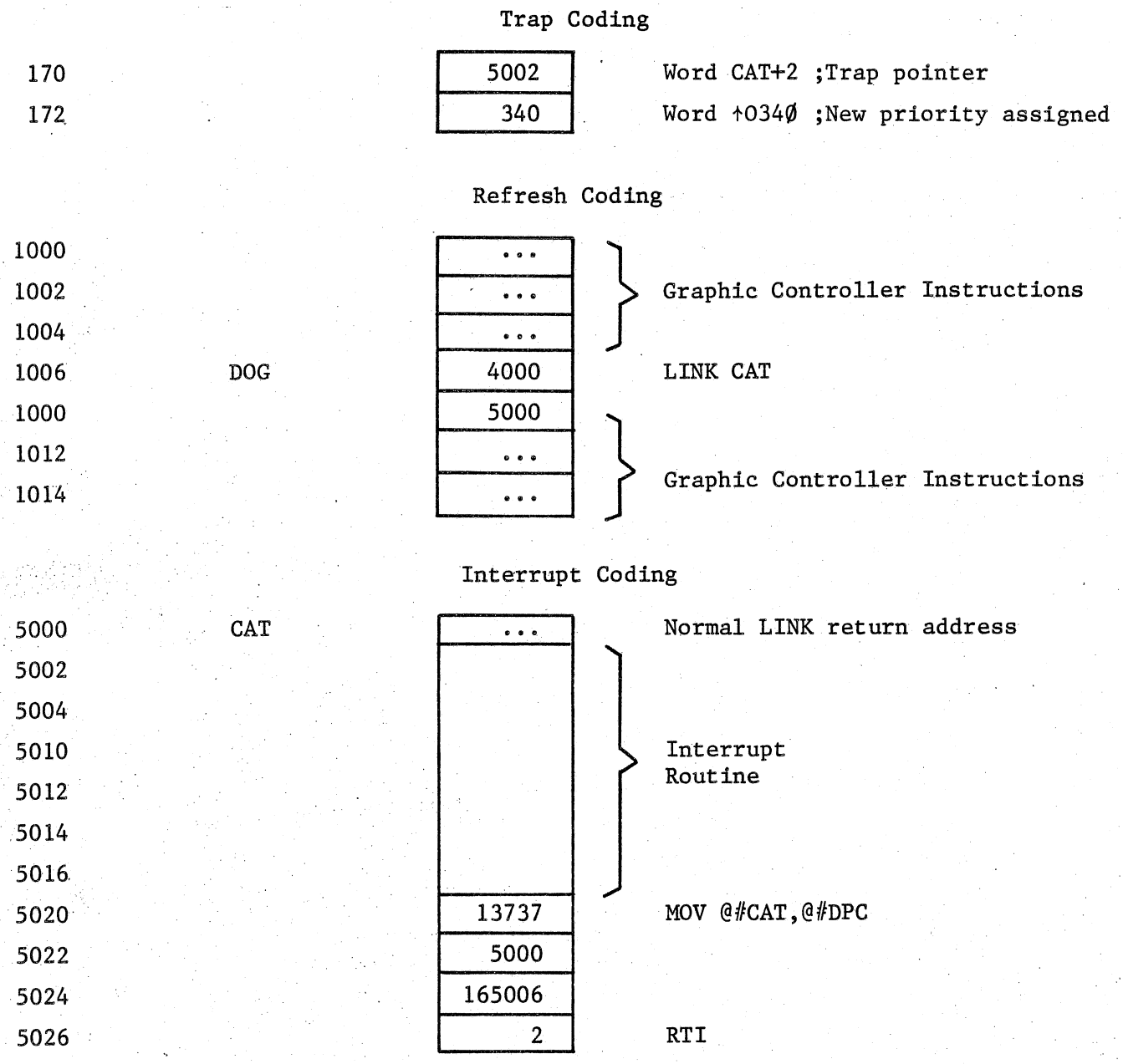


Figure 7-2 Synchronized Linkage Program Coding Example.

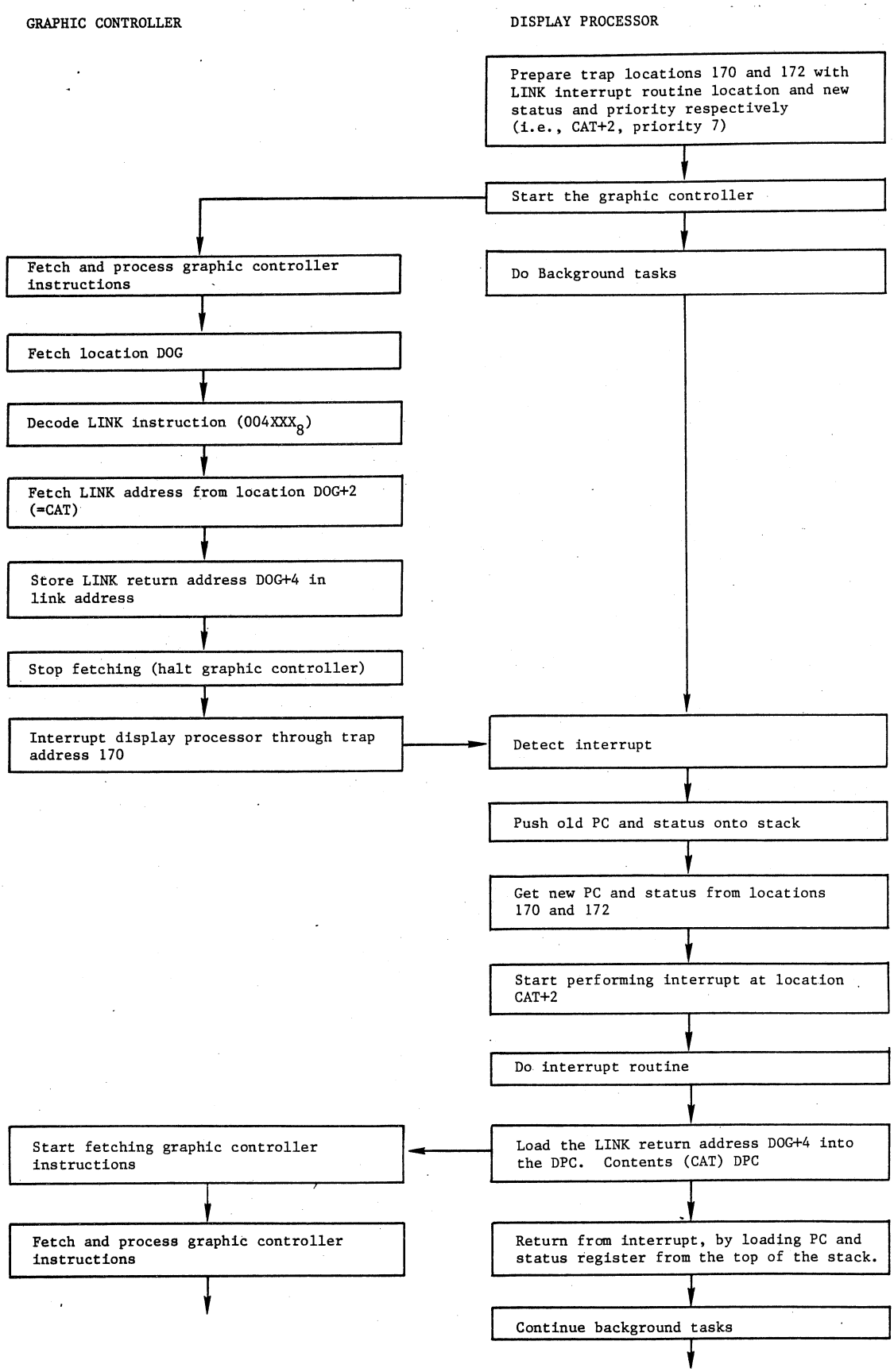


Figure 7-3 Synchronized Linkage Flow Chart Example.

trap address (location 170). This, in turn, causes the display processor to trap to the next instruction in the refresh file. Figure 7-4 is an example of program coding using the sync link method. Figure 7-5 is a flow chart for the coding example.

The result of using the sync link method is that display processor instructions can be placed directly in a refresh file. When the graphic controller encounters the LINK instruction, it halts while the interrupt routine (immediately following the LINK instruction in the refresh file) is processed by the display processor.

Return to processing of refresh file instructions by the graphic controller is accomplished by means of a relink command. The form of this command, which is shown in figure 7-4, never changes. Thus the assembler can generate the necessary coding with a single macro instruction.

The relink operation need not always be to the next sequential group of instructions in the refresh file. Several relinks can be provided for each LINK operation. Each relink can jump to any appropriate location in the refresh file as determined by decisions made in the interrupt routine. All that is necessary to jump anywhere in memory is to change the second word of the relink command to reflect the desired location.

Using the sync link method enables the capabilities of the graphic controller and the display processor to be time shared to solve a problem in a manner that is practically transparent to the programmer. Note that no interrupt handler is required if the sync link method is used for several LINK instructions in a refresh file. The sync link method uses the display processor interrupt trapping mechanism as an automatic interrupt handler to establish the required return paths.

7.3.4 SUPER SYNC

The main feature of the super sync method is that the LINK instruction identifies to the display processor a point in time that is synchronized with a location in the refresh file. This permits parallel processing by the display processor and the graphic controller that is synchronized with the displayed image. Figure 7-6 is an example of program coding using the super sync method. Figure 7-7 is a flow chart for the coding example.

An obvious advantage of using the super sync method is that the graphic controller is required to be halted for a minimum amount of time. Therefore, the

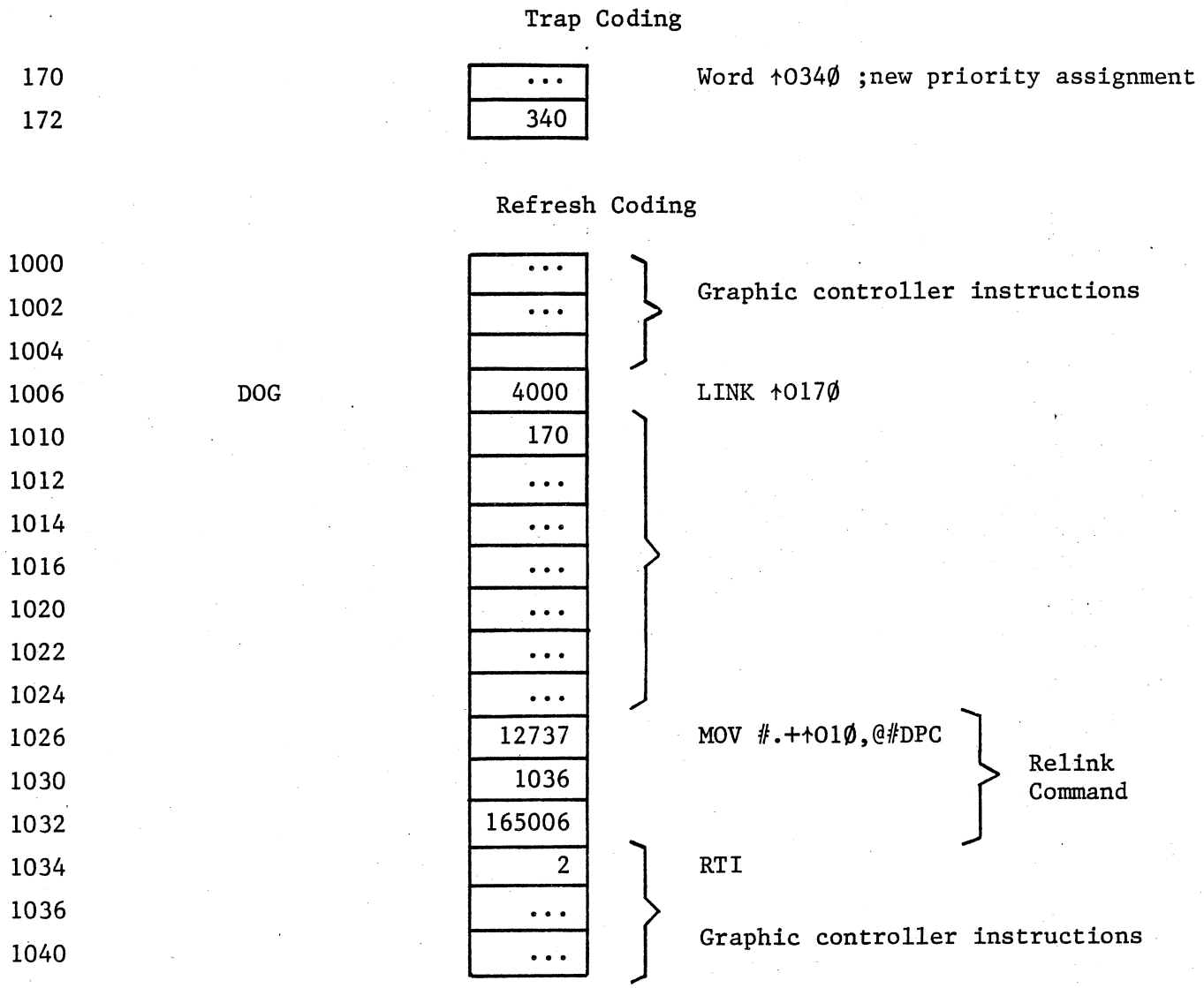


Figure 7-4 Sync Link Program Coding Example.

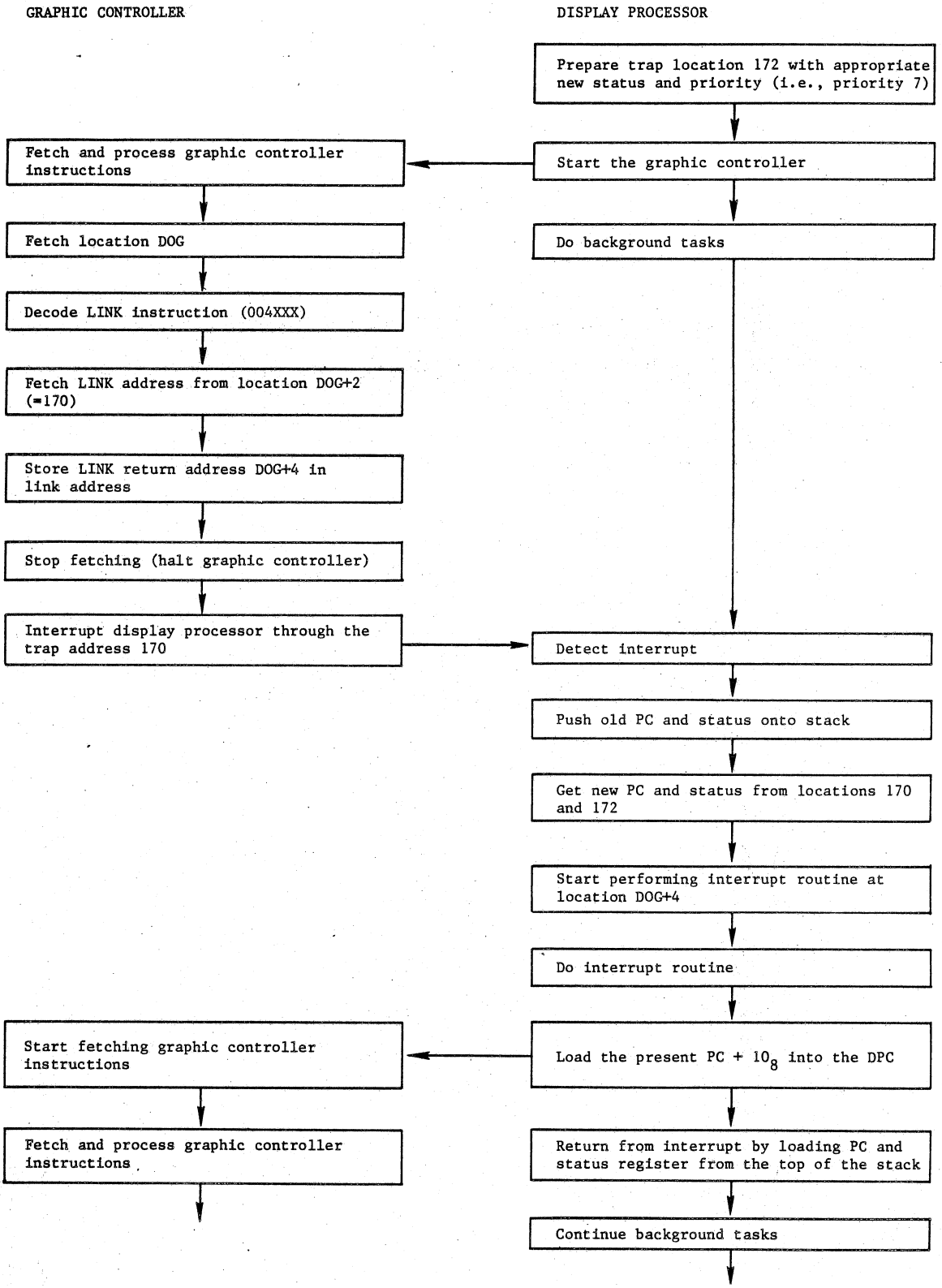
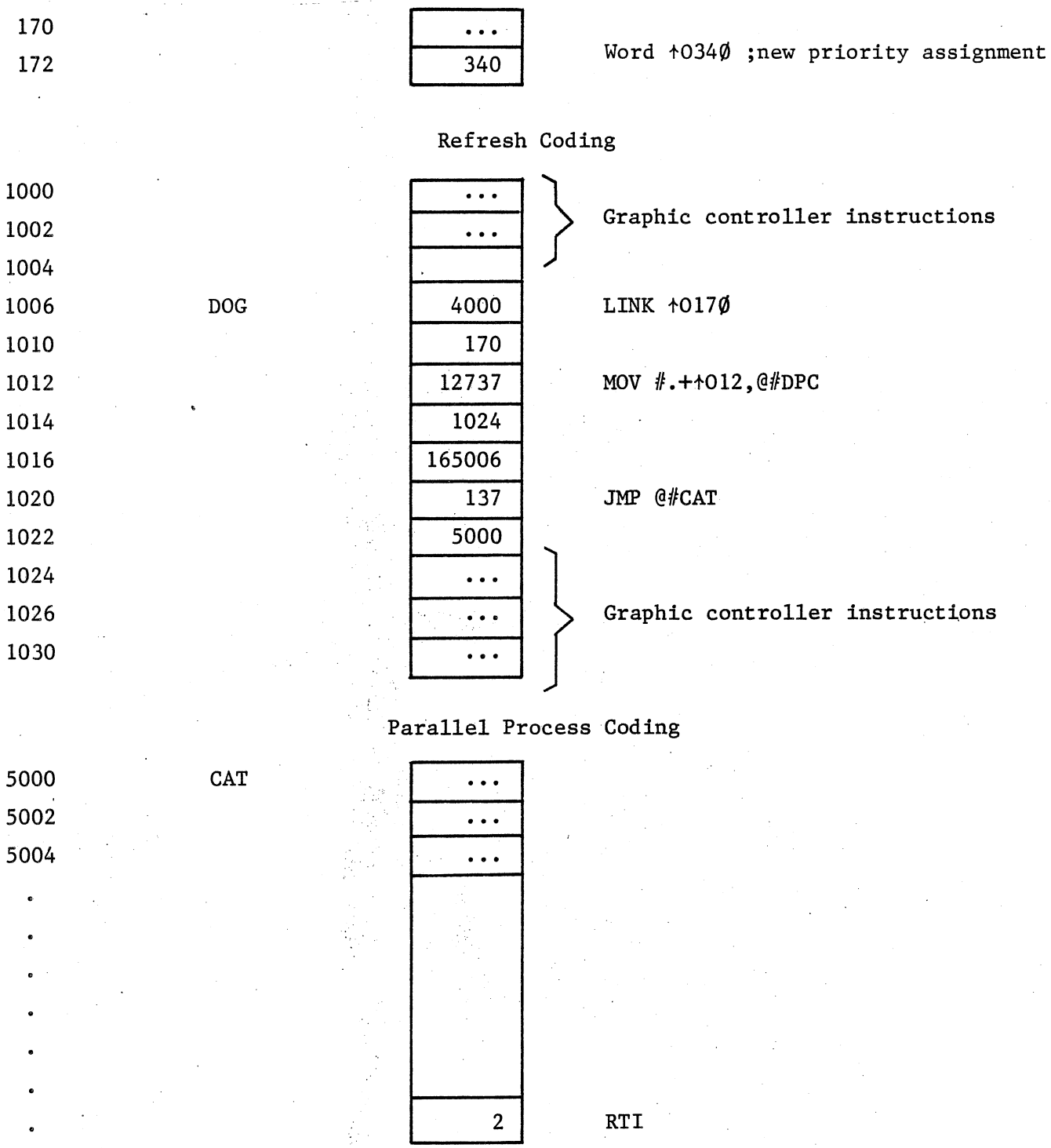


Figure 7-5 Sync Link Flow Chart Example.



GA-77-419-12

Figure 7-6 Super Sync Program Coding Example.

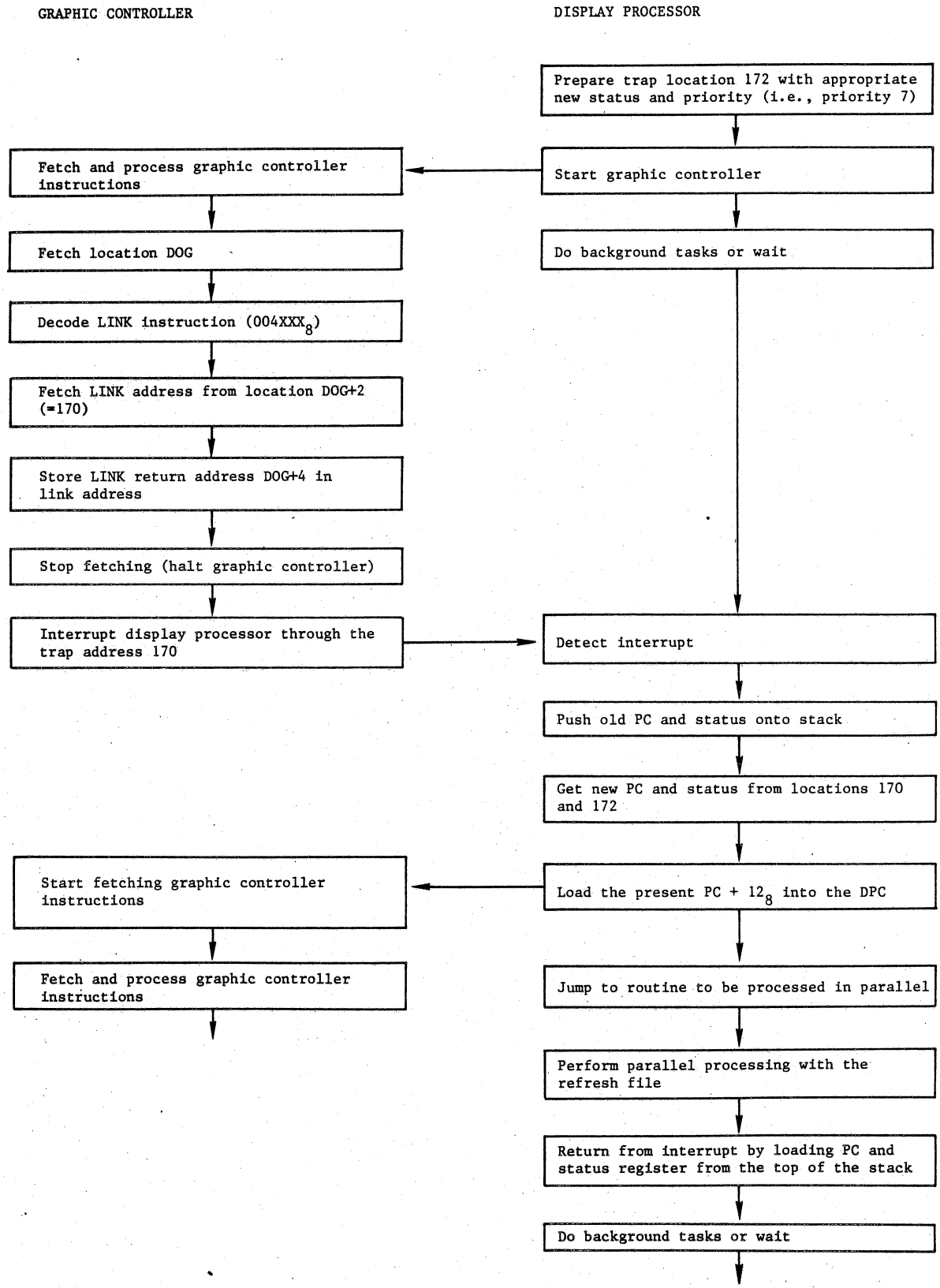


Figure 7-7 Super Sync Flow Chart Example.

maximum data load that can be handled by the graphic controller is not reduced significantly. If data internal to the graphic controller is required by the display processor, it can be read before the graphic controller is restarted and then processed while the graphic controller is running.

The super sync method can be used to perform both simple and intricate functions. It can also be used as a coarse real time clock that is synchronized with each refresh frame. This allows peripherals such as keyboards and PED's to be controlled on a frame rate basis.

Using the super sync method, extensive modifications can be made to a refresh file and synchronized with the actual drawing of the image. After any portion of an image is drawn, the graphic controller does not process the corresponding portion of the refresh file for a period at least equal to the refresh rate selected. Therefore, required modifications to the refresh file can be made while the graphic controller is busy causing other portions of the image (or other images) to be drawn. As long as the modifications are performed in the allotted time, there is no danger of refreshing the displayed image with a partially modified refresh file.

Thus entire blocks of memory can be cleared and rewritten without having to halt the graphic controller. Without the synchronizing ability provided by the super sync method, the graphic controller would have to be halted during the entire period that modifications were being made. The interrupt nesting ability of the display processor permits multiple use of the super sync method in a refresh file.

7.4 THE GRAPHIC CONTROLLER AS A DEVICE

Although the graphic controller operates independently of the display processor, it is under the control of the display processor at all times. The graphic controller can be halted and restarted at any time by the display processor and the graphic controller registers are at all times available to the display processor for purposes of reading, writing, or testing data as required. As far as the display processor is concerned, therefore, the graphic controller can be considered as a device connected to the controller bus.

Section 4 contains complete descriptions of all graphic controller registers and lists the address that is assigned to each. With the exception of the function control stop register (FUNS), the sense register (SENS), and the mask register (MKR), data should not be read from or written into any graphic controller registers

unless the graphic controller is halted. With the graphic controller running, such operations normally result in an error interrupt to the display processor through location 10.

7.5 THE PARALLEL INTERFACE AS A DEVICE

As previously discussed, if a parallel interface is installed in the terminal controller, GCP+ assumes that communications with the host computer are to be handled via this interface (parallel interface no. 1).

NOTE

> Register addresses for four parallel interface cards have been assigned to permit future expansion. The discussions in this section assume that a single parallel interface (no. 1)is installed for handling communications with the host computer. Refer to Section 4 for complete descriptions of all parallel interface registers and their assigned addresses.

The parallel interface can operate in either a DMA mode or a single-word transfer mode. The DMA mode is initiated when a non-zero value (two's complement of the number of words to be transferred) is written into the word count register (WCR1) and continues until the specified number of words has been transferred. Bit 2 in the status register (STR1) determines the direction of transfer and the value in the memory address register (MAR1) determines the starting memory address to be used for the DMA operation.

When the parallel interface is operated in the single-word transfer mode, data sent from the host computer to the GRAPHIC 7 is read from the output data register (ODR1) while data to be sent from the GRAPHIC 7 to the host computer is written into the input data register (IDR1). In actuality, the ODR1 and IDR1 are a single dual-purpose register and, therefore, the same address is used for both. The direction of data transfer is determined by control signals to or from the host computer as appropriate (refer to paragraph 4.4.2).

7.5.1 PROGRAMMING EXAMPLES

Proper handling of the parallel interface requires knowledge of the handshaking requirements of the communications with the host computer. The following coding examples illustrate methods for handling the parallel interface in its various operating modes.

a. To transfer a single word from the host computer (output data transfer):

```
WAIT:   TSTB  @#STR1          ;TEST 'OUTPUT CONTROL' BIT OF STR1
        BPL   WAIT            ;  UNTIL SET BY HOST
        MOV   @#ODR1,RØ       ;MOVE DATA WORD INTO R0
        BIS   #4Ø,@#STR1      ;SET 'OUTPUT WORD RECEIVED'BIT IN STR1
ACKNLG: BIT   #4Ø,@#STR1      ;WAIT FOR ACKNOWLEDGE
        BNE   ACKNLG          ;  BY HOST CLEARING THE BIT
```

b. To transfer a single word to the host computer (input data transfer):

```
        MOV   RØ,@#IDR1       ;MOVE DATA WORD INTO IDR1
        BIS   #2ØØØØ,@#STR1   ;SET 'INPUT WORD REQUEST' BIT IN STR1
WAIT:   TST   @#STR1          ;WAIT FOR HOST TO ACKNOWLEDGE BY
        BMI   WAIT            ;  CLEARING 'INPUT NOT READY' BIT OF STR1
```

c. To set up a DMA transfer from the host computer (output data transfer):

```
WAIT    TSTB  @#STR1          ;TEST 'OUTPUT CONTROL' BIT OF STR1
        BPL   WAIT            ;  UNTIL SET BY HOST
        BIC   #1Ø,@#STR1      ;ENABLE DMA OUTPUT MODE
        MOV   #DATABF,@#MAR1  ;SET MAR1 TO INTERNAL BUFFER ADDR
        MOV   #-1ØØ.,@#WCR1   ;START DMA TRANSFER OF 100 WORDS
COMPL:  BIT   #2Ø,@#STR1      ;TEST 'DMA COMPLETE' BIT IN STR1
        BEQ   COMPL           ;  (BIT IS SET WHEN DMA COMPLETE)
```

d. To set up a DMA transfer to the host computer (input data transfer):

```
        BIS   #1Ø,@#STR1      ;ENABLE DMA INPUT MODE
        MOV   #DATABF,@#MAR1  ;SET MAR1 TO INTERNAL BUFFER ADDR
        MOV   #-1ØØ.,@#WCR1   ;START DMA TRANSFER OF 100 WORDS
WAIT:   BIT   #2Ø,@#STR1      ;TEST 'DMA COMPLETE'BIT IN STR1
        BNE   WAIT            ;  (BIT IS SET WHEN DMA IS COMPLETE)
```

7.5.2 INTERRUPT OPERATION

The coding examples in paragraph 7.5.1 assumed that the interrupt capabilities provided by the parallel interface were not used. An interrupt capability is provided for both input and output data transfers. These interrupts can be enabled or disabled separately as required by changing the status of bit 14 (input interrupt enable) and bit 6 (output interrupt enable) in the status register. Setting a bit enables the associated interrupt while clearing a bit disables it.

When the input interrupt enable bit is set, an interrupt to the display processor occurs when the host computer acknowledges that data has been taken or when an input DMA operation is complete. When the output interrupt enable bit is set, an interrupt to the display processor occurs when output data is available from the host computer or when an output DMA operation is complete.

An attention interrupt is also provided by the parallel interface for special applications. This interrupt is enabled when bit 11 (attention interrupt enable) of the status register is set and disabled when bit 11 is cleared. The attention interrupt is associated with status register bits 9 (attention no. 1) and 10 (attention no. 2) which reflect the states of the two attention signals from the host computer. When the attention interrupt is enabled, an interrupt to the display processor occurs whenever one of the two attention signals changes from a low to a high state.

Bits 0 (spare input no. 1) and 12 (spare input no. 2) of the status register are provided to enable special signals to be sent from the display processor to the host computer via the parallel interface. These bits may be programmed as required.

7.6 THE SERIAL INTERFACE AS A DEVICE

Up to nine serial interface ports can be associated with one GRAPHIC 7 system. One port is contained on the ROM and status logic card and four ports are contained on each multiport serial interface card (up to two multiport serial interface cards may be installed in a terminal controller). All serial interface ports operate at speeds up to 9600 baud. The following paragraphs describe the use of the ports provided by each type of card.

7.6.1 ROM AND STATUS LOGIC CARD PORT

The serial interface port on the ROM and status logic card is used to interface a teletypewriter to the GRAPHIC 7 for maintenance and diagnostic purposes. This interface operates in a manner similar to the standard teletypewriter interface used in conjunction with minicomputers of the PDP-11 type manufactured by Digital Equipment Corporation (DEC). Instructions for its use are contained in the DEC PDP-11/04/34/45/55/60 Processor Handbook which should be used as a supplement to this manual. Registers associated with this port are described in paragraph 4.4.1. Note that the register formats are the same as the formats for corresponding registers associated with the multiport serial interface ports.

7.6.2 MULTIPORT SERIAL INTERFACE PORTS

Ports on the multiport serial interface cards are the ports through which peripheral devices (keyboards, PHOTOPENs, PED's, and the hardcopy unit) communicate with the display processor. The host computer also communicates with the display processor through a multiport serial interface port when a parallel interface is not used for the purpose. All four ports on each multiport serial interface card can

function as basic serial interface ports. Additionally, the first port on each card is provided with full RS-232C capabilities for the purpose of communicating with a host computer or driving a modem. Refer to paragraph 4.4.1 for a list of the devices assigned to each port and a description of each type of register associated with the ports.

Using a multiport serial interface port is somewhat dependent upon the device to which the port is connected. The following paragraphs, therefore, discuss each type of device separately and give examples of how the associated port is used.

7.6.2.1 Host Computer

When communications with the host computer are handled via a serial interface, the host computer is connected to serial interface port 1. Examples of representative coding sequences used to handle these communications are as follows:

a. To receive data from the host computer:

```
WAIT:   TSTB   @#RSR1          ;TEST 'RECEIVER DONE'BIT OF RSR1
        BPL    WAIT            ;  (DONE INDICATES CHARACTER RECEIVED
                                   FROM HOST)
        MOV    @RDB1, RØ       ;PUT CHAR IN LOW ORDER BYTE OF R0
```

b. To send data to the host computer:

```
        MOV    RØ,@#TDB1       ;MOVE CHAR FROM LOW ORDER BYTE OF R0 TO TDB1
WAIT:   TSTB   @#TSR1          ;TEST 'TRANSMITTER READY' BIT OF TSR1
        BPL    WAIT            ;  UNTIL SET BY SERIAL INTERFACE
                               ;(READY INDICATES CHARACTER SENT TO HOST)
```

NOTE

The preceding examples do not use the interrupt capabilities provided by the serial interface port. If desired, interrupt processing techniques may also be used.

7.6.2.2 Keyboards

GCP+ accepts inputs from alphanumeric/function keyboards via serial interface ports 3 and 7. All inputs from port 3 are identified as inputs from keyboard No. 1 and inputs from port 7 are identified as inputs from keyboard No. 2.

An example of a coding sequence used to accept keyboard inputs is as follows:

To obtain an alpha character from a keyboard:

```
WAIT:  TSTB   @#RSR3        ;TEST 'RECEIVER DONE' BIT OF RSR3
       BPL    WAIT          ;  UNTIL SET BY KEYSTROKE
       MOV    @#RDB3,RØ     ;PUT CHAR IN LOW ORDER BYTE OF R0
       BIC    #1776ØØ,RØ    ;CLEAR R0 EXCEPT FOR 7-BIT ASCII CHAR CODE
       BIT    #100,RØ       ;SEE IF FUNCTION
                            ;OR MATRIX KEY
       BEQ    1$            ;BRANCH FOR FUNCTION
                            ;OR MATRIX KEY
;CODE TO PROCESS CHARACTER
;
1$:;CODE TO PROCESS FUNCTION OR MATRIX KEY
```

The lighting of function or matrix keys on a keyboard requires that five bytes be sent to the keyboard via the associated interface. The first byte (224_8) sets up the keyboard to accept the data in the four bytes that follow. Bytes 2 and 3 contain data for lighting function keys 0 through 7 and 8 through 15 respectively. Bytes 4 and 5 contain data for lighting matrix keys 0 through 7 and 8 through 15 respectively. A key is lighted when its corresponding bit is set and not lighted when its corresponding bit is cleared. Function and matrix keys on a keyboard are designated as follows:

FUNCTION KEYS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

MATRIX KEYS

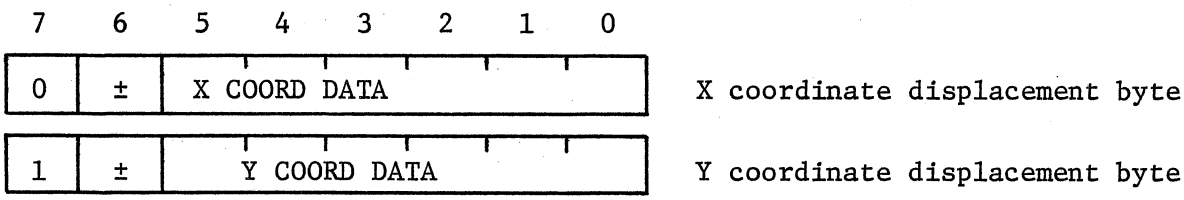
7	8	9	15
4	5	6	14
1	2	3	13
10	0	11	12

An example of coding that could be used to light function keys 1, 3, and 10 and matrix keys 5, 12, and 13 is as follows:

```
        MOV   #224,RØ       ;SET UP KEYBOARD
        JSR   PC,OUT        ;  TO ACCEPT LAMP DATA
        MOV   #12,RØ        ;LIGHT FUNCTION
        JSR   PC,OUT        ;  KEYS 1 AND 3
        MOV   #4,RØ         ;LIGHT FUNCTION
        JSR   PC,OUT        ;  KEY 10
        MOV   #4Ø,RØ        ;LIGHT MATRIX
        JSR   PC,OUT        ;  KEY 5
        MOV   #6Ø,RØ        ;LIGHT MATRIX
        JSR   PC,OUT        ;  KEYS 12 AND 13
         .
         .
         .
OUT:    MOV   RØ,@#TDB3     ;MOVE BYTE FROM RO to TDB3
WAIT:   TSTB  @#TSR3        ;TEST 'TRANSMITTER READY' BIT OF TSR3
        BPL   WAIT          ;  UNTIL SET BY INTERFACE (BYTE TAKEN)
        RTS   PC            ;RETURN FROM SUBROUTINE
```

7.6.2.3 PED's

PED's (e.g., trackball, forcestick or data tablet) are assigned to serial interface ports 4 and 8. Inputs from port 4 are identified by GCP+ as coming from PED no. 1 while inputs from port 8 are identified as coming from PED no. 2. Inputs from trackball/forcestick represent coordinate displacement data in the form of two successive 8-bit bytes that are updated at a maximum rate of 37.5 Hz. No inputs are generated unless the PED is moved. The format of each byte is as follows (note that the coordinate data in each byte is in two's complement form):



The handling of PED data is similar to the handling of inputs from a keyboard. Refer to paragraph 7.6.2.2 for examples of coding that can be used.

7.6.2.4 Hardcopy Unit

The hardcopy unit is assigned to serial interface port 5. To obtain a hardcopy of a displayed image, it is only necessary to write a non-zero value into the transmit data buffer of port 5 (TDB5). An example of coding that may be used to obtain a hard copy is as follows:

```
MOV   #1,RØ         ;PUT NON-ZERO VALUE IN R0
MOV   RØ,@#TDB5     ;MOVE VALUE FROM R0 to TDB5
```

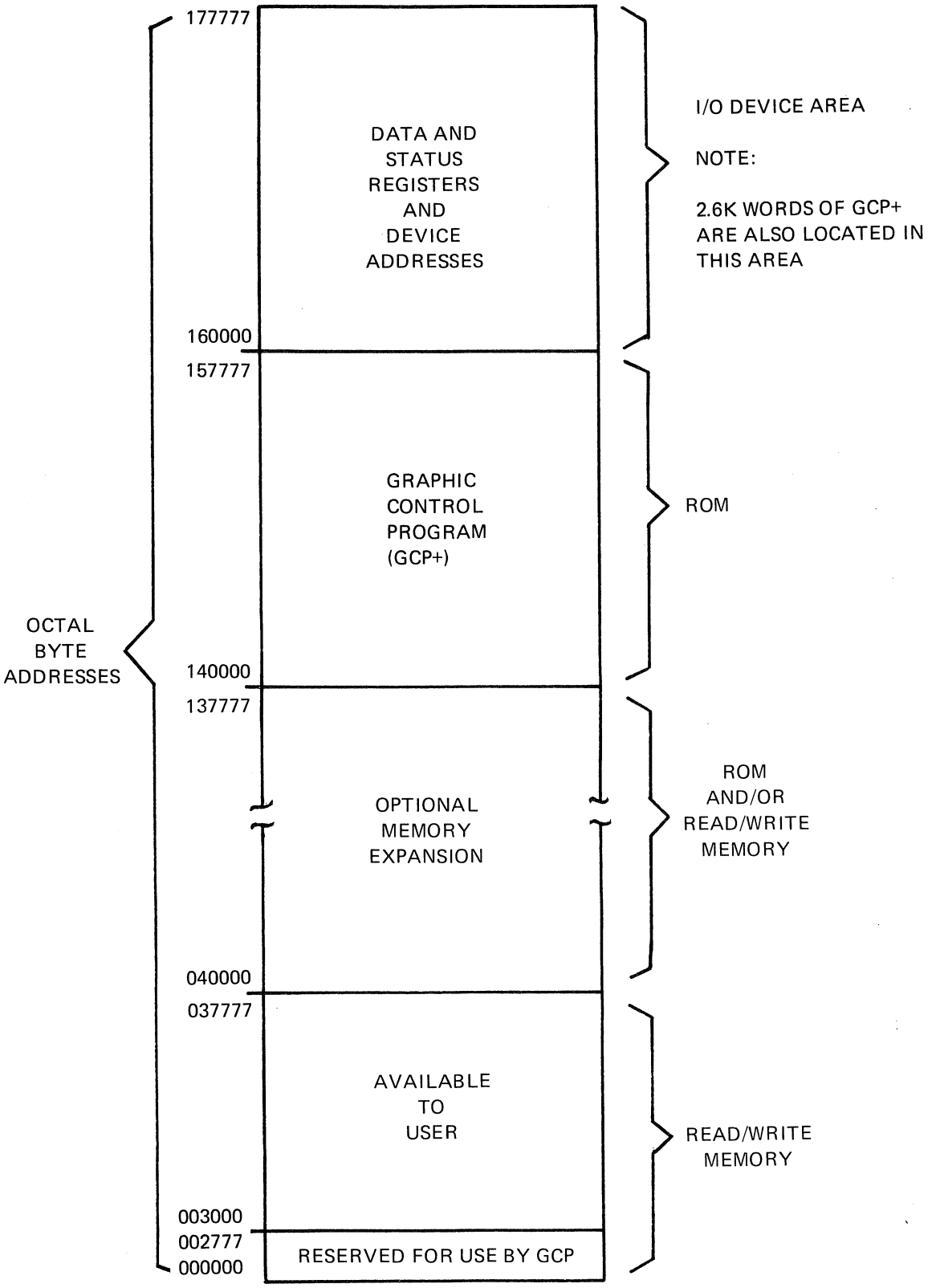
NOTE

A second request for a hardcopy must not be made during the time that the hardcopy unit is operating (approximately 8-10 seconds). Additionally, the displayed image should remain static during the time that the hardcopy unit is operating.

An optional hardcopy multiplexer unit is available for the hardcopy unit that permits it to be used with more than one terminal controller. If this option is installed and a request is made for a hardcopy while the hardcopy unit is operating, the request is stored by the internal logic of the multiplexer unit and the copy request is deferred until the copy request from the other terminal controller is processed. In all cases the multiplexer unit returns a copy complete message to the terminal controller to indicate the success or failure of the last copy request. This is done by setting the 'receiver done' bit (bit 7) in the receive status register of port 5 (RSR5) and by loading 0_8 (success) or a 377_8 (failure) into the receive data buffer (RDB5). The 377 code indicates that the hardcopy is either off line or the hardcopy multiplexer switch is in the manual mode.

NOTE

Remote generation of hardcopies requires that a serial cable be connected between serial interface port 5 and the hardcopy unit (or multiplex switch).



H-79-0348-1

Figure A-1 GRAPHIC 7 System Memory Map.

TABLE A-1

GRAPHIC 7 LOCAL MODE COMMAND SUMMARY

Keyboard Entry	Operation
RETURN	Executes local mode command or returns system to local monitor level.
nnnnnn/	Displays contents of memory address nnnnnn (octal).
/	Increments memory address counter by two and displays address contents.
∧ or ↑	Decrements memory address counter by two and displays address contents.
Bn	Select different memory bank. (BØ 0-32K; B1 32-64K; B2 64-96K; B3 96-128K; and B4 16-32K RAM).
S	Transfers GRAPHIC 7 to system mode operation.
T RETURN	Transfers to the verification test pattern.
L RETURN	Loads memory from paper tape reader.
nnnnL RETURN	Loads selected option from expansion module.
U RETURN	Unload all options.
O RETURN	Display status of all options loaded.
Q	Decrements contents of display processor Q register by two and displays result. Used with diagnostics to indicate address at which display processor halted.
nnnnnnD RETURN	Directs graphic controller to display refresh file beginning at address nnnnnn (octal).
nnnnnnG RETURN	Transfers control of display processor to program beginning at memory address nnnnnn (octal).
Y RETURN	Calls teletypewriter emulation program. After entering emulation program, function key FØ clears CRT screen. Function key F1 selects full or half duplex operation; receipt of octal code 035 from the host computer or pressing function key F13 transfers GRAPHIC 7 to system operating mode.
RUB OUT	Deletes last octal entry from keyboard.

TABLE A-2

GRAPHIC CONTROLLER INSTRUCTION SET SUMMARY

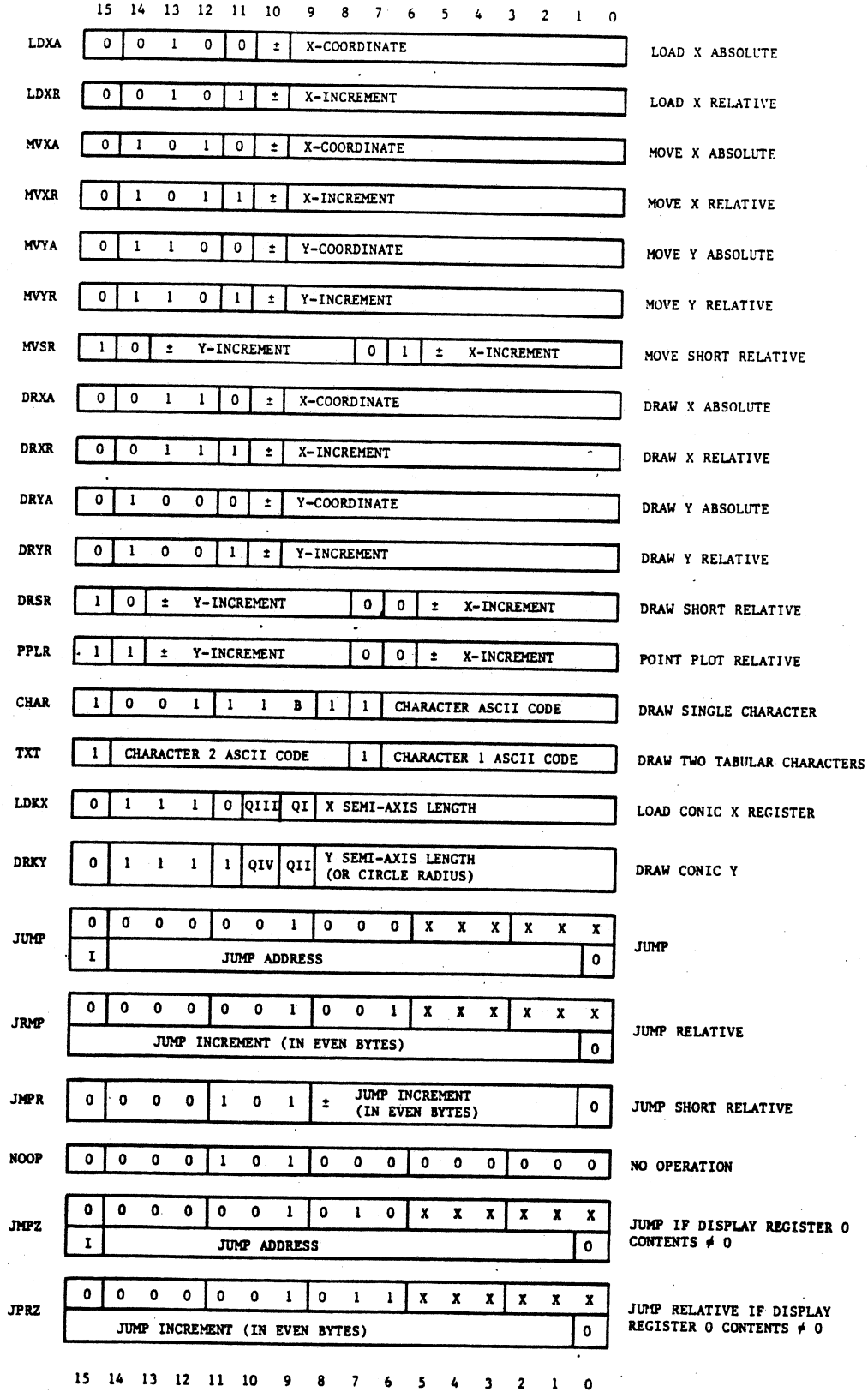


TABLE A-2

GRAPHIC CONTROLLER INSTRUCTION SET SUMMARY (Cont)

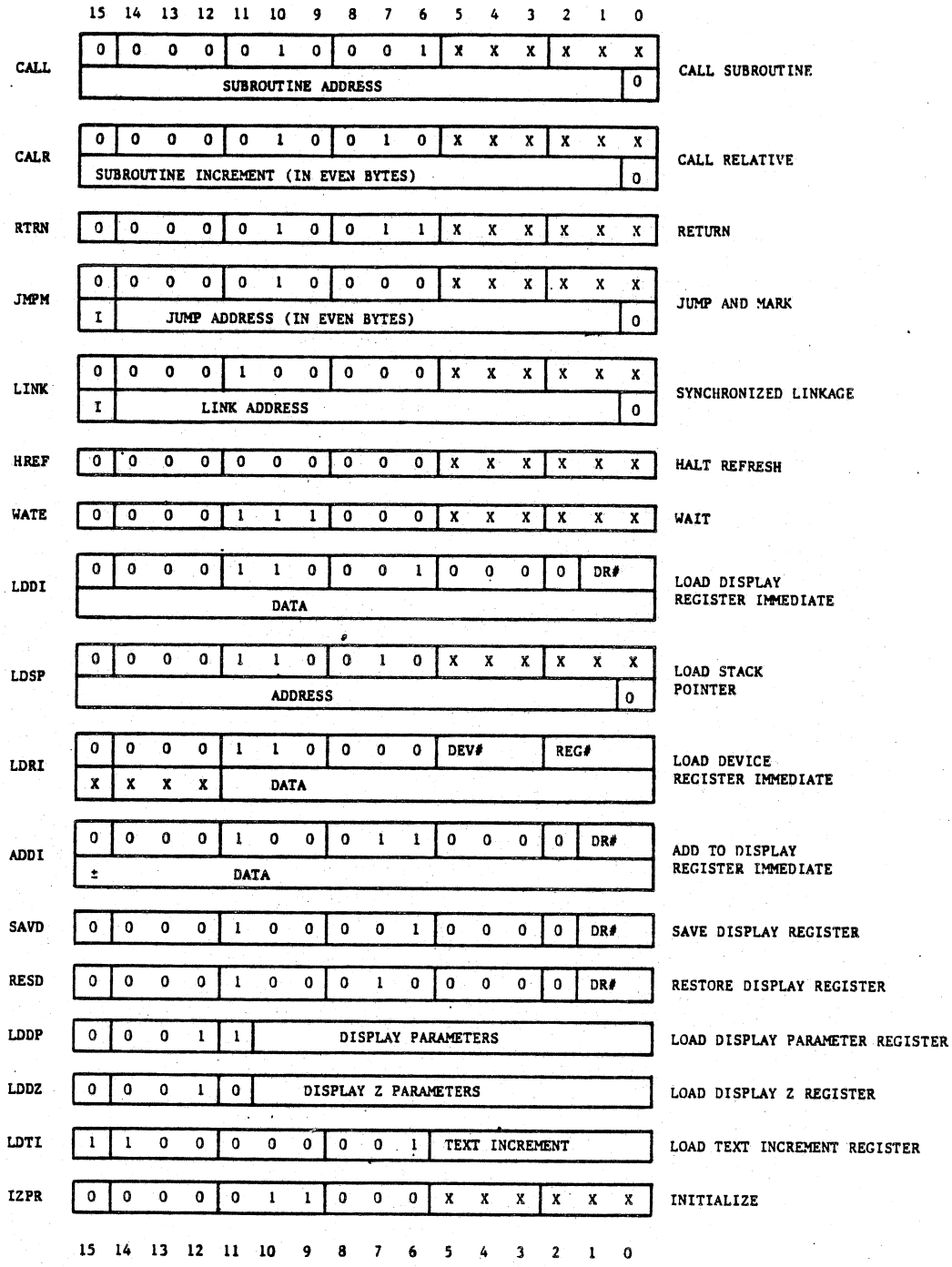


TABLE A-3

GRAPHIC CONTROLLER REGISTER FORMAT SUMMARY

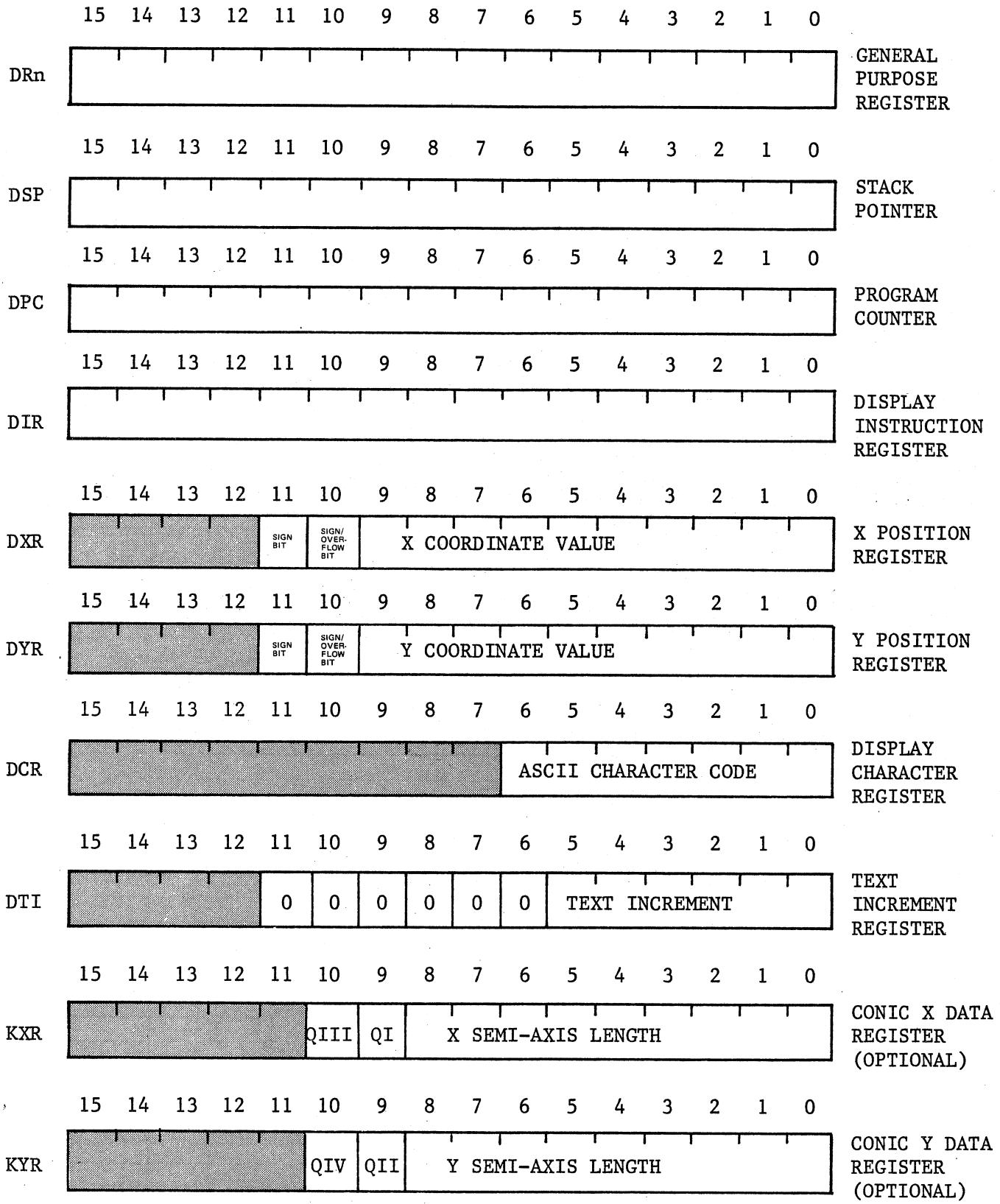


TABLE A-3

GRAPHIC CONTROLLER REGISTER FORMAT SUMMAY (Cont)

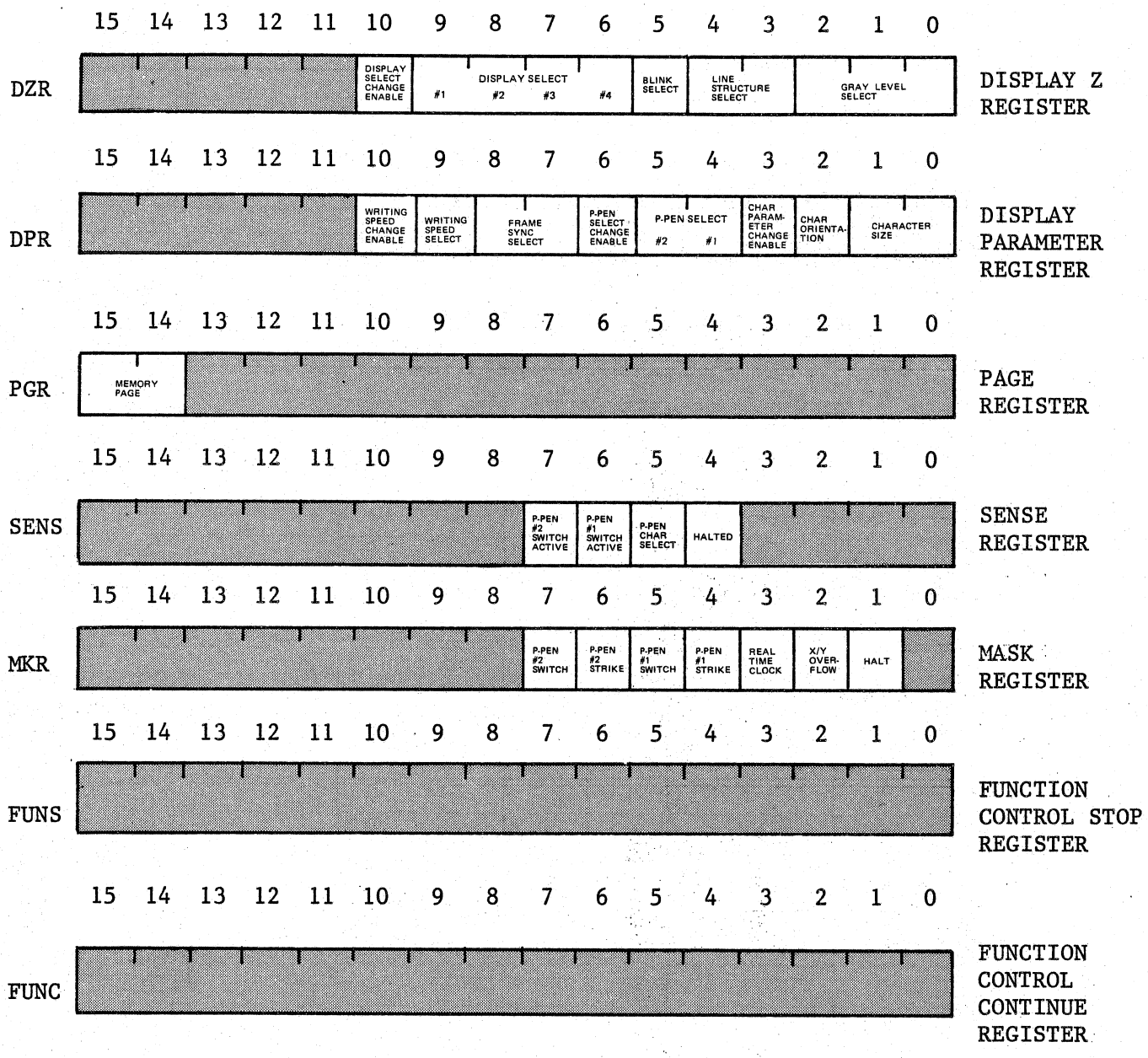
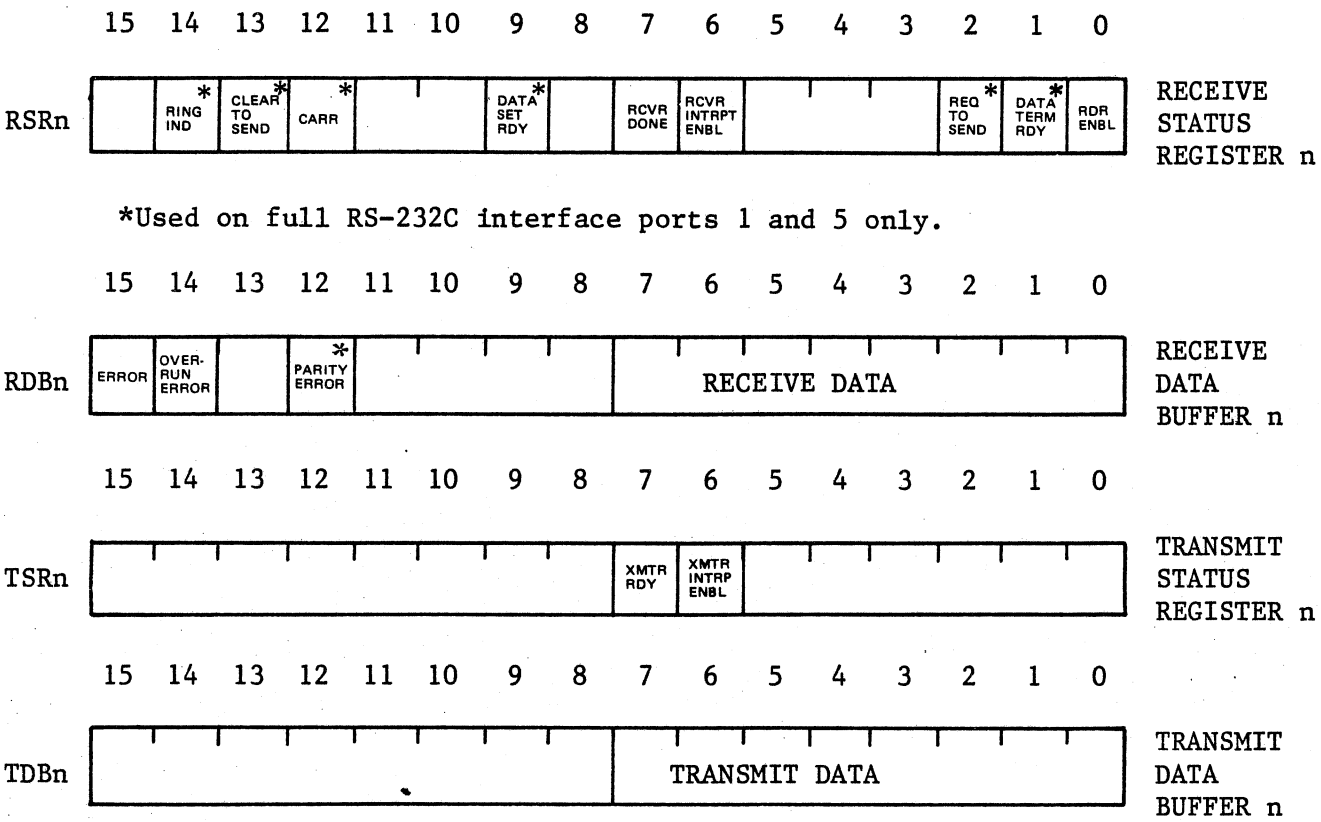


TABLE A-4

SERIAL INTERFACE REGISTER FORMAT SUMMARY



NOTE: Unidentified bits not used.

TABLE A-5

PARALLEL INTERFACE REGISTER FORMAT SUMMARY

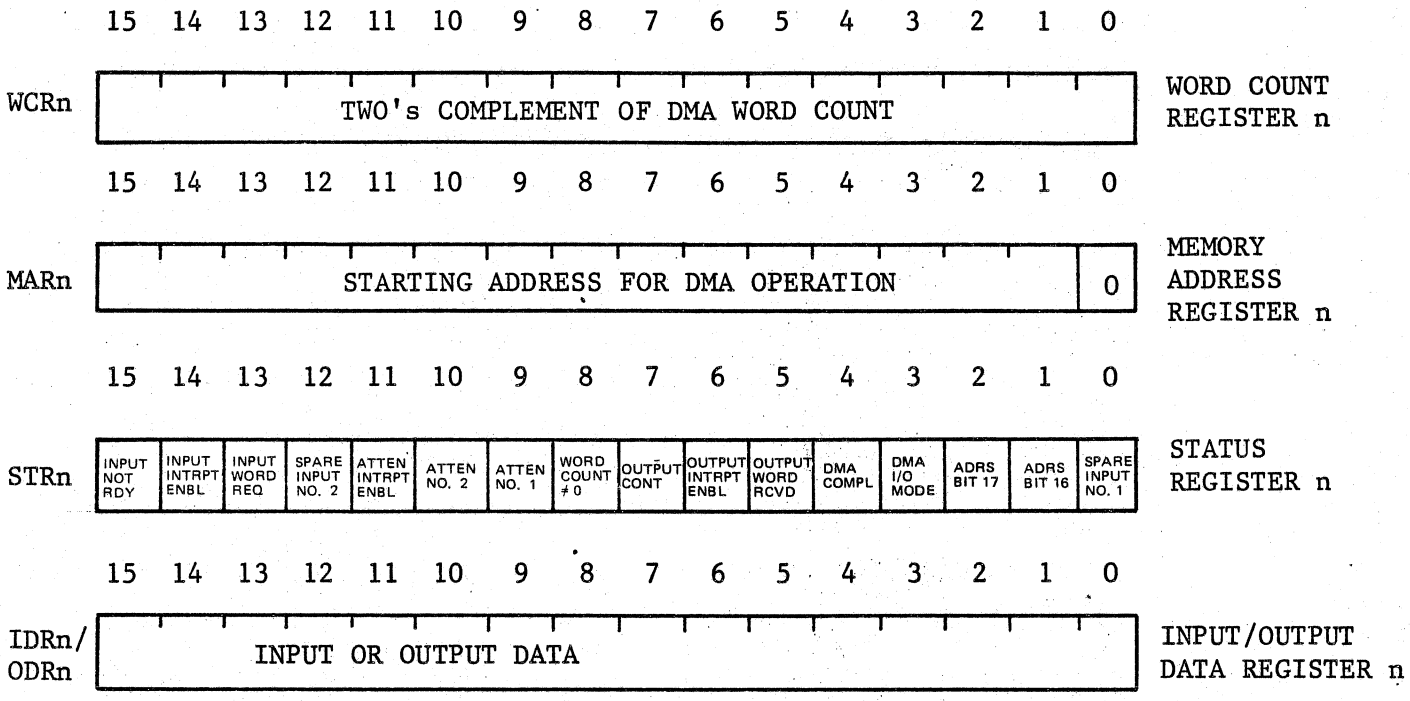


TABLE A-6

GRAPHIC 7 REGISTER DESIGNATIONS AND ADDRESS ASSIGNMENTS

Device	Register	Mnemonic	Address
DISPLAY PROCESSOR	Processor Status Word	PSW	177776
	Reserved		177760 thru 177774
ROM AND STATUS LOGIC SERIAL INTERFACE PORT	TTY Receive Status Register	TTYRSR	177560
	TTY Receive Data Buffer	TTYRDB	177562
	TTY Transmit Status Register	TTYTSR	177564
	TTY Transmit Data Buffer	TTYTDB	177566
PARALLEL INTERFACE CARD NO. 1 (OPTIONAL)	Word Count Register 1	WCR1	172410
	Memory Address Register 1	MAR1	172412
	Status Register 1	STR1	172414
	Input/Output Data Register 1	IDR1/ODR1	172416
PARALLEL INTERFACE CARD NO. 2 (OPTIONAL)	Word Count Register 2	WCR2	172430
	Memory Address Register 2	MAR2	172432
	Status Register 2	STR2	172434
	Input/Output Data Register 2	IDR2/ODR2	172436
PARALLEL INTERFACE CARD NO. 3 (OPTIONAL)	Word Count Register 3	WCR3	172450
	Memory Address Register 3	MAR3	172452
	Status Register 3	STR3	172454
	Input/Output Data Register 3	IDR3/ODR3	172456
PARALLEL INTERFACE CARD NO. 4 (OPTIONAL)	Word Count Register 4	WCR4	172470
	Memory Address Register 4	MAR4	172472
	Status Register 4	STR4	172474
	Input/Output Data Register 4	IDR4/ODR4	172476
MULTIPORT SERIAL INTERFACE CARD NO. 1	Port No. 1 (Host Computer)		
	Receive Status Register 1	RSR1	176500
	Receive Data Buffer 1	RDB1	176502
	Transmit Status Register 1	TSR1	176504
	Transmit Data Buffer 1	TDB1	176506
	Port No. 2 (Spare)		
	Receive Status Register 2	RSR2	176510
	Receive Data Buffer 2	RDB2	176512
	Transmit Status Register 2	TSR2	176514
	Transmit Data Buffer 2	TDB2	176516

TABLE A-6

GRAPHIC 7 REGISTER DESIGNATIONS AND ADDRESS ASSIGNMENTS (Cont)

Device	Register	Mnemonic	Address
MULTIPORT SERIAL INTERFACE CARD NO. 1 (Cont)	Port No. 3 (Function Kybd 1)		
	Receive Status Register 3	RSR3	176520
	Receive Data Buffer 3	RDB3	176522
	Transmit Status Register 3	TSR3	176524
	Transmit Data Buffer 3	TDB3	176526
	Port No. 4 (PED 1)		
	Receive Status Register 4	RSR4	176530
	Receive Data Buffer 4	RDB4	176532
	Transmit Status Register 4	TSR4	176534
	Transmit Data Buffer 4	TDB4	176536
MULTIPORT SERIAL INTERFACE CARD NO. 2 (OPTIONAL)	Port No. 5 (Hard Copy Unit)		
	Receive Status Register 5	RSR5	176540
	Receive Data Buffer 5	RDB5	176542
	Transmit Status Register 5	TSR5	176544
	Transmit Data Buffer 5	TDB5	176546
	Port No. 6 (Spare)		
	Receive Status Register 6	RSR6	176550
	Receive Data Buffer 6	RDB6	176552
	Transmit Status Register 6	TSR6	176554
	Transmit Data Buffer 6	TDB6	176556
	Port No. 7 (Function Kybd 2)		
	Receive Status Register 7	RSR7	176560
	Receive Data Buffer 7	RDB7	176562
	Transmit Status Register 7	TSR7	176564
	Transmit Data Buffer	TDB7	176566
	Port No. 8 (PED 2)		
	Receive Status Register 8	RSR8	176570
	Receive Data Buffer 8	RDB8	176572
	Transmit Status Register 8	TSR8	176574
	Transmit Data Buffer 8	TDB8	176576
GRAPHIC CONTROLLER	Stack Pointer	DSP	165000
	General Purpose Register 0	DR0	165002
	General Purpose Register 1	DR1	165004
	Program Counter	DPC	165006
	Display Instruction Register	DIR	165010
	Text Increment Register	DTI	165012
	Display Parameter Register	DPR	165014
	Page Register	PGR	165014
	Display Z Register	DZR	165016
	X Position Register	DXR	165020
	Y Position Register	DYR	165022
	Display Character Register	DCR	165024

TABLE A-6

GRAPHIC 7 REGISTER DESIGNATIONS AND ADDRESS ASSIGNMENTS (Cont)

Device	Register	Mnemonic	Address
GRAPHIC CONTROLLER (Cont)	Conic X Data Register	KXR	165026
	Conic Y Data Register	KYR	165030
	General Purpose Register 2	DR2	165032
	General Purpose Register 3	DR3	165034
	Function Control Continue Register	FUNC	165036
	Function Control Stop Register	FUNS	165040
	Sense Register	SENS	177660
	Mask Register	MKR	177662
64K READ/WRITE MEMORY CARDS NO. 1 AND NO. 2	Page Register 1	PR1	172342
	Page Register 2	PR2	172344
	Page Register 3	PR3	172346
	Reserved		172340 172350 thru 172356

TABLE A-7

DISPLAY PROCESSOR TRAP ADDRESSES

Interruption Device	Interrupt	Trap Address
DISPLAY PROCESSOR	CPU Error	4
	Reserved Instruction	10
	Breakpoint Trap	14
	(Reserved)	24
	Emulator Trap	30
	Trap Instruction	34
ROM AND STATUS LOGIC CARD, SERIAL INTERFACE PORT	TTY Receive	60
	TTY Transmit	64
GRAPHIC CONTROLLER	Real Time Clock	100
	Halt	140
	X or Y Position Overflow	144
	PHOTOPEN No. 1 Strike	150
	PHOTOPEN No. 2 Strike	154
	PHOTOPEN No. 1 Switch Actuated	160
	PHOTOPEN No. 2 Switch Actuated	160
	Sync Link	170
PARALLEL INTERFACE CARD NO. 1	Input Data (to Host Computer)	120
	Output Data (from Host Computer)	124
	Attention (Optional)	130
PARALLEL INTERFACE CARD NO. 2	Input Data	400
	Output Data	404
	Attention (Optional)	410
PARALLEL INTERFACE CARD NO. 3	Input Data	420
	Output Data	424
	Attention (Optional)	430
PARALLEL INTERFACE CARD NO. 4	Input Data	440
	Output Data	444
	Attention (Optional)	450
MULTIPORT SERIAL INTERFACE CARD NO. 1	Port 1 - Host Computer	
	Input	300
	Output	304
	Port 2 - Spare	
	Input	310
	Output	314
	Port 3 - Alphanumeric/function Keyboard No. 1	
	Input	320
	Output	324

TABLE A-7

DISPLAY PROCESSOR TRAP ADDRESSES (Cont)

Interruption Device	Interrupt	Trap Address
MULTIPORT SERIAL INTERFACE CARD NO. 1 (Cont)	Port 4 - PED No. 1	
	Input	330
	Output	334
MULTIPORT SERIAL INTERFACE CARD NO. 2	Port 5 - Hard Copy Unit	
	Input	340
	Output	344
	Port 6 - Spare	
	Input	350
	Output	354
	Port 7 - Alphanumeric/function Keyboard No. 2	
	Input	360
	Output	364
	Port 8 - PED No. 2	
	Input	370
	Output	374

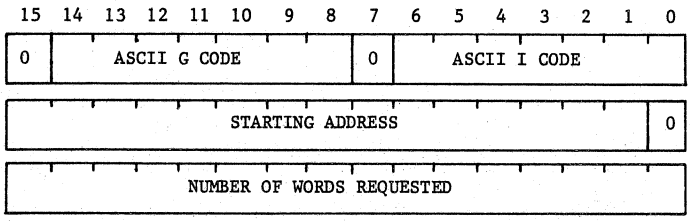
TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7

GI (H→G7)

GIVE IMAGE

(Page 5-23)

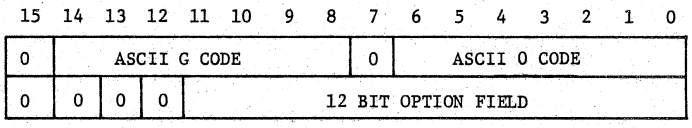


OCTAL	HEX
043511	4749

GO (H→G7)

GIVE OPTION STATUS

(Page 5-78)

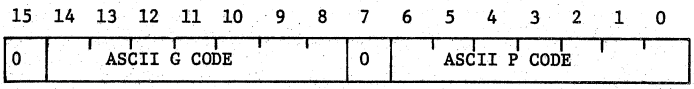


OCTAL	HEX
043517	474F

GP (H→G7)

GIVE PED NO. 1

(Page 5-49)

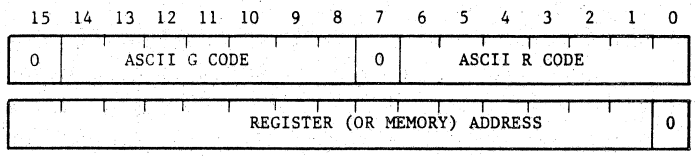


OCTAL	HEX
043520	4750

GR (H→G7)

GIVE REGISTER

(Page 5-26)

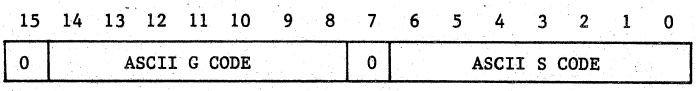


OCTAL	HEX
043522	4752

GS (H→G7)

GET STATUS OF PEDS

(Page 5-47)

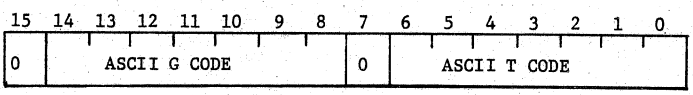


OCTAL	HEX
043523	4753

GT (H→G7)

GIVE PED NO. 2

(Page 5-49)



OCTAL	HEX
043524	4754

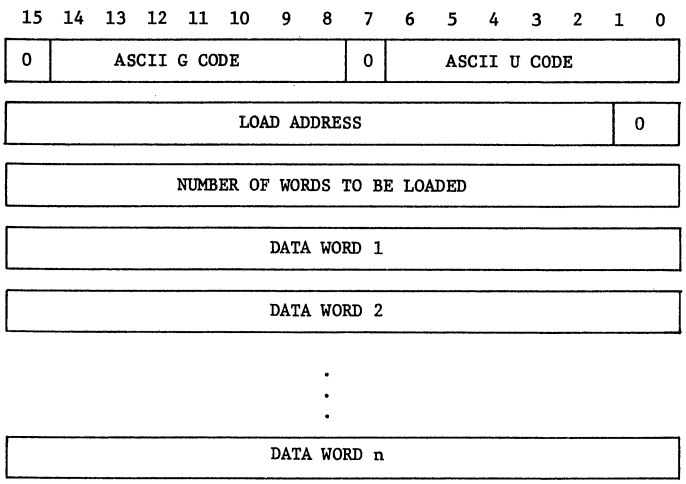
TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

GU (H→G7)

GRAPHIC UPDATE

(Page 5-64)

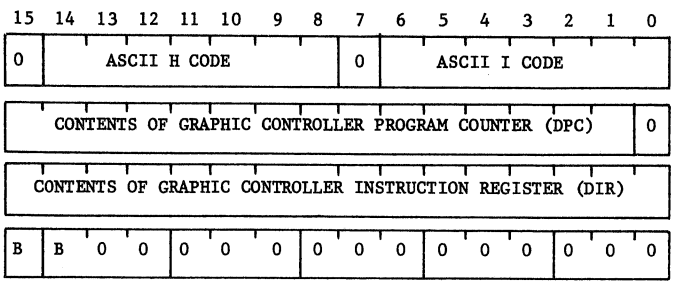


OCTAL	HEX
043525	4755

HI (G7→H)

HALT INTERRUPT

(Page 5-31)

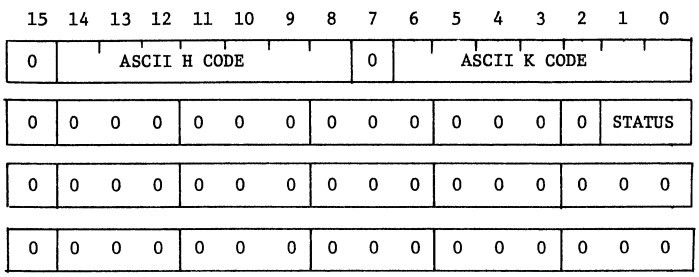


OCTAL	HEX
044111	4849

HK (G7→H)

HARD COPY COMPLETED

(Page 5-60

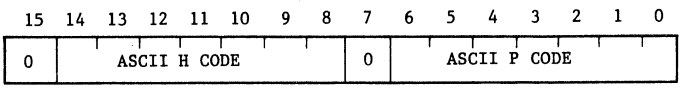


OCTAL	HEX
044113	484B

HP (H→G7)

HALT PICTURE

(Page 5-21)

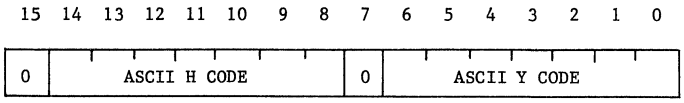


OCTAL	HEX
044120	4850

HY (H→G7)

INITIATE HARD COPY

(Page 5-59)



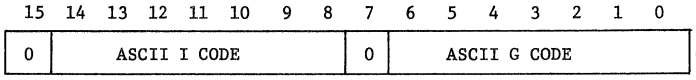
OCTAL	HEX
044131	4859

TABLE 8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

IG (H→G7)

INITIALIZE GCP+ TO SUPPORT FSP
(Page 45-62)

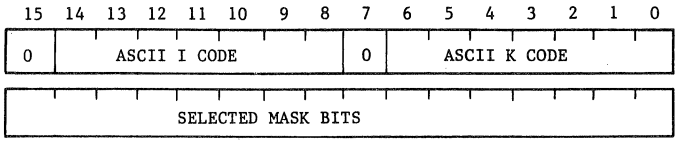


Command header

OCTAL	HEX
044507	4947

IK (H→G7)

INTERRUPT CONTROL
(Page 5-28)



Command header

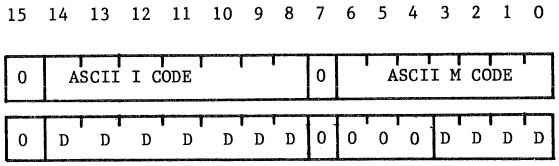
Word 1

OCTAL	HEX
044513	494B

Bit	Function or Interrupt Condition
0	Not used
1	Graphic controller halted
2	X or Y position overflow
3	Real time clock
4	PHOTOPEN 1 strike
5	Switch of PHOTOPEN 1 actuated
6	PHOTOPEN 2 strike
7	Switch of PHOTOPEN 2 actuated
8	PHOTOPEN item number enabled
9	Not used
10	PED no. 2 (serial interface port 8)
11	Alphanumeric/function keyboard no. 2 (serial interface port 7)
12	Spare (serial interface port 6)
13	PED no. 1 (serial interface port 4)
14	Alphanumeric/function keyboard no. 1 (serial interface port 3)
15	Spare (serial interface port 2)

IM (H→G7)

INITIALIZE I/O MESSAGE FORMATS
(Page 5-11)



Command Header

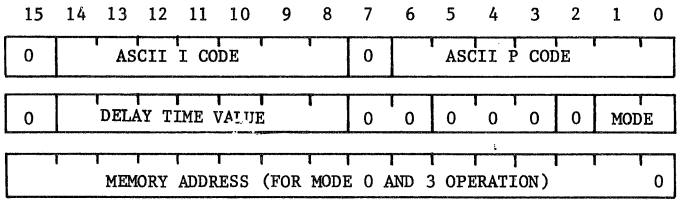
Word 1

OCTAL	HEX
044515	494D

Bit	Description
0	Polling
1	Send poll message back
2	Host to GRAPHIC 7 data not packed
3	Activate error detection
4-6	Reserved for future expansion
8-14	Special poll character

IP (H→G7)

INITIALIZE PED NO. 1
(Page 5-44)



Command header

Word 1

Word 2

OCTAL	HEX
044520	4950

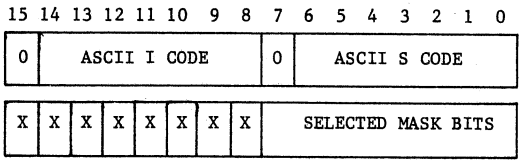
TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

IS (H→G7)

ENABLE SELECTED INTERRUPTS

(Page 5-29)



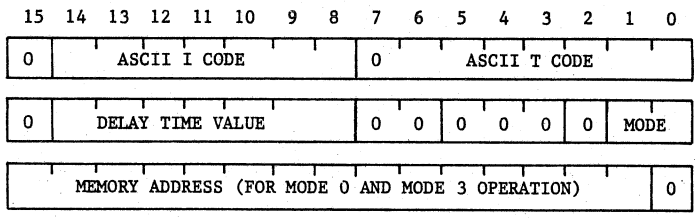
OCTAL	HEX
044523	4953

BIT	FUNCTION OR INTERRUPT CONDITION
0	Not used
1	Graphic controller halted
2	X or Y position overflow
3	Real time clock
4	PHOTOPEN 1 strike
5	Switch of PHOTOPEN 1 actuated
6	PHOTOPEN 2 strike
7	Switch of PHOTOPEN 2 actuated
8-15	These bits are ignored by GCP+

IT (H→G7)

INITIALIZE PED NO. 2

(Page 5-47)

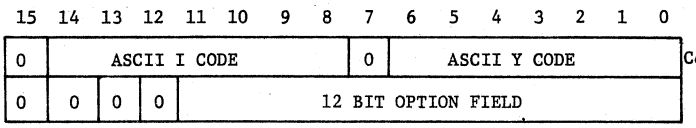


OCTAL	HEX
044524	4954

IY (H→G7)

INITIALIZE 2

(Page 5-78)



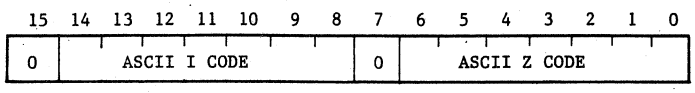
OCTAL	HEX
044531	4959

(OCTAL) OPTION FIELD	ACTION
Ø	Load all system automatic load options
4ØØØ	Unload all options
1 to 3777	Load specified option (if unloaded), initialize option, and update option status
4001 to 7777	

IZ (H→G7)

INITIALIZE

(Page 5-7)

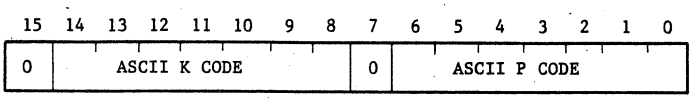


OCTAL	HEX
044532	495A

KP (H→G7)

CONTINUE PICTURE

(Page 5-22)

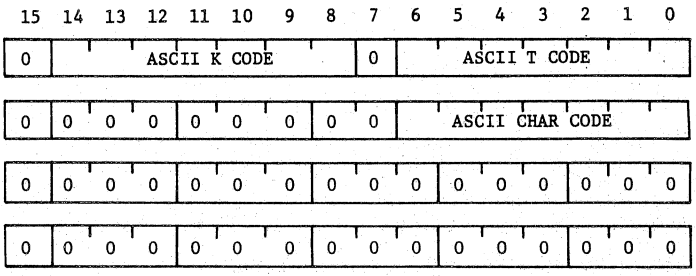


OCTAL	HEX
045520	4B50

TABLE A-8

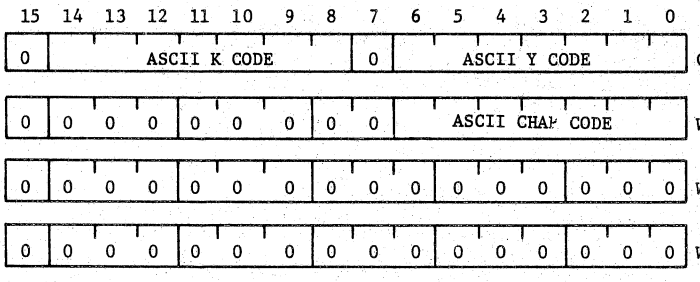
ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

KT (G7→H)
ALPHANUMERIC KEYBOARD NO. 2
(Page 5-39)



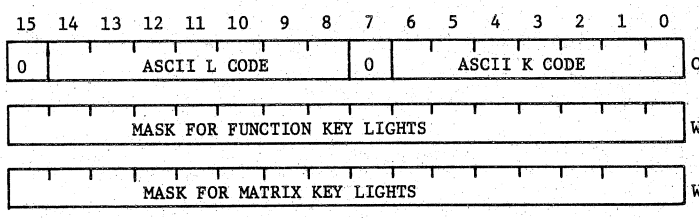
OCTAL 045524 HEX 4B54

KY (G7→H)
ALPHANUMERIC KEYBOARD NO. 1
(Page 5-38)



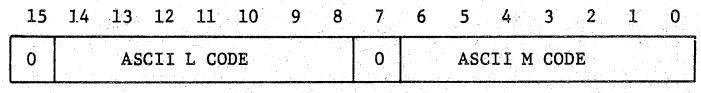
OCTAL 045531 HEX 4B59

LK (H→G7)
LIGHT KEYS ON FUNCTION KEYBOARD NO. 1
(Page 5-37)



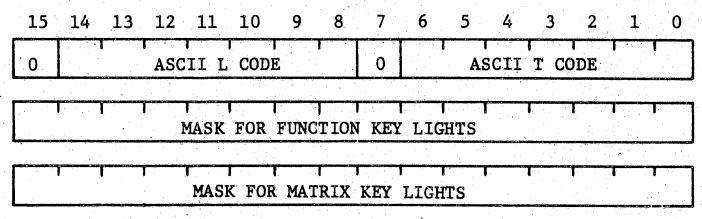
OCTAL 046113 HEX 4C4B

LM (H→G7)
LARGE MEMORY IN USE FOR FSP
(Page 5-70)



OCTAL 046115 HEX 4C4D

LT (H→G7)
LIGHT KEYS ON FUNCTION KEYBOARD NO. 2
(Page 5-38)

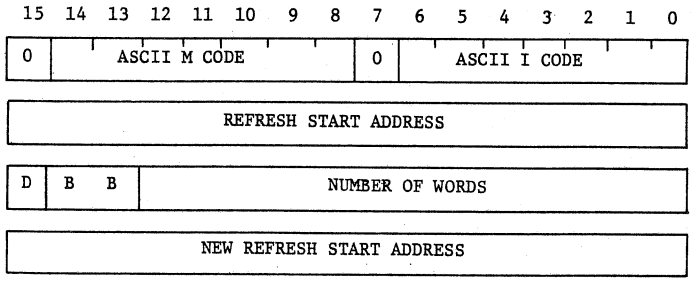


OCTAL 046124 HEX 4C54

TABLE A- 8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

MI (H→G7)
MOVE IMAGE
(Page 5-65)



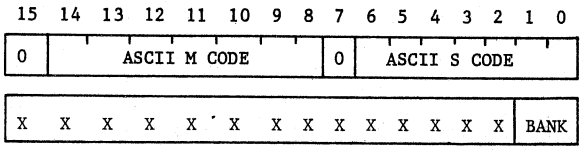
OCTAL	HEX
046511	4D49

BIT 15 VALUE

0	Transfer refresh data to current bank (ignore bits 13 and 14)
1	Transfer refresh data to bank specified in bits 13 and 14

BITS 14 13	DESTINATION BANK
0 0	0
0 1	1
1 0	2
1 1	3

MS (H→G7)
MEMORY BANK SELECT
(Page 5-17)

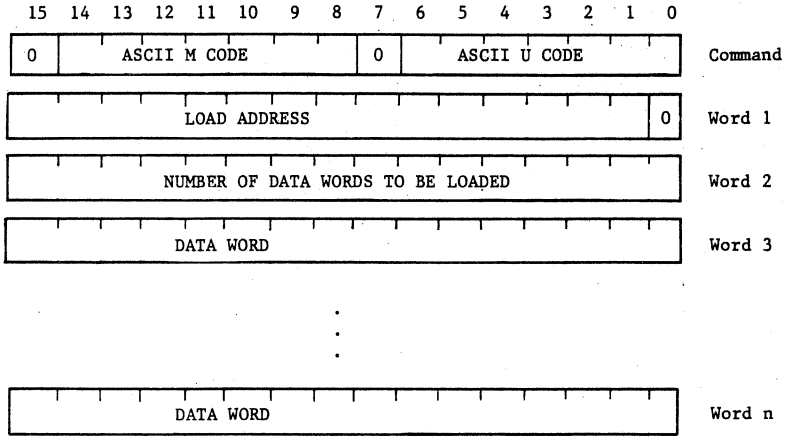


OCTAL	HEX
046523	4D53

Bits

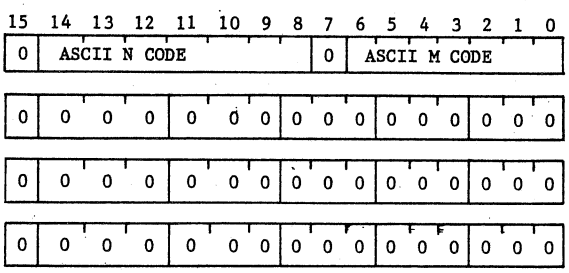
0	1	BANK NUMBER SELECTED
0	0	0
0	1	1
1	0	2
1	1	3

MU (H→G7)
MEMORY UPDATE
(Page 5-18)



OCTAL	HEX
046525	4D55

NM (G7→H)
NO MESSAGES READY
(Page 5-16)



OCTAL	HEX
047115	4E4D

TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

			OCTAL	HEX
NN (H→G7) ENABLE ERROR NUMBER (Page 5-68)	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 | ASCII N CODE | 0 | ASCII N CODE	Command header	047116	4E4E
	X X X X X X X X X X X X INDICATOR	Word 1		

BITS

3 2 1 0

1 0 0 0 Display error number on indicator 1
0 1 0 0 Display error number on indicator 2
0 0 1 0 Display error number on indicator 3
0 0 0 1 Display error number on indicator 4

If all bits are set to 1, then the error number is displayed on all four indicators.

NO (H→G7) NO OPERATION (Page 5-16)	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 | ASCII N CODE | 0 | ASCII O CODE	Command Header	047117	4E4F

NP (H→G7) ENABLE BOX DISPLAY (Page 5-66)	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 | ASCII N CODE | 0 | ASCII P CODE	Command header	047120	4E50
	X X X X X X X X X X X X INDICATOR	Word 1		

BITS

3 2 1 0

1 0 0 0 Display box on indicator 1
0 1 0 0 Display box on indicator 2
0 0 1 0 Display box on indicator 3
0 0 0 1 Display box on indicator 4

If all bits are set to 1, then the box is displayed on all four indicators. Any combination of indicators is permitted.

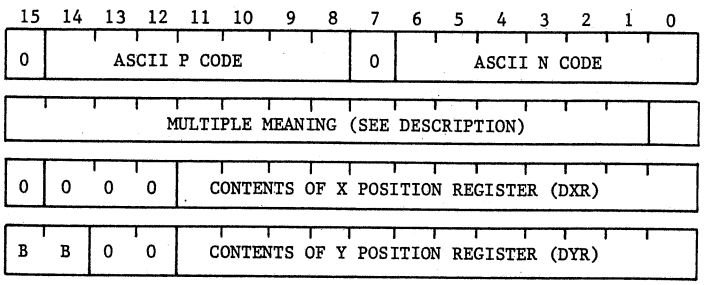
PL (H→G7) POLL GRAPHIC 7 FOR NEXT MESSAGE (Page 5-15)	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 | ASCII P | 0 | ASCII L	Command Header	050114	504C

PM (H→G7) CHANGE PHOTOPEN NO. 1 MODE (Page 5-52)	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 | ASCII P CODE | 0 | ASCII M CODE	Command header	050115	504D
	X X X X X X X X X X X INDICATOR | MODE	Word 1		

TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

PN (G7→H)
TURN PHOTOPEN NO. 1 STRIKE/SCAN
(Page 5-55)

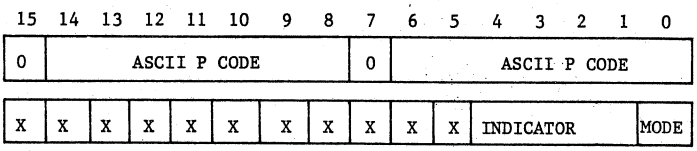


Command header — OCTAL 050116 — HEX 504E
Word 1
Word 2
Word 3

BITS

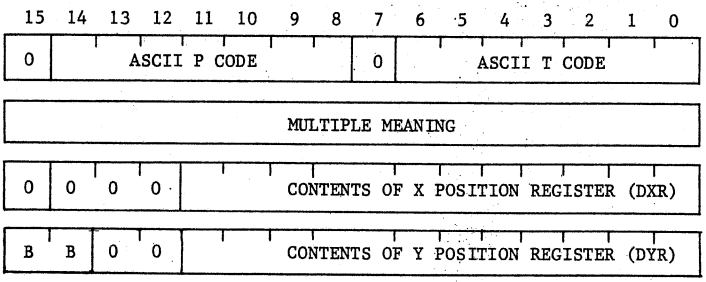
15	14	
0	0	Bank Ø
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

PP (H→G7)
CHANGE PHOTOPEN NO. 2 MODE
(Page 5-54)



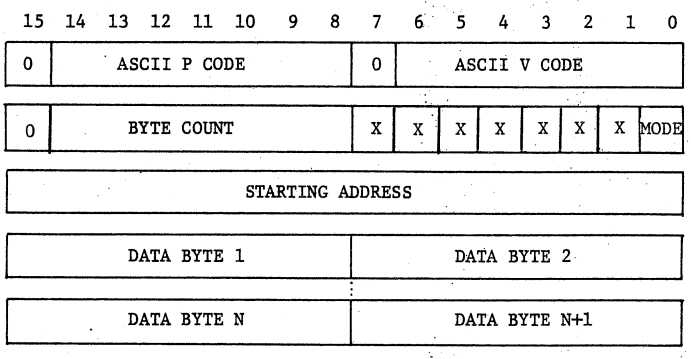
Command header — OCTAL 050120 — HEX 5050
Word 1

PT (G7→H)
RETURN PHOTOPEN NO. 2 STRIKE/SCAN
(Page 5-57)



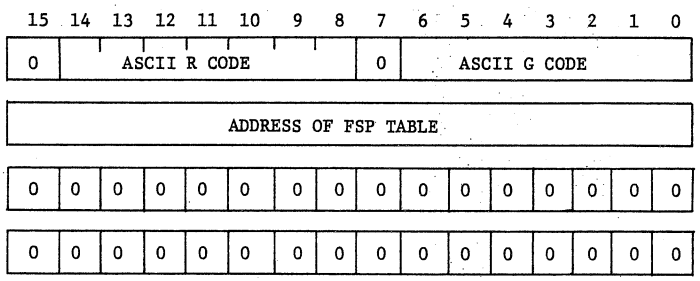
Command header — OCTAL 050124 — HEX 5054
Word 1
Word 2
Word 3

PV (H→G7)
PACKED VECTOR
(Page 5-72)



Command header — OCTAL 050126 — HEX 5056
Word 1
Word 2
Word 3
Word N

RG (G7→H)
RETURN FSP TABLE ADDRESS
(Page 5-63)



Command header — OCTAL 051107 — HEX 5247
Word 1
Word 2
Word 3

TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

RI (G7→H)
RETURN MESSAGE
(Page 5-24)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		OCTAL	HEX
0	ASCII R CODE							0	ASCII I CODE							Command header	051111	5249
STARTING ADDRESS															0	Word 1		
NUMBER OF WORDS TO BE TRANSFERRED																Word 2		
B	B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 3		

RK (G7→H)
FUNCTION KEYBOARD NO. 1
(Page 5-41)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		OCTAL	HEX
0	ASCII R CODE							0	ASCII K CODE							Command header	051113	524B
0	0	0	0	0	0	0	0	FUNCTION OR MATRIX KEY CODE								Word 1		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 2		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 3		

RL (G7→H)
FUNCTION KEYBOARD NO. 2
(Page 5-42)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		OCTAL	HEX
0	ASCII R CODE							0	ASCII L CODE							Command header	051114	524C
0	0	0	0	0	0	0	0	FUNCTION OR MATRIX KEY CODE								Word 1		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 2		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 3		

RO (G7→H)
RETURN OPTION
(Page 5-79)

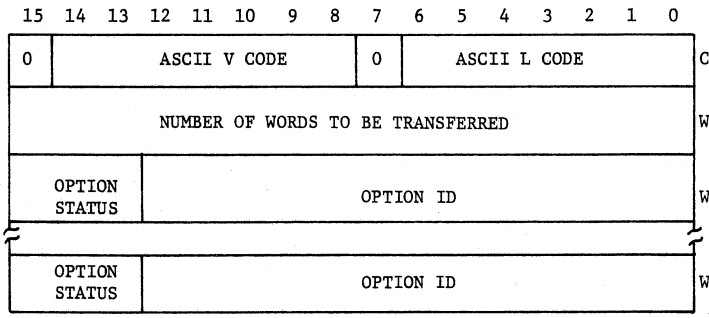
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		OCTAL	HEX
0	ASCII R CODE							0	ASCII O CODE							Command header	051117	524F
OPTION STATUS								12 BIT OPTION ID								Word 1		
OPTION INITIALIZATION ADDRESS																Word 2		
OPTION LAST ADDRESS + 2																Word 3		

Multiple option status return.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	ASCII R CODE							0	ASCII O CODE							Command header
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 1
NUMBER OF WORDS TO BE TRANSFERRED																Word 2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Word 3

TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

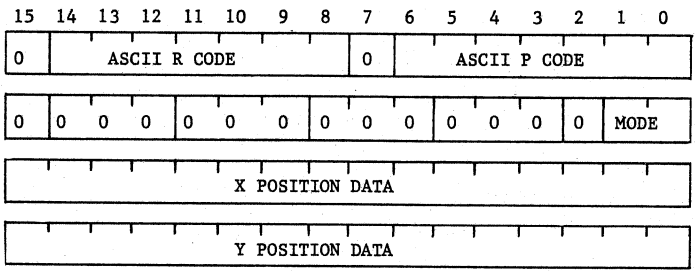


	OCTAL	HEX

(OCTAL) BIT

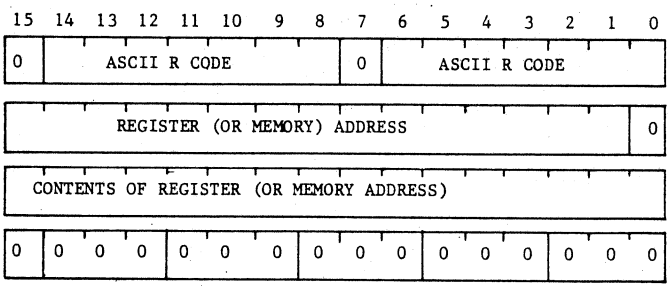
15 14 13 12	
0 0 0 0	Local detected option ,unloaded
0 0 0 1	Local checksum error ,unloaded
0 0 1 0	Local hardware not present ,unloaded
0 0 1 1	Local self test = NOGO ,unloaded
0 1 0 0	Local Self Test = GO ,loaded
0 1 0 1	Unfound option (for single RO message only)

RP (G7→H)
RETURN PED NO. 1
(Page 5-50)



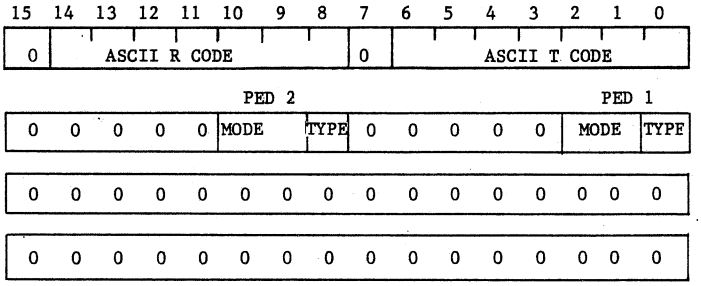
	OCTAL	HEX
	051120	5250

RR (G7→H)
RETURN REGISTER
(Page 5-27)



	OCTAL	HEX
	051122	5252

RT (G7→H)
RETURN PED STATUS
(Page 5-48)



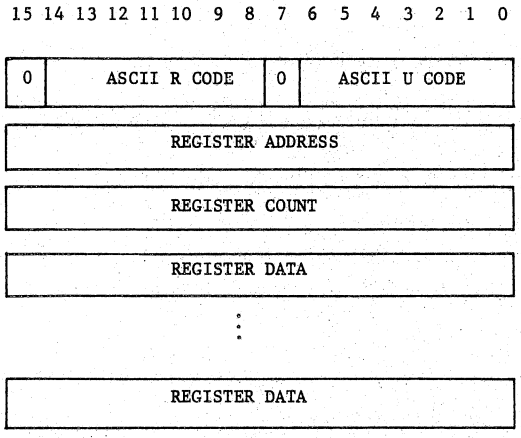
	OCTAL	HEX
	051124	5254

TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

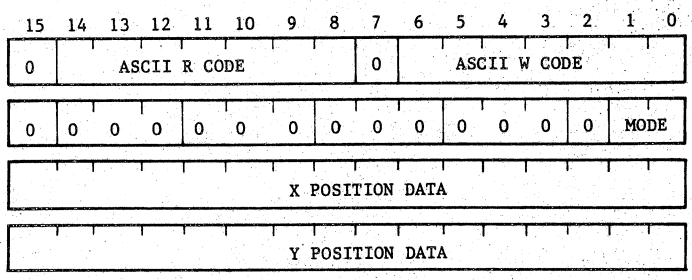
Bits	Value	Type PED
0 or 8	0	Trackball/forcestick
	1	Data tablet
1,2 or 9,10	00	Automatic tracking mode (data tablet only)
	01	Automatic mode
	10	Request mode
	11	Tracking mode

RU (H→G7)
REGISTER UPDATE
(Page 5-20)



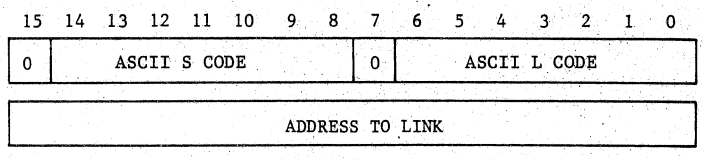
Command Header — OCTAL 051125 — HEX 5255
Word 1
Word 2
Word 3
Word n

RW (G7→H)
RETURN PED NO. 2
(Page 5-51)



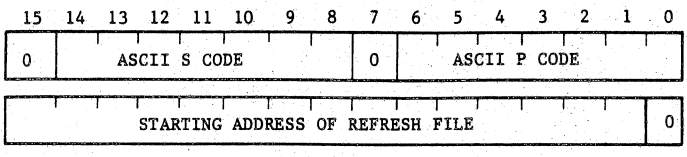
Command header — OCTAL 051127 — HEX 5257
Word 1
Word 2
Word 3

SL (H→G7)
LINK APPLICATION PROGRAM TO SCAN ROUTINE
(Page 5-54)



Command header — OCTAL 046123 — HEX 4C53
Word 1

SP (H→G7)
START PICTURE
(Page 5-21)

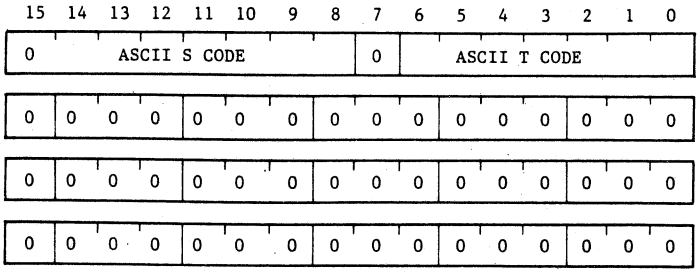


Command header — OCTAL 051520 — HEX 5350
Word 1

TABLE A-8

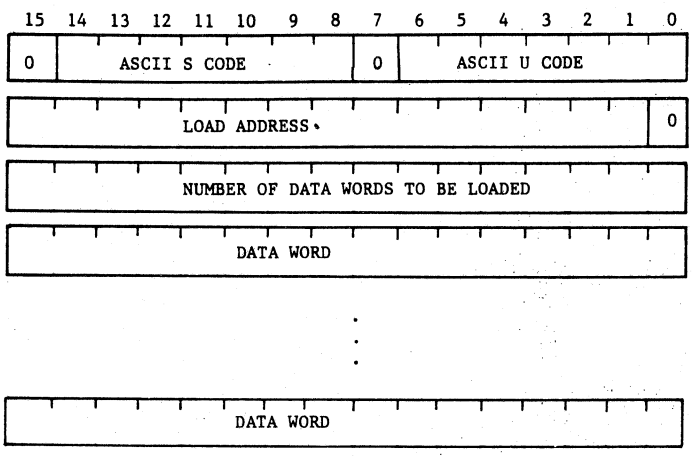
ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

ST (G7→H)
PHOTOPEN NO. 2 SWITCH ACTUATED
(Page 5-58)



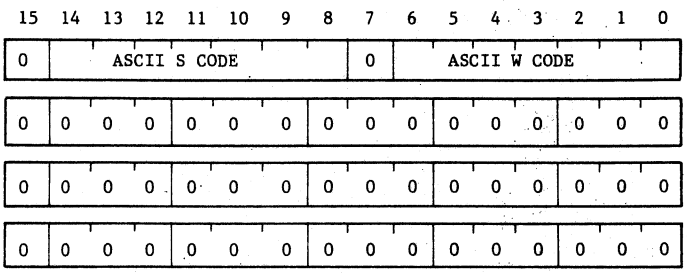
OCTAL	HEX
051524	5354

SU (H→G7)
SELECTIVE UPDATE
(Page 5-19)



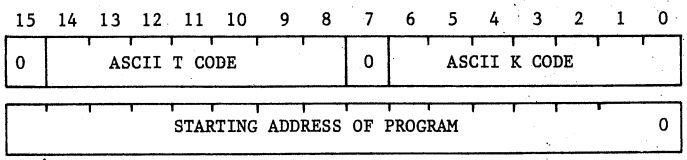
OCTAL	HEX
051525	5355

SW (G7→H)
PHOTOPEN NO. 1 SWITCH ACTUATED
(Page 5-58)



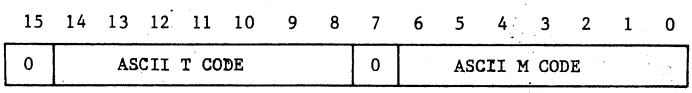
OCTAL	HEX
051527	5357

TK (H→G7)
TRANSFER CONTROL
(Page 5-23)



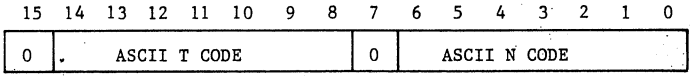
OCTAL	HEX
052113	544B

TM (H→G7)
ASSIGN DATA TABLET AS PED NO. 1
(Page 5-43)



OCTAL	HEX
052115	544D

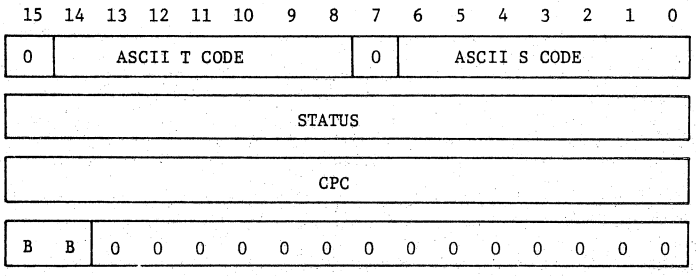
TN (H→G7)
ASSIGN DATA TABLET AS PED NO. 2
(Page 5-43)



OCTAL	HEX
052116	544E

TS (G7→H)

2-D/3-D Coordinate Converter Status (Page 5-83)

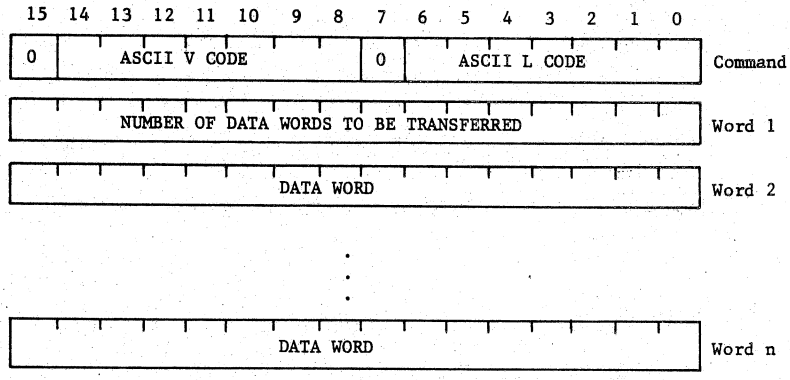


Bits 15 14	Bank Number
0 0	0
0 1	1
1 0	2
1 1	3

OCTAL	HEX
052123	5453

VL (G7→H)

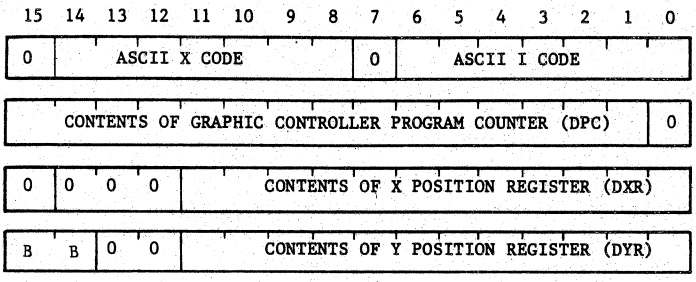
VARIABLE LENGTH (Page 5-25)



OCTAL	HEX
053114	564C

XI (G→H)

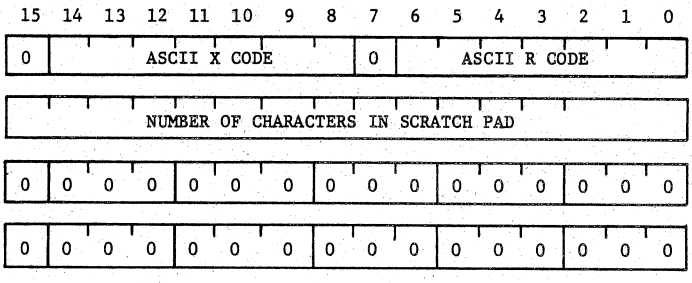
X OR Y POSITION OVERFLOW INTERRUPT (Page 5-32)



OCTAL	HEX
054111	5849

XR (G7→H)

SCRATCHPAD READY FOR ALPHANUMERIC KEYBOARD NO. 1 (Page 5-40)

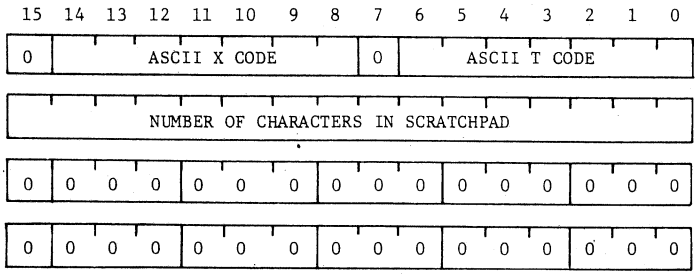


OCTAL	HEX
054122	5852

TABLE A-8

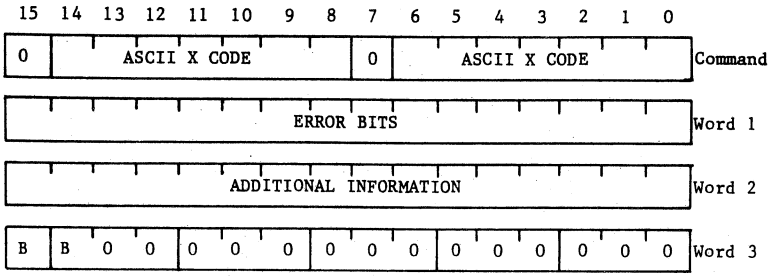
ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

XT (G7→H)
SCRATCHPAD READY FOR ALPHANUMERIC KEYBOARD NO. 2
(Page 5-41)



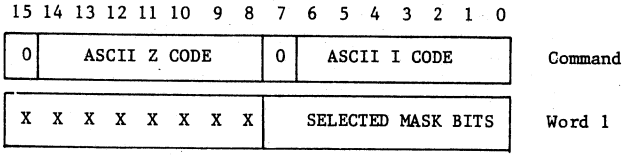
OCTAL 054124 HEX 5854

XX (G7→H)
ERROR STATUS
(Page 5-8)



OCTAL 054130 HEX 5858

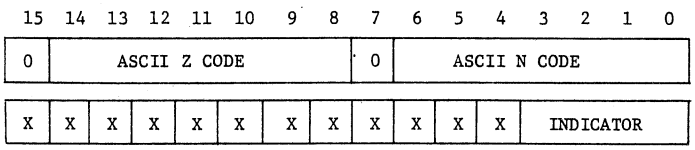
ZI (H→G7)
DISABLE SELECTED INTERRUPTS
(Page 5-30)



OCTAL 055111 HEX 5949

BIT	FUNCTION OR INTERRUPT CONDITION
0	Not used
1	Graphic controller halted
2	X or Y position overflow
3	Real time clock
4	PHOTOPEN 1 strike
5	Switch of PHOTOPEN 1 actuated
6	PHOTOPEN 2 strike
7	Switch of PHOTOPEN 2 actuated
8-15	These bits are ignored by GCP+

ZN (H→G7)
DISABLE ERROR NUMBER
(Page 5-69)



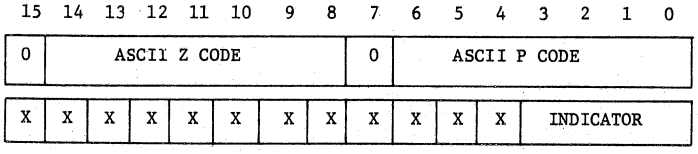
OCTAL 055116 HEX 594E

BITS 3 2 1 0	
1 0 0 0	Remove error number from indicator 1
0 1 0 0	Remove error number from indicator 2
0 0 1 0	Remove error number from indicator 3
0 0 0 1	Remove error number from indicator 4

TABLE A-8

ALPHABETICAL INDEX OF MESSAGES BETWEEN HOST AND GRAPHIC 7 (Cont)

ZP (H→G7)
DISABLE BOX DISPLAY
(Page 5-67)

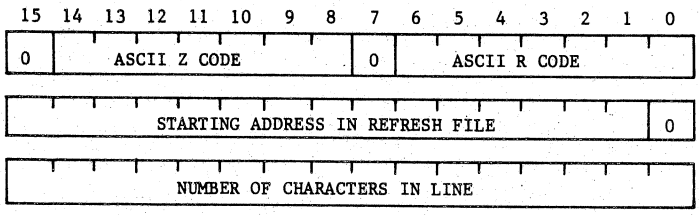


BITS

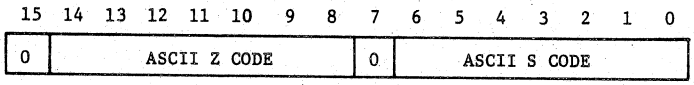
3 2 1 0

1 0 0 0 Remove box from indicator 1
0 1 0 0 Remove box from indicator 2
0 0 1 0 Remove box from indicator 3
0 0 0 1 Remove box from indicator 4

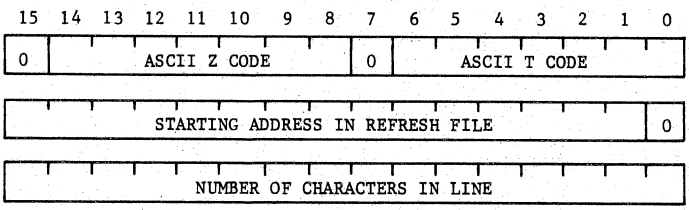
ZR (H→G7)
INITIALIZE SCRATCHPAD FOR ALPHANUMERIC KEYBOARD NO. 1
(Page 5-34)



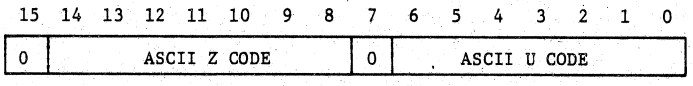
ZS (H→G7)
ZERO OUT SCRATCHPAD NO. 1
(Page 5-36)



ZT (H→G7)
INITIALIZE SCRATCHPAD FOR ALPHANUMERIC KEYBOARD NO. 2
(Page 5-35)



ZU (H→G7)
ZERO OUT SCRATCHPAD NO. 2
(Page 5-36)



	OCTAL	HEX
ZP	055120	5950
ZR	055122	5952
ZS	055123	5953
ZT	055124	5954
ZU	055125	5955

TABLE A-9

CHARACTER GENERATOR CODE ASSIGNMENTS

b_6 b_5 b_4 → / BITS					$0\ 0\ 0$	$0\ 0\ 1$	$0\ 1\ 0$	$0\ 1\ 1$	$1\ 0\ 0$	$1\ 0\ 1$	$1\ 1\ 0$	$1\ 1\ 1$
b_3	b_2	b_1	b_0	col / row	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL		SP	0	@	P	`	p
0	0	0	1	1			!	1	A	Q	a	q
0	0	1	0	2			"	2	B	R	b	r
0	0	1	1	3			#	3	C	S	c	s
0	1	0	0	4			$	4	D	T	d	t
0	1	0	1	5			%	5	E	U	e	u
0	1	1	0	6			&	6	F	V	f	v
0	1	1	1	7			'	7	G	W	g	w
1	0	0	0	8			(	8	H	X	h	x
1	0	0	1	9			)	9	I	Y	i	y
1	0	1	0	10			*	:	J	Z	j	z
1	0	1	1	11			+	;	K	[	k	{
1	1	0	0	12			,	<	L	\	l	|
1	1	0	1	13			-	=	M	]	m	}
1	1	1	0	14	SHIFT OUT		.	>	N	^	n	~
1	1	1	1	15	SHIFT IN		/	?	O	_	o	

GA-77-419-17

TABLE A-10

MULTIPORT SERIAL INTERFACE PORT ASSIGNMENTS

Card	Connector	Port Designation	Assignment
Multiport Serial Interface Card No. 1	J2 (RS-232C) or J3	1	Host Computer
	J4	2	Spare
	J5	3	Alphanumeric/ Function Keyboard No. 1
	J6	4	PED No. 1
Multiport Serial Interface Card No. 2	J2 (RS-232C) or J3	5	Hard Copy Unit
	J4	6	Spare
	J5	7	Alphanumeric/ Function Keyboard No. 2
	J6	8	PED No. 2

NOTE

A paper tape reader may be connected to multiport serial interface port 1, 2, or 3 or to the serial interface port on the ROM and status logic card.

TABLE A-11

STANDARD TRANSFER TABLE

Memory Address (octal)	Information or GCP Routine
157700	GCP+ date (month and year)
157702	GCP+ date (day of month)
157704	GCP+ release number
157706	Number of GCP+ field changes
157710	ZERO (display maintenance routine)
157720	PLUS (display maintenance routine)
157730	MINUS (display maintenance routine)
157740	LOADER (calls absolute loader routine)
157750	DEBUG (calls command processing routine of local operating mode)
157760	SYSTEM (transfers to system operating mode)
157770	TEST (calls verification test pattern)

NOTE

In the local operating mode, information can be examined or control can be transferred using local mode commands. In the system operating mode, host-to-GRAPHIC 7 TK messages can be used to transfer control.

TABLE A-12

RECOMMENDED TABULAR INCREMENTS

Character Size	Nominal Height (12 in. x 12 in. display) (inches)	Recommended Text Increment (Horizontal Increment Between Characters) (octal)	Recommended Line Feed Increment (Vertical Increment Between Lines) (octal)
0	0.125	12	17
1	0.187	17	26
2	0.250	24	36
3	0.375	36	55

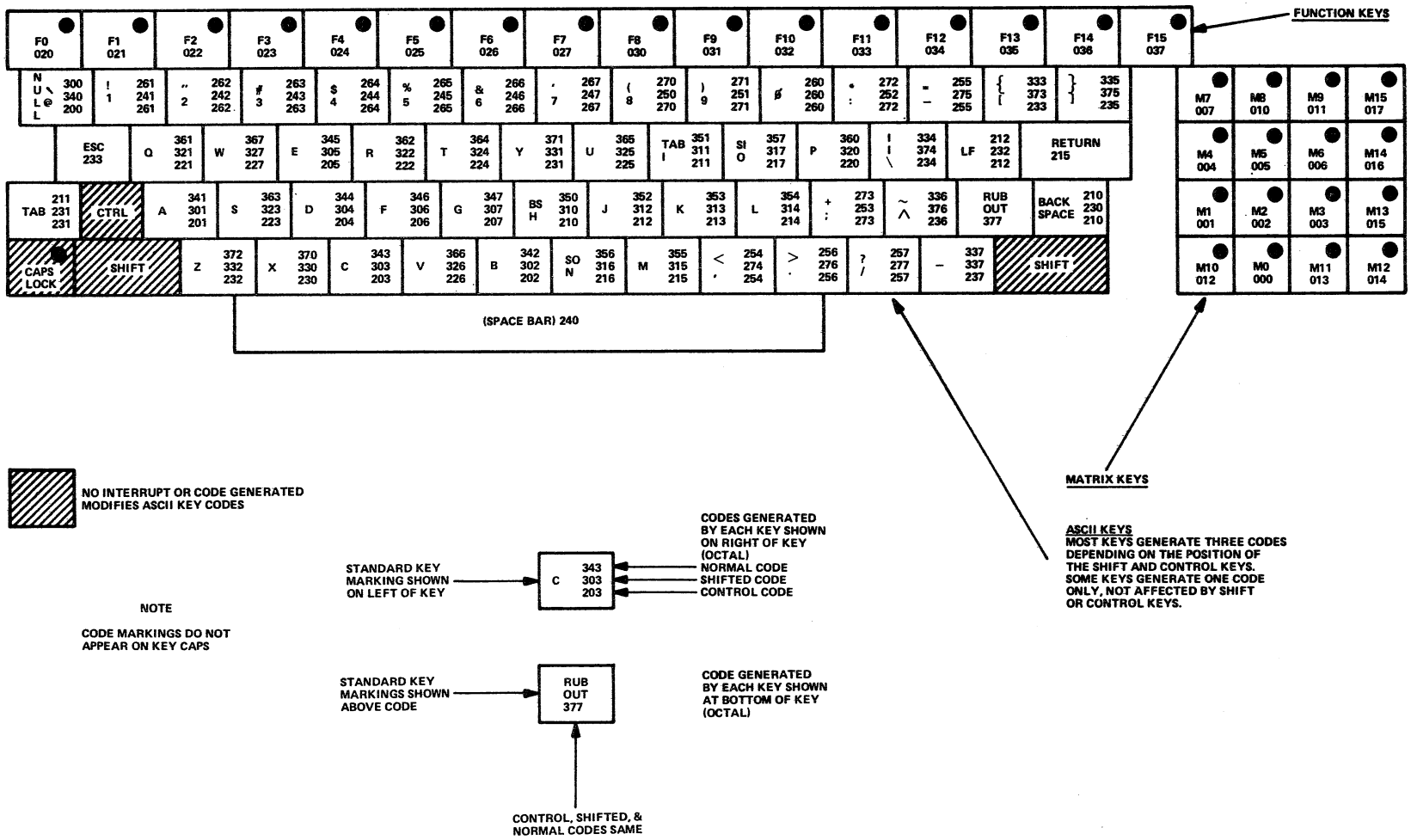


Figure A-2 Model 5784 Keyboard Layout and Code Assignments.

TABLE A-13

7-BIT ASCII CODE

Char.	Octal Code	Dec.	Oct. SHF 10=	Dec.	Char.	Octal Code	Dec.	Oct. SHF 10=	Dec.
NUL	000	0	0	0	US	037	31	17400	7936
SOH	001	1	400	256	SP	040	32	20000	8192
STX	002	2	1000	512	!	041	33	20400	8448
ETX	003	3	1400	768	"	042	34	21000	8960
EOT	004	4	2000	1024	#	043	35	21400	8960
ENQ	005	5	2400	1280	$	044	36	22000	9216
ACK	006	6	3000	1536	%	045	37	22400	9472
BEL	007	7	3400	1792	&	046	38	23000	9728
BS	010	8	4000	2048	'	047	39	23400	9984
HT	011	9	4400	2304	(	050	40	24000	10240
LF	012	10	5000	2560	)	051	41	24400	10496
VT	013	11	5400	2816	*	052	42	25000	10752
FF	014	12	6000	3072	+	053	43	25400	11008
CR	015	13	6400	3328	,	054	44	26000	11264
SO	016	14	7000	3584	-	055	45	26400	11520
SI	017	15	7400	3840	.	056	46	27000	11776
DLE	020	16	10000	4096	/	057	47	27400	12032
DC1	021	17	10400	4352	0	060	48	30000	12288
DC2	022	18	11000	4608	1	061	49	30400	12544
DC3	023	19	11400	4864	2	062	50	31000	12800
DC4	024	20	12000	5120	3	063	51	31400	13056
NAK	025	21	12400	5376	4	064	52	32000	13312
SYN	026	22	13000	5632	5	065	53	32400	13568
ETB	027	23	13400	5888	6	066	54	33000	13824
CAN	030	24	14000	6144	7	067	55	33400	14080
EM	031	25	14400	6400	8	070	56	34000	14336
SUB	032	26	15000	6656	9	071	57	34400	14592
ESC	033	27	15400	6912	:	072	58	35000	14848
FS	034	28	16000	7168	;	073	59	35400	15104
GS	035	29	16400	7424	<	074	60	36000	15360
RS	036	30	17000	7680	=	075	61	36400	15616

TABLE A-13

7-BIT ASCII CODE (Cont)

Char.	Octal Code	Dec.	Oct. SHF 10=	Dec.	Char.	Octal Code	Dec.	Oct. SHF 10=	Dec.
>	076	62	37000	15872	]	135	93	56400	23808
?	077	63	37400	16128	^	136	94	57000	24064
@	100	64	40000	16384	—	137	95	57400	24320
A	101	65	40400	16640	`	140	96	60000	24576
B	102	66	41000	16896	a	141	97	60400	24832
C	103	67	41400	17152	b	142	98	61000	25088
D	104	68	42000	17408	c	143	99	61400	25344
E	105	69	42400	17664	d	144	100	62000	25600
F	106	70	43000	17920	e	145	101	62400	25856
G	107	71	43400	18176	f	146	102	63000	26112
H	110	72	44000	18432	g	147	103	63400	26368
I	111	73	44400	18688	h	150	104	64000	26624
J	112	74	45000	18944	i	151	105	64400	26880
K	113	75	45400	19200	j	153	106	65000	27136
L	114	76	46000	19456	k	153	107	65400	27392
M	115	77	46400	19712	l	154	108	66000	27648
N	116	78	47000	19968	m	155	109	66400	27904
O	117	79	47400	20224	n	156	110	67000	28160
P	120	80	50000	20480	o	157	111	67400	28416
Q	121	81	50400	20736	p	160	112	70000	28672
R	122	82	51000	20992	q	161	113	70400	28928
S	123	83	51400	21248	r	162	114	71000	29184
T	124	84	52000	21504	s	163	115	71400	29440
U	125	85	52400	21760	t	164	116	72000	29696
V	126	86	53000	22016	u	165	117	72400	29952
W	127	87	53400	22272	v	166	118	73000	30208
X	130	88	54000	22528	w	167	119	73400	30464
Y	131	89	54400	22784	x	170	120	74000	30720
Z	132	90	55000	23040	y	171	121	74400	30976
[	133	91	55400	23296	z	172	122	75000	31232
\	134	92	56000	23552	{	173	123	75400	31488

TABLE A-13

7-BIT ASCII CODE (Cont)

Char.	Octal Code	Dec.	Oct. SHF 10=	Dec.	Char.	Octal Code	Dec.	Oct. SHF 10=	Dec.
|	174	124	76000	31744	~	176	126	77000	32256
}	175	125	76400	32000	DEL	177	127	77400	32512

TABLE A-14

GRAPHIC 7 REGISTERS

Register	Memory Address	I/O READ	I/O WRITE
SENSE WORD (SENS)	177660	Yes	No
MASK REGISTER (MKR)	177662	Yes	Yes
STACK POINTER (DSP)	165000	Yes	*
GENERAL PURPOSE REGISTER (DRO)	165002	Yes	*
GENERAL PURPOSE REGISTER (DR1)	165004	Yes	*
PROGRAM COUNTER (DPC)	165006	Yes	Yes
DISPLAY INSTRUCTION REGISTER (DIR)	165010	Yes	*
TEXT INCREMENT REGISTER (DTI)	165012	Yes	*
DISPLAY PARAMETER REGISTER (DPR)	165014	Yes	*
PAGE REGISTER (PGR)	165014	**	Yes
DISPLAY Z REGISTER (DZR)	165016	Yes	*
X REGISTER (DXR)	165020	Yes	*
Y REGISTER (DYR)	165022	Yes	*
CHARACTER REGISTER (DCR)	165024	Yes	*
X CONIC REGISTER (KXR) (optional)	165026	Yes	*
Y CONIC REGISTER (KYR) (optional)	165030	Yes	*
GENERAL PURPOSE REGISTER (DR2)	165032	Yes	*
GENERAL PURPOSE REGISTER (DR3)	165034	Yes	*
#FUNCTION CONTROL CONTINUE (FUNC)	165036	No	Yes
FUNCTION CONTROL STOP (FUNS)	165040	No	Yes
ERROR REGISTER (ERR)	165312	Yes	Yes
PHOTOPEN ITEM NO. (PIN)	165314	Yes	Yes

*These registers are written by refresh commands and read by programmed data transfers.

**The 2 bit Page Register is read as bits 14 and 15 of the DPR.

#A write to FUNC register while the graphic controller is running will cause an error trap through address 4 (error trap).

TABLE A-14

GRAPHIC 7 REGISTERS (Cont)

	Register	Memory Address	I/O READ	I/O WRITE	TRAP ADDRESS
SERIAL INTERFACE (SINGLE PORT)	RECEIVE STATUS (RSR)	177560	Yes	Yes	60
	REC. DATA BUFFER (RDB)	177562	Yes	No	
	TRANSMIT STATUS (TSR)	177564	Yes	Yes	64
	TRANS. DATA BUFFER (TDB)	177566	No	Yes	
SERIAL INTERFACE (4 PORTS)	RECEIVE STATUS (RSR)	176500	Yes	Yes	300
	REC. DATA BUFFER (RDB)	176502	Yes	No	
	PARAMETER CONTROL (PCR)		No	Yes	
	TRANSMIT STATUS (TSR)	176504	Yes	Yes	304
	TRANS. DATA BUFFER (TDB)	176506	No	Yes	
	RECEIVE STATUS (RSR)	176510	Yes	Yes	310
	REC. DATA BUFFER (RDB)	176512	Yes	No	
	TRANSMIT STATUS (TSR)	176514	Yes	Yes	314
	TRANS. DATA BUFFER (TDB)	176516	No	Yes	
	RECEIVE STATUS (RSR)	176520	Yes	Yes	320
	REC. DATA BUFFER (RDB)	176522	Yes	No	
	TRANSMIT STATUS (TSR)	176524	Yes	Yes	324
	TRANS. DATA BUFFER (TDB)	176526	No	Yes	
	RECEIVE STATUS (RSR)	176530	Yes	Yes	330
	REC. DATA BUFFER (RDB)	176532	Yes	No	
	TRANSMIT STATUS (TSR)	176534	Yes	Yes	334
	TRANS. DATA BUFFER (TDB)	176536	No	Yes	
PARALLEL INTERFACE	WORD COUNT (WCR)	172410	Yes	Yes	
	MEMORY ADDRESS (MAR)	172412	Yes	Yes	
	STATUS (STR)	172414	Yes	Yes	
	INPUT DATA (IDR)	172416	No	Yes	120
	OUTPUT DATA (ODR)		Yes	No	124

NOTE

Each of the following devices have associated logical unit numbers as indicated:

DEVICE	LOGICAL UNIT NO.
Host (serial)	1
Spare	2
Alphanumeric/Function Keyboard #1	3
PED #1	4
Hard Copy Unit	5
Spare	6
Alphanumeric Function Keybaord #2	7
PED #2	8

It is required that these devices be interfaced to the port (on the Serial Multiport board) that has the associated logical unit number. These numbers are assigned and strapped to the ports at delivery. For certain installations, more than 1 Multiport board may be required. Each Multiport board has 4 ports which can be strapped to any logical unit number.

TABLE A-15

GRAPHIC 7 INSTRUCTION TIMING

Instruction	Time (μsec)
HALT	0.3(1)
JUMP	2.7
JRMP	2.7
JMPZ	3.3 (1.8 if DRØ ≠ Ø)
JPRZ	3.0 (1.8 if DOØ = Ø)
JMPM	3.9
CALL	3.3
CALR	3.6
RTRN	2.7
IZPR	1.2
LINK	3.0(1)
SAVD	DRO:2.4, DR1:2.7, DR2 or DR3:3.0
RESD	DRO:3.0, DR1:3.3, DR2 or DR3:3.6
ADDI	DRO:2.4, DR1:2.7, DR2 or DR3:3.0
JMPR	1.5
LDRI	1.8
LDDI	μD0:2.4, μD1:2.7, μD2 or μD3:3.0
LDSP	2.4
WATE	0.3 to 0.6(2)

Instruction	Time (μsec)
DRKN	$9.0 + 2(R_t)^{(3)} + 4(Q_t)^{(4)(9)}$
LDDZ	2.7 (0.9 if Display Select unchanged)
LDDP	2.4
LDXA	0.9
LDXR	0.9
DRXA	$0.7 + V_t^{(5)(9)}$
DRXR	$0.7 + V_t^{(5)(9)}$
DRYA	$0.7 + V_t^{(5)(9)}$
DRYR	$0.7 + V_t^{(5)(9)}$
MVXA	$0.4 + P_t^{(6)(9)}$
MVXR	$0.4 + P_t^{(6)(9)}$
MVYA	$0.4 + P_t^{(6)(9)}$
MVYR	$0.4 + P_t^{(6)(9)}$
DRSR	2.4
MVSR	2.4
PPLR	2.1
LDTI	0.9
TEXT	$2.3 + 2\ C_t^{(8)(9)}$
CHAR	$0.7^{(10)} + C_t^{(8)(9)}$ (2.2 + C_t if blinked Sincle character)

Notes:

1. Time required until display halted and interrupt process started.
2. Time required to restart after frame sync pulse.
3. $R_t = 40\ ^{L}RAD/^{L}MAX^*$ where ^{L}RAD is the horizontal radius of the conic.
4. $Q_t = 40\ ^{L}QUAD/^{L}MAX^*$ where $^{L}QUAD$ is the straight line distance from the quadrant start point to the end.
5. $V_t = 40\ ^{L}VECT/^{L}MAX^*$ where $^{L}VECT$ is the length of the vector.
6. $P_t = 25\ ^{L}POS/^{L}MAX^*$ where ^{L}POS is the length of the position move.
7. L = Length of position move or vector.
8. C_t = KxN, where N=number of strokes. For example: The letter A is 14 strokes.

Size	Ratio	Time/Stroke	
0	1	0.15 μs	
1	1.5	0.225 μs	Smallest size = 1/8"
2	2	0.30 μs	
3	3	0.45 μs	

(The average total time required to draw a small character is 3.4 μsec.)

9. Instruction time should be rounded up to the next higher 0.3 μsec increment.
10. 2.2 μsec if character is blinked.
11. A dit is a display coordinate least significant bit length.

*L_{MAX} = the maximum on-axis dimension of the display grid.

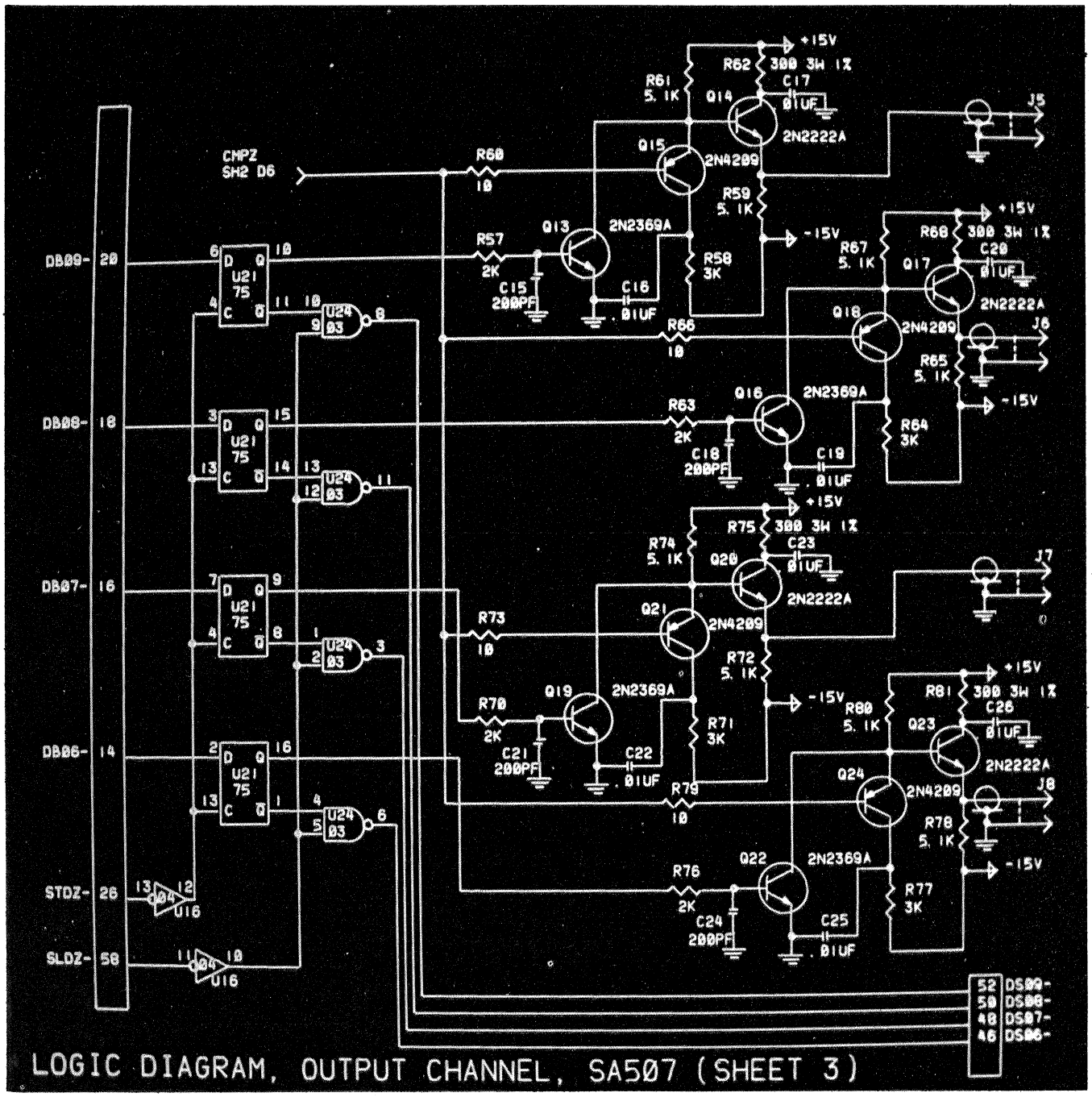


Figure A-3 Schematic (9.5 msec) Display Image.

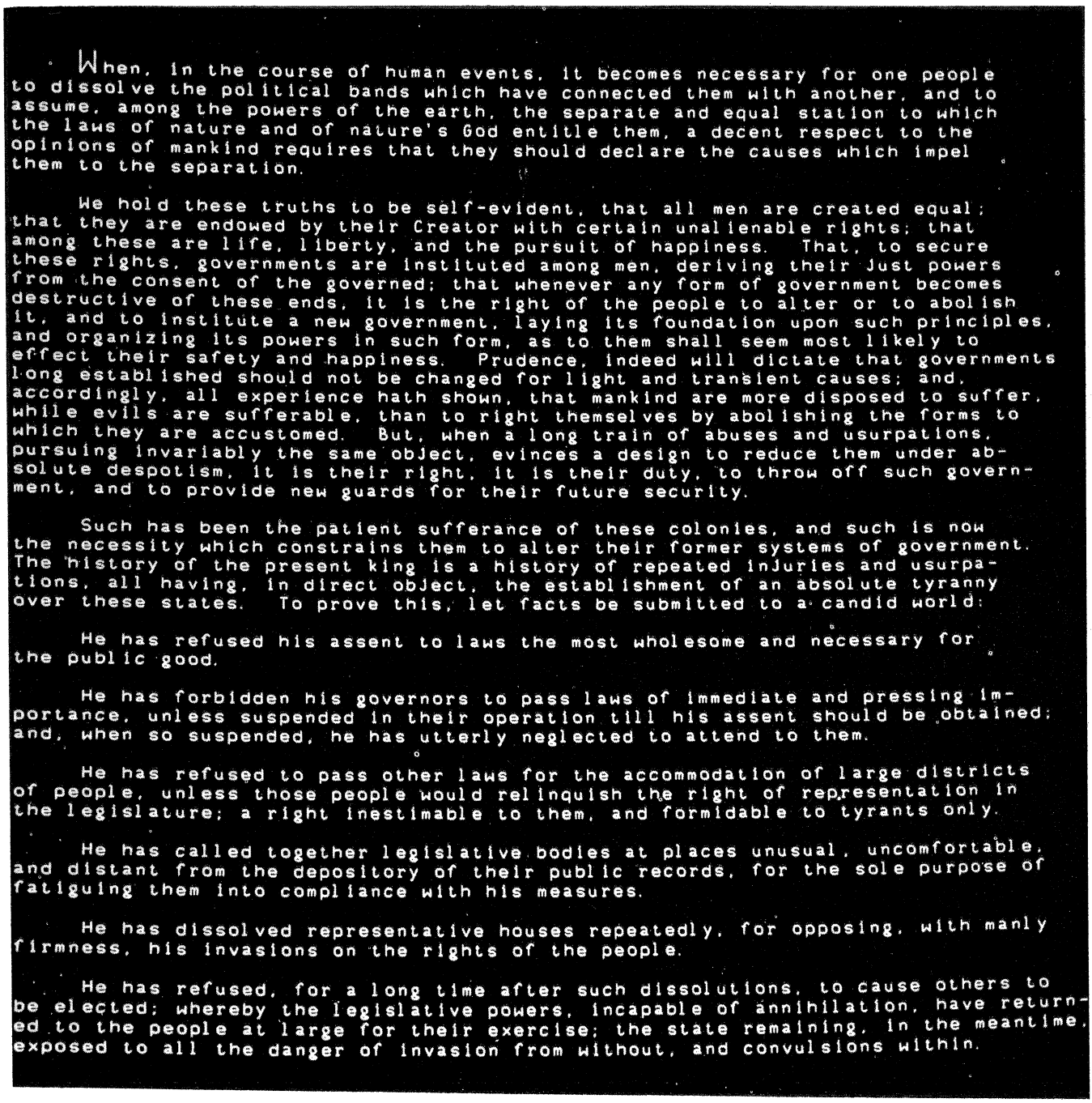


Figure A-4 Text (11.5 msec) Display Image.

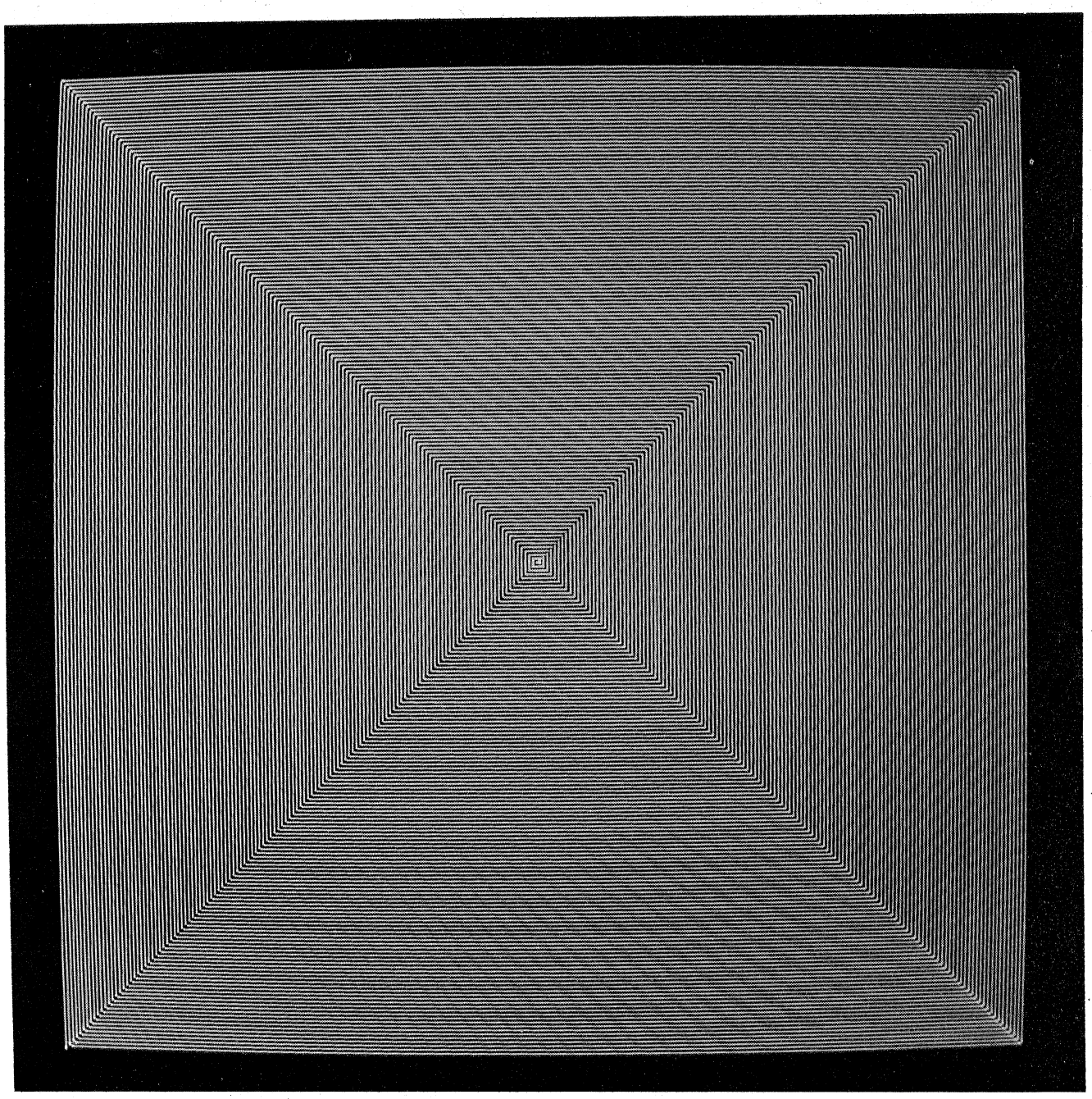


Figure A-5 Vectors (11.0 msec) Display Image.

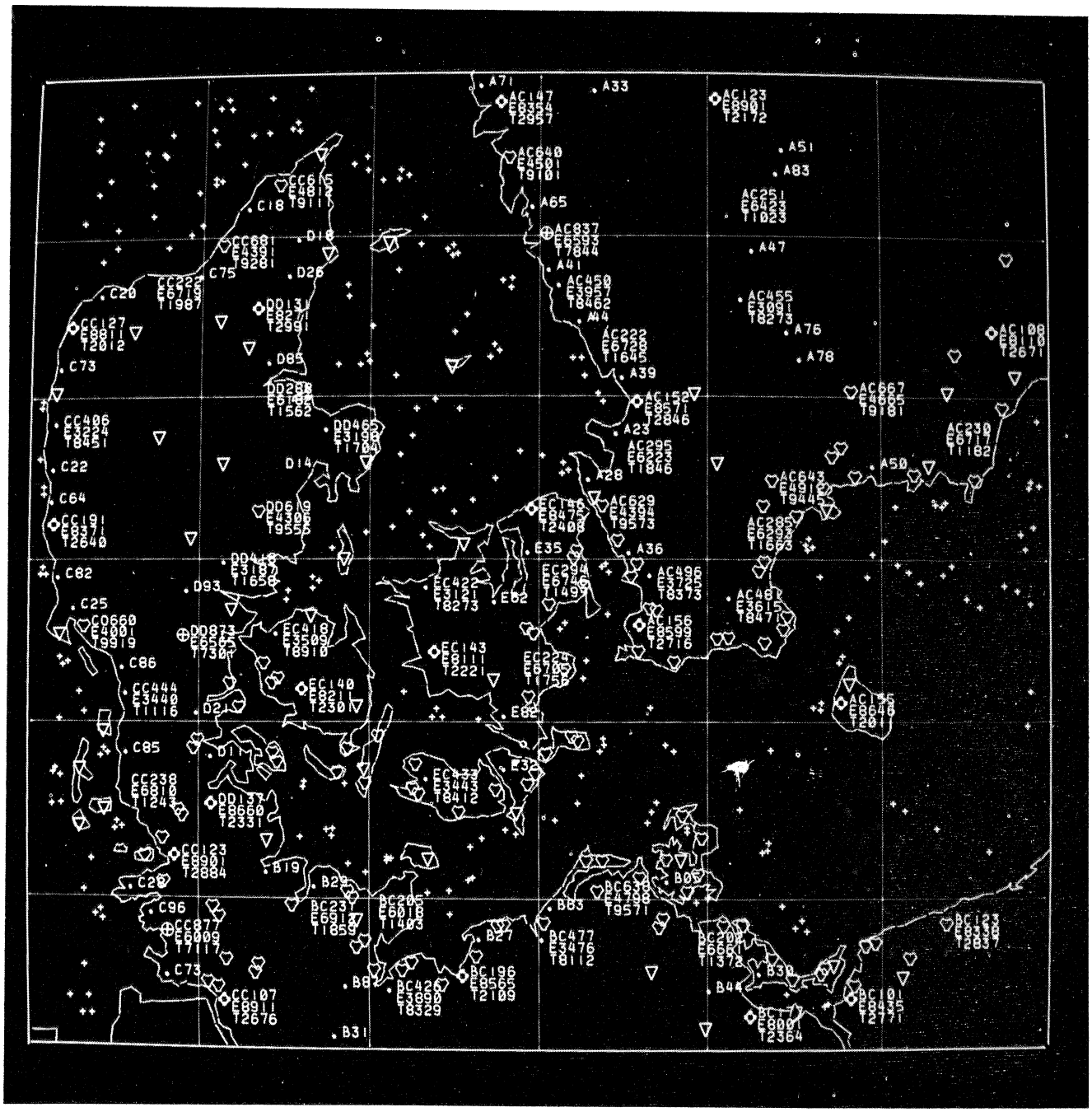


Figure A-6 Map (15.0 msec) Display Image.

APPENDIX B
GRAPHIC 7 MACRO DESCRIPTIONS

B.1 GENERAL

This appendix describes the GRAPHIC 7 display macros that have been developed by Sanders for programmers who use the MACRO-11 assembly language for their applications programs. Table B-1 lists the macros in alphabetical order. Table B-2 describes the macros and defines the arguments accepted by each. Table B-3 shows the program structures for two typical refresh files that use the macros.

The following conventions are used in table B-2 to define macro arguments.

1. All numbers are octal.

2. Lower case letters indicate variable arguments. With the exception of the following, each letter represents a single octal digit (leading zeros are not required for arguments shorter than the specified field):

   a. "a" and "b" each represent a single ASCII character.

   b. "arg" represents a specific argument identified in the macro description.

   c. "label" represents a label assigned by the applications programmer or an absolute or relative value.

   d. "character string" represents any string of ASCII characters as determined by the applications programmer.

3. Upper case letters indicate specific arguments as discussed in the macro descriptions.

NOTES

1. Standard graphic controller instructions referenced in this appendix are described in detail in Section 3. Coordinate converter instructions (CCOM, CCOS, CCXT, CCYT, and CSIN) are described in Sanders publication H-78-0061.

2. All register mnemonics listed in table A-6 are defined in the GRAPHIC 7 macros and may be used as arguments for MACRO-11 instructions.

3. The following labels are used within the GRAPHIC 7 macros and must not be duplicated in any user written program that employs these macros:

ARG.OK	TI.
CH.	TMP$
DI$LØP	TXT.
DPR.	YINC.
DZR.	

TABLE B-1

GRAPHIC 7 DISPLAY MACROS

Macro	Function
ADDI	Add to display register immediate
ADR	Absolute draw
AMV	Absolute move
CALL	Call subroutine
CALR	Call relative
CCOM	Coordinate converter command
CCOS	Load coordinate converter cosine register
CCXT	Load coordinate converter X translation register
CCYT	Load coordinate converter Y translation register
CHAR	Draw single character
CIRCLE	Draw circle
COLOR	Select color
CSIN	Load coordinate converter sine register
DISEND	Display end
DISINT	Display initialize
DRKY	Draw conic Y
DRSR	Draw short relative

TABLE B-1

GRAPHIC 7 DISPLAY MACROS (Cont)

Macro	Function
DRXA	Draw X absolute
DRXR	Draw X relative
DRYA	Draw Y absolute
DRYR	Draw Y relative
ENTR	Provide subroutine entry point
HREF	Halt refresh
IZPR	Initialize
JMPM	Jump and mark
JMPR	Jump short relative
JMPZ	Jump if display register 0 contents ≠ 0
JPRZ	Jump relative if display register 0 contents ≠ 0
JRMP	Jump relative
JUMP	Jump
LDDI	Load display register immediate
LDDP	Load display parameter register
LDDZ	Load display Z register
LDKX	Load conic X register
LDRI	Load device register immediate
LDSP	Load stack pointer
LDTI	Load text increment register
LDXA	Load X absolute
LDXR	Load X relative
LINK	Synchronized linkage
MVSR	Move short relative
MVXA	Move X absolute
MVXR	Move X relative
MVYA	Move Y absolute
MVYR	Move Y relative
NEWL	New line
NEWLR	New line relative
NOOP	No operation
PPLR	Point plot relative
RDR	Relative draw

TABLE B-1

GRAPHIC 7 DISPLAY MACROS (Cont)

Macro	Function
RESD	Restore display register
RLINK	Relink
RMV	Relative move
RTRN	Return
SAVD	Save display register
SETLF	Set line feed
SETMRG	Set margin
SETTI	Set text increment
TEXT	Draw tabular character string
TXT	Draw two tabular characters
WATE	Wait

TABLE B-2

DETAILED MACRO DESCRIPTIONS

Macro Call	Description
ADDI r,nnnnnn	Inserts an ADDI (add to display register immediate) instruction into the refresh file. Argument "r", which must be "0", "1", "2", or "3", specifies one of the general purpose registers (DR0 through DR3) of the graphic controller. Argument "nnnnnn" specifies the value (-100000 to 77777) to be added to the register.
ADR xxxx,yyyy	Causes an absolute draw to position X,Y by inserting two instructions into the refresh file. The first is an LDXA (load X absolute) instruction with the X coordinate defined by argument "xxxx". The second is a DRYA (draw Y absolute) instruction with the Y coordinate defined by argument "yyyy". Both arguments can vary from -2000 to 1777.
AMV xxxx,yyyy	Causes an absolute move to position X,Y by inserting two instructions into the refresh file. The first is an LDXA (load X absolute) instruction with the X coordinate defined by argument "xxxx". The second is a MVYA (move Y absolute) instruction with the Y coordinate defined by argument "yyyy". Both arguments can vary from -2000 to 1777.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Function
CALL label	Inserts a CALL (call subroutine) instruction into the refresh file with the subroutine address defined by argument "label". Argument "label" may define any even location in memory.
CALR label	Inserts a CALR (call relative) instruction into the refresh file with the subroutine address specified by argument "label". Argument "label" may define any even location in memory.
CCOM arg	Inserts a CCOM (coordinate converter command) instruction into the refresh file to turn the 2-D coordinate converter option on or off as specified by argument "arg". Argument "arg" must be either "ON" or "OFF". (Octal values for "ON" and "OFF" are 2 and 0, respectively.
CCOS nnnn	Inserts a CCOS (load coordinate converter cosine register) instruction into the refresh file. The CCOS instruction contains the cosine of the angle through which display elements are to be rotated by the 2-D coordinate converter option. Argument "nnnn" specifies the cosine value which may range from -2000 (-1.000_{10}) to 1777 ($+0.9990234375_{10}$).
CCXT nnnn	Inserts a CCXT (load coordinate converter X translation register) instruction into the refresh file. The CCXT instruction contains the distance along the X axis that display elements are to be translated by the 2-D coordinate converter option. Argument "nnnn" specifies the translation distance in terms of coordinates. This distance may vary from -2000 to 1777.
CCYT nnnn	Inserts a CCYT (load coordinate converter Y translation register) instruction into the refresh file. The CCYT instruction contains the distance along the Y axis that display elements are to be translated by the 2-D coordinate converter option. Argument "nnnn" specifies the translation distance in terms of coordinates. This distance may vary from -2000 to 1777.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
CHAR a,B	Inserts a CHAR (draw single character) into the refresh file. Argument "a" specifies the ASCII character to be drawn (if the character to be drawn is a space, it must be enclosed in angle brackets: < >). If argument "B" is absent, the character will be displayed steadily; if argument "B" is present, the character will blink. No tabular text increment move is made following the drawing of the character.
CIRCLE nnnn	Inserts a DRKY (draw conic Y) instruction into the refresh file. If a CIRLCE macro is not preceded by an LDKX (load conic X) macro, a circle will be drawn. Argument "nnnn" specifies the radius of the circle which may vary from -2000 to 1777. If an LDKX macro precedes a CIRCLE macro, the CIRCLE macro has the same effect as a DRKY macro that uses "3" for the "q" argument.
COLOR <arg1,arg2>	Inserts an LDDI instruction into refresh to select one of four colors and to select the CRT display indicator. (LDDI is a two word instruction.)

Argument	Description	Octal Value
CRT1	Change color display 1	4000
CRT2	Change color display 2	2000
CRT3	Change color display 3	1000
CRT4	Change color display 4	400
ALL	Change color all displays	7400
RED	Select RED	0
ORANGE	Select ORANGE	2
YELLOW	Select YELLOW	12
GREEN	Select GREEN	52

CSIN nnnn	Inserts a CSIN (load coordinate converter sine register) instruction into the refresh file. The CSIN instruction contains the sine of the angle through which display elements are

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
CSIN nnnn (Cont)	to be rotated by the 2-D coordinate converter option. Argument "nnnn" specifies the sine value which may range from -2000 (-1000_{10}) to 1777 ($+0.9990234375_{10}$).
DISEND	Used in conjunction with DISINT macro to terminate a refresh file. DISEND causes a JUMP DI$LØP (jump to location DI$LØP) instruction to be inserted into the refresh file. Instructions associated with location DI$LØP are identified under the description of the DISINT macro.

NOTE

The DISEND macro must be used only in refresh files that begin with a DISINT macro.

DISINT

Used as the initial macro when generating a refresh file for which standard display parameters are desired. Causes the following display processor and graphic controller instructions to be generated:

```
CLR FUNS                              ;clear graphic controller
                                      ; function control stop rgtr
MOV DI$LØP,@#165ØØ6                   ;start graphic controller by
                                      ; loading location DI$LØP
                                      ; into graphic controller
                                      ; program counter
WAIT                                  ;display processor wait
BR .-2                                ;wait loop
DI$LØP: LDZZ <ALL,BLOFF,LINE,BR7>     ;set Z-axis parameters:
                                      ; all CRT's enabled, no
                                      ; blink, solid line, max
                                      ; brightness
LDDP <FAST,F6Ø,NOPP,NOROTATE,CSØ>     ;set display parameters:
                                      ; fast writing speed,
                                      ; 60 Hz frame sync, no
                                      ; PHOTOPENs enabled, non-
```

TABLE B-2

DETAILED MACRO DESCRIPTION (Cont)

Macro Call	Description
DISINT (Cont)	; rotated text, min
	; character size
LDTI 12	;set text increment to 12
IZPR	;initialize graphic
	; controller
LDXA Ø	;center
MVYA Ø	; CRT beam(s)
WATE	;wait for frame sync

NOTE

User written display macros immediately follow the DISINT macro and are normally terminated with a DISEND macro.

DRKY q,nnn

Inserts a DRKY (draw conic Y) instruction into the refresh file. Argument "q" specifies unblanking of quadrants II (upper left) and IV (lower right) as follows:

q	quadrants unblanked
0	neither
1	II
2	IV
3	II and IV

Argument "nnn" (which may vary from 0 to 777) specifies the semi-axis dimension of an ellipse in terms of coordinates along Y axis. If a DRKY instruction is not preceded by an LDKX (load conic Y) instruction, a circle will be drawn with a radius equal to nnn. In this case, "q" specifies unblanking of upper and lower semicircles as follows:

q	semicircles unblanked
0	neither
1	upper
2	lower
3	upper and lower

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
DRSR xx,yy	Inserts a DRSR (draw short relative) instruction into the refresh file. Arguments "xx" and "yy" specify the vector length in terms of the X and Y axes, respectively. Both "xx" and "yy" may vary from -40 to 37.
DRXA nnnn	Inserts a DRXA (draw X absolute) instruction into the refresh file. The X axis coordinate to which the vector is to be drawn is specified by argument "nnnn" which may vary from -2000 to 1777.
DRXR nnnn	Inserts a DRXR (draw X relative) instruction into the refresh file. The relative distance, in terms of coordinates, that the vector is to be drawn with respect to the X axis is specified by argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.
DRYA nnnn	Inserts a DRYA (draw Y absolute) instruction into the refresh file. The Y axis coordinate to which the vector is to be drawn is specified by argument "nnnn" which may vary from -2000 to 1777.
DRYR nnnn	Inserts a DRYR (draw Y relative) instruction into the refresh file. The relative distance, in terms of coordinates, that the vector is to be drawn with respect to the Y axis is specified by argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.
ENTR	Inserts a word (equal to 000000) into the refresh file. This word is normally used as an entry point for a subroutine called by a JMPM (jump and mark) instruction. The location of the word is used to store the return address.
HREF	Inserts an HREF (halt refresh) instruction into the refresh file.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
IZPR	Inserts an IZPR (initialize) instruction into the refresh file. This instruction, which should be at the beginning of each refresh file, initializes ramp generating circuits to ensure that no variations in the displayed image are introduced by the hardware.
JMPM label,I	Inserts a JMPM (jump and mark) instruction into the refresh file. The jump address is specified by argument "label". If argument "I" is absent, the jump will be direct (to the jump address); if argument "I" is present, the jump will be indirect (to the address contained in the jump address). Direct jumps cannot be made to addresses higher than 77776.
JMPR label	Inserts a JMPR (jump short relative) instruction into the refresh file. Argument "label" may specify any even location that is within -200 or +376 bytes of the current program counter location.
JMPZ label,I	Inserts a JMPZ (jump if display register 0 contents ≠ 0) instruction into the refresh file. The jump address is specified by argument "label". If argument "I" is absent, the jump will be direct (to the jump address); if argument "I" is present, the jump will be indirect (to the address contained in the jump address). Direct jumps cannot be made to addresses higher than 77776.
JPRZ label	Inserts a JPRZ (jump relative if display register 0 contents ≠ 0) into the refresh file. Argument "label" may define any even location in memory.
JRMP label	Inserts a JRMP (jump relative) instruction into the refresh file. Argument "label" may define any even location in memory.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
JUMP label,I	Inserts a JUMP instruction into the refresh file. The jump address is specified by argument "label". If argument "I" is absent, the jump will be direct (to the jump address); if argument "I" is present, the jump will be indirect (to the address contained in the jump address). Direct jumps cannot be made to addresses higher than 77776.
LDDI r,nnnnnn	Inserts an LDDI (load display register immediate) instruction into the refresh file. Argument "r", which must be "0", "1", "2", or "3", specifies one of the general purpose registers (DR0 through DR3) of the graphic controller. Argument "nnnnnn" specifies the value (-100000 to 77777) to be loaded into the register.
LDDP <arg1,arg2,...,argn>	Inserts a LDDP (load display parameter register) instruction into the refresh file. Multiple arguments, which must be enclosed in angle brackets, may be used in any order from the following list (arguments are required only for parameters that are to be changed):

Argument	Description	Octal Value
FAST	Select fast writing speed	2000
SLOW	Select slow writing speed	3000
NOPP	Disable all PHOTOPENs	100
PP1	Enable PHOTOPEN 1	120
PP2	Enable PHOTOPEN 2	140
ALLPP	Enable all PHOTOPENs	160
F6Ø	Select 60 Hz frame sync (see note)	2
F4Ø	Select 40 Hz frame sync (see note)	4
F3Ø	Select 30 Hz frame sync (see note)	6
ROTATE	Rotate characters 90° ccw	4
NOROTATE	Select normal character orientation	0

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description

LDDP <arg1,arg2,...,argn> (Cont)

Argument	Description	Octal Value
CSØ	Select character size 0 (smallest)	10
CS1	Select character size 1	11
CS2	Select character size 2	12
CS3	Select character size 3 (largest)	13

NOTE

A frame sync select argument (F6Ø, F4Ø, or F3Ø) must be used with an LDDP macro at least once in each refresh file. Otherwise, the selected frame sync rate will be indeterminate.

LDDZ <arg1,arg2,...,argn>

Inserts an LDDZ (load display Z register) instruction into the refresh file. Multiple arguments, which must be enclosed in angle brackets, may be used in any order from the following list (arguments are required only for parameters that are to be changed):

Argument	Description	Octal Value
CRT1	Enable Z axis of display 1	3000
CRT2	Enable Z axis of display 2	2400
CRT3	Enable Z axis of display 3	2200
CRT4	Enable Z axis of display 4	2100
ALL	Enable Z axis of all displays	3700
NOCRT	Disable Z axis of all displays	2000
BLOFF	Disable blinking of display image	0
BLON	Enable blinking of display image	40
LINE	Select solid vectors	0
DOT	Select dotted vectors	10
DASH	Select dashed vectors	20
DOTDASH	Select dot-dash (center line) vectors	30

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description

LDDZ <arg1,arg2,...,argn> (Cont)

Argument	Description	Octal Value
BRØ	Select intensity level 0 (off)	0
BR1	Select intensity level 1	1
BR2	Select intensity level 2	2
BR3	Select intensity level 3	3
BR4	Select intensity level 4	4
BR5	Select intensity level 5	5
BR6	Select intensity level 6	6
BR7	Select intensity level 7 (brightest)	7

LDKX q,nnn

Inserts an LDKX (load conic X register) instruction into the refresh file. Argument "q" specifies unblanking of quadrants I (upper right) and III (lower left) as follows:

q	quadrants unblanked
0	neither
1	I
2	III
3	I and III

Argument "nnn" (which may vary from 0 to 777) specifies the semi-axis dimension of an ellipse in terms of coordinates along the X axis.

LDRI d,r,nnnn

Inserts an LDRI (load device register immediate) instruction into the refresh file. Argument "d" specifies the number of the device (0 to 7) containing the register to be loaded. Argument "r" specifies the number of the register (0 to 7) within the device. Argument "nnnn", which may vary from -2000 to 3777, specifies the value to be loaded into the register.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
LDSP nnnnnn	Inserts an LDSP (load stack pointer) instruction into the refresh file. Argument "nnnnnn" specifies the stack address that is to be loaded into the graphic controller stack pointer.
LDTI nn	Inserts an LDTI (load text increment register) instruction into the refresh file. Argument "nn", which may vary from 0 to 77, specifies the text increment to be used for tabular characters contained in the arguments of TXT (draw two tabular characters) and TEXT (draw tabular character string) macros. If argument "nn" is not present and a text increment has previously been established, the established text increment will be used. If argument "nn" is not present and a text increment has not previously been established, a default increment of 12 will be used.
LDXA nnnn	Inserts an LDXA (load X absolute) instruction into the refresh file. The X axis coordinate to which the CRT beam is to be moved is specified by argument "nnnn" which may vary from -2000 to 1777.
LDXR nnnn	Inserts an LDXR (load X relative) instruction into the refresh file. The relative distance, in terms of coordinates, that the CRT beam is to be moved with respect to the X axis is specified by argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.
LINK label,I	Inserts a LINK (synchronized linkage) instruction into the refresh file. The link address is specified by argument "label". If argument "I" is absent, the link will be direct (to the link address); if argument "I" is present, the link will be indirect (to the address contained in the link address). Direct links cannot be made to addresses higher than 77776. If argument "label" is absent, the LINK

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
LINK label,I	instruction inserted into the refresh file will specify a direct link to address 170.
MVSR xx,yy	Inserts an MVSR (move short relative) instruction into the refresh file. Arguments "xx" and "yy" specify the distances the CRT beam is to be moved in terms of coordinates along the X and Y axes, respectively. Both "xx" and "yy" may vary from -40 to 37.
MVXA nnnn	Inserts an MVXA (move X absolute) instruction into the refresh file. The X axis coordinate to which the CRT beam is to be moved is specified by argument "nnnn" which may vary from -2000 to 1777.
MVXR nnnn	Inserts an MVXR (move X relative) instruction into the refresh file. The relative distance, in terms of coordinates, that the CRT beam is to be moved with respect to the X axis is specified by argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.
MVYA nnnn	Inserts an MVYA (move Y absolute) instruction into the refresh file. The Y axis coordinate to which the CRT beam is to be moved is specified by argument "nnnn" which may vary from -2000 to 1777.
MVYR nnnn	Inserts an MVYR (move Y relative) instruction into the refresh file. The relative distance, in terms of coordinates, that the CRT beam is to be moved with respect to the Y axis is specified by argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.
NEWL	Causes a relative move to the left hand margin of a new tabular line by inserting two instructions into the refresh file. The first is an LDXA (load X absolute) instruction that specifies the margin established by a previous SETMRG (set margin) macro. The second is an MVYR (move Y relative)

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
NEWL (Cont)	instruction that specifies the Y axis increment established by a previous SETLF (set line feed) macro. If a SETMRG macro has not been used previously, a margin of 0 is assumed. If a SETLF macro has not been used previously, a line feed increment of 15 is assumed.
NEWLR ssss,1111,R	Causes a relative move to the beginning of a new tabular line by inserting two instructions into the refresh file. The first instruction inserted is an LDXR (load X relative) instruction that specifies the number of spaces (text increment units) that the CRT beam is to be moved with respect to the X axis. If argument "R" is absent, the CRT beam will be moved to a new line for normal characters; if "R" is present, the beam will be moved to a new line for rotated characters. Argument "ssss" specifies the number of spaces. Positive arguments backspace the beam to the left; negative arguments space the beam to the right. If a text increment has not been established by a previous macro, a value of 12 is assumed. Argument "ssss" multiplied by the text increment may vary from -2000 to 1777. The second instruction inserted is an MVYR (move Y relative) instruction that specifies the number of lines (line feed units) that the CRT beam is to be moved with respect to the Y axis. Argument "1111" specifies the number of lines. Positive arguments move the beam down; negative arguments move the beam up. If a line feed increment has not been established by a SETLF (set line feed) macro, a value of 15 is assumed. Argument "1111" multiplied by the line feed increment may vary from -2000 to 1777.
NOOP	Inserts a NOOP (no operation) instruction into the refresh file.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
PPLR xx,yy	Inserts a PPLR (point plot relative) instruction into the refresh file. Arguments "xx" and "yy" specify the distances the CRT beam is to be moved in terms of coordinates along the X and Y axes, respectively. A point is then displayed at the new position of the CRT beam. Both "xx" and "yy" may vary from -40 to 37.
SETLF nnnn	Establishes the line feed increment to be used by the NEWL (new line) and the NEWLR (new line relative) macros. The increment is specified in terms of Y axis coordinates by argument "nnnn". Positive arguments result in increments that move the CRT beam down; negative arguments result in increments that move the CRT beam up. Argument "nnnn" may vary from -2000 to 1777. Refer to table A-13 for recommended line feed increments.
SETMRG nnnn	Establishes the left hand margin to be used for the NEWL (new line) macro. The margin is defined in terms of character spaces (text increments) by argument "nnnn". The actual X coordinate to be used as the margin is calculated by multiplying "nnnn" times the text increment established by a previous macro. If a text increment has not been established, a value of 12 is assumed. Argument "nnnn" multiplied by the text increment may vary from -2000 to 1777.
SETTI nnnn	Establishes the text increment to be used by the LDTI (load text increment register) instruction the NEWLR (new line relative) macro and the SETMRG (set margin) macro. The increment is specified in terms of X axis coordinates by argument "nnnn". Positive arguments result in increments that move the CRT beam to the right; negative arguments result in increments that move the CRT beam to the left. Argument "nnnn" may vary from -2000 to 1777. Refer to table A-13 for recommended text increments.

TABLE B-2

DETAILED MACRO DESCRIPTIONS (Cont)

Macro Call	Description
TEXT <character string>	Inserts multiple TXT (draw two tabular characters) instructions into the refresh file. One TXT instruction is inserted for each pair of ASCII characters specified by "character string". If an odd number of characters is specified, a null is inserted as the second character of the final TXT instruction.
TXT a,b	Inserts a TXT (draw two tabular characters) instruction into the refresh file. Arguments "a" and "b" specify the first and the second ASCII characters, respectively, to be drawn.
WATE	Inserts a WATE (wait) instruction into the refresh file.
COLOR <arg1,arg2>	Inserts an LDDI instruction into refresh to select one of four colors and to select the CRT display indicator. (LDDI is a two word instruction.)

TABLE B-3

TYPICAL PROGRAM STRUCTURES

```
.TYPE TABB3.CPY
00100   ;
00200   ;
00300   ;SAMPLE PROGRAM NO. 1.  A SIMPLE DRIVER, NOT USING THE DISINT
00400   ;  AND DISEND MACROS, WITH A PLACE FOR DISPLAY INSTRUCTIONS.
00500   ;
00600   .TITLE  SAMPL1
00700   .SBTTL  SAMPL1 DRIVER
00800   ;
00900   ;
01000           .ASECT                                  ;BEGIN ASSEMBLY
01100           .=2000                                  ;  AT ADDRESS 2000
01200           .BLKW   30                              ;SAVE SPACE FOR DISPLAY PROCESSOR STACK
01300   BEGIN:  RESET                                   ;CLEAR PROCESSOR BUS
01400           MOV     #BEGIN,SP                       ;LOAD DISPLAY PROCESSOR STACK POINTER
01500           CLR     @#FUNS                          ;HALT GRAPHIC CONTROLLER
01600           MOV     #START,@#DPC                    ;START GRAPHIC CONTROLLER
01700   LOOP:   WAIT                                    ;DISPLAY PROCESSOR WAIT
01800           BR      LOOP                            ;WAIT LOOP
01900   START:  LDDZ    <ALL,BLOFF,LINE,BR7>            ;SET Z PARAMETERS
02000           LDDP    <FAST,F60,NOPP,NOROTATE,CS0>    ;SET P PARAMETERS
02100           LDTI    12                              ;SET TEXT INCREMENT TO 12
02200   AGAIN:  IZPR                                    ;INITIALIZE GRAPHIC CONTROLLER
02300           LDXA    0                               ;MOVE CRT BEAM(S)
02400           MVYA    0                               ;  TO CENTER
02500           WATE                                    ;WAIT FOR FRAME SYNC
02600   ;
02700   ;       BODY
02800   ;        OF
02900   ;          USER
03000   ;            WRITTEN
03100   ;             DISPLAY
03200   ;              PROGRAM
03300   ;               GOES
03400   ;                HERE
03500   ;
03600           JUMP    AGAIN                           ;SHOW THE PICTURE AGAIN
03700           .END    BEGIN
03800   ;
03900   ;
04000   ;SAMPLE PROGRAM NO. 2.  A SIMPLE DRIVER, USING THE DISINT
04100   ;  AND DISEND MACROS, WITH A PLACE FOR DISPLAY INSTRUCTIONS.
04200   ;
04300   .TITLE  SAMPL2
04400   .SBTTL  SAMPL2 DRIVER
04500   ;
04600   ;
04700           .ASECT                                  ;BEGIN ASSEMBLY
04800           .=2000                                  ;  AT ADDRESS 2000
04900           .BLKW   30                              ;SAVE SPACE FOR DISPLAY PROCESSOR STACK
05000   BEGIN:  RESET                                   ;CLEAR PROCESSOR BUS
05100           MOV     #BEGIN,SP                       ;LOAD DISPLAY PROCESSOR STACK POINTER
05200           DISINT                                  ;SET PARAMETERS AND START GRAPHIC CONTROLLER
05300   ;
05400   ;       BODY
05500   ;        OF
05600   ;          USER
05700   ;            WRITTEN
05800   ;             DISPLAY
05900   ;              PROGRAM
06000   ;               GOES
06100   ;                HERE
06200   ;
06300           DISEND                                  ;SHOW THE PICTURE AGAIN
06400           .END    BEGIN
06500   ;
06600   ;
```

APPENDIX C

GCP+ PROGRAMMING CAUTIONS

C-1. When GCP+ is initialized the command header error detection is disabled. Normally, the user should send an IM message to activate error detection.

NOTE

For the previous version of the Graphics Control Program (i.e., GCP) error detection was always enabled; but with GCP+, error detection is disabled at initialization. Although there is a difference between GCP and GCP+ in the error detection area, all previously developed GCP programs should still run with GCP+.

C-2. No user refresh programs should start below address 3000 (octal).

NOTE

For GCP, user refresh programs could not start below address 2000 (octal). Any previously developed GCP program should still run with GCP+ provided that the user refresh program doesn't start below address 3000.

C-3. When writing refresh programs, the user should ensure that the 32-word depth or limit of the graphic controller stack is not exceeded.

C-4. When MU, SU, and GU messages are sent from the host to GCP+, the user should ensure that the words counts associated with these messages are correct.

C-5. Each release of GCP+ has a different checksum value.

C-6. GCP+ transfers some data to host computer via input DMA mode. Customers upgrading from GCP to GCP+ should ensure that their parallel interface has all ECO's incorporated. (Original parallel interfaces released with GCP had a logic error in input DMA section 1.)

APPENDIX D

GCP+ RELEASE HISTORY

RELEASE	VERSION	DATE	CHECKSUM
1	Ø	31 AUG 79	107731
2	ØA (NASA)	25 FEB 80	147325
3	1	2 MAY 80	116415

THE INTENT AND PURPOSE OF THIS PUBLICATION IS TO PROVIDE ACCURATE AND MEANINGFUL INFORMATION TO SUPPORT EQUIPMENT MANUFACTURED BY SANDERS ASSOCIATES, INC. YOUR COMMENTS AND SUGGESTIONS ARE REQUESTED.

PLEASE USE THE FORM ON THE REVERSE SIDE TO REPORT ANY PROBLEMS YOU HAVE HAD WITH THIS PUBLICATION OR THE EQUIPMENT IT DESCRIBES.

FOLD FOLD

FIRST CLASS
PERMIT NO. 568
NASHUA, N.H.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by

Sanders Associates, Inc.
Information Products Division
Daniel Webster Highway, South
P.O. Box 868
Nashua, NH 03061

ATTN: DEPARTMENT 1-2894 (NHQ 1-447)

FOLD FOLD

Information Products Division
Federal Systems Group

SANDERS

Name:_____

Company:_____

Address:_____

_____

Telephone:[ ]_____

Date:_____

Sanders Equipment_____

Part Number_____

Software/Firmware System_____

Version _____

Host computer_____

Host operating system_____ Version_____

Host-GRAPHIC 7 interface_____

My problem is: hardware ☐ software ☐

firmware ☐ manual ☐

Description of problem (or suggestion for improvement):

Related tech manual number _____