H-80-0444

Vistagraphic™ 3000/ Graphic 8™

COMPUTER GRAPHICS DISPLAY SYSTEM

SERIES 8X00

PROGRAMMER'S REFERENCE MANUAL

FEBRUARY 1981

CALCOMP
A Sanders Graphics Company
SANDERS

H-80-0444

Vistagraphic™ 3000/ Graphic 8™

COMPUTER GRAPHICS DISPLAY SYSTEM
SERIES 8X00

PROGRAMMER'S REFERENCE MANUAL

FEBRUARY 1981

CALCOMP
A Sanders Graphics Company
SANDERS

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

Original issue: February 1981

Reprint: August 1981

Reprint: November 1981

Reprint: February 1982

Reprint: April 1982

Change 1: September 1982

Reprint: September 1982

Reprint: January 1983

RECORD OF CHANGES

CHANGE NO.	DATE	TITLE OR BRIEF DESCRIPTION	ENTERED BY
1	Sept 82	Correct errors, general update	

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

Summary of Controller Instructions

Summary of Controller Instructions (Cont)

Summary of Host-GRAPHIC 8 Messages

Summary of Host-GRAPHIC 8 Messages (Cont)

MESSAGE	ORIGIN	DESCRIPTION	PAGE
PL	Host	Poll GRAPHIC 8 for next message	5-15
PV	Host	Packed vector	5-65
RG	GRAPHIC 8	Return FSP table address	5-55
RI	GRAPHIC 8	Return image	5-24
RK	GRAPHIC 8	Function keyboard No. 1	5-39
RL	GRAPHIC 8	Function keyboard No. 2	5-39
RO	GRAPHIC 8	Return option	5-72
RP	GRAPHIC 8	Return PED No. 1	5-48
RR	GRAPHIC 8	Return register	5-27
RT	GRAPHIC 8	Return PED status	5-46
RU	Host	Register update	5-20
RW	GRAPHIC 8	Return PED No. 2	5-48
SP	Host	Start picture	5-21
SU	Host	Selective update	5-19
TK	Host	Transfer control	5-23
TM	Host	Assign data tablet as PED No. 1	5-41
TN	Host	Assign data tablet as PED No. 2	5-41
TS	GRAPHIC 8	2D/3D coordinate converter status	5-75
VL	GRAPHIC 8	Variable length (follows RI or RO)	5-25
YI	GRAPHIC 8	X or Y position overflow interrupt	5-32
XR	GRAPHIC 8	Scratchpad ready for alphanumeric keyboard No. 1	5-38
XT	GRAPHIC 8	Scratchpad ready for alphanumeric keyboard No. 2	5-38
XX	GRAPHIC 8	Error status	5-8
ZI	Host	Disable selected interrupts	5-30
ZN	Host	Disable error number	5-62
ZP	Host	Disable box display	5-60
ZR	Host	Initialize scratchpad for alphanumeric keyboard No. 1	5-34
ZS	Host	Zero scratchpad No. 1	5-35
ZT	Host	Initialize scratchpad for alphanumeric keyboard No. 2	5-34
ZU	Host	Zero scratchpad No. 2	5-35

SECTION 1

GRAPHIC 8 SYSTEM DESCRIPTION

1.1 INTRODUCTION

The Sanders Associates, Inc. GRAPHIC 8™ is a high-performance, intelligent computer graphics terminal system incorporating refreshed raster display technology. The Graphic 8 combines sophisticated display processing techniques, developed by Sanders in the popular GRAPHIC 7™ refreshed stroke graphics product line, with new CRT raster graphics features. It is designed to interface a host computer and to support operator CRT display monitor stations configured with interactive devices, such as keyboards, trackballs, forcesticks, and data tablets. Also, it can produce permanent hard copy records of displayed data.

The GRAPHIC 8 system utilizes many of the key features of the GRAPHIC 7 system. For instance, the GRAPHIC 8 can use any of the existing GRAPHIC 7 high-speed, parallel host computer interfaces or the RS-232C port. Existing GRAPHIC 7 software, including the Sanders' FORTRAN Support Package (FSP), may be used with the GRAPHIC 8. And both systems use a common display processor instruction set.

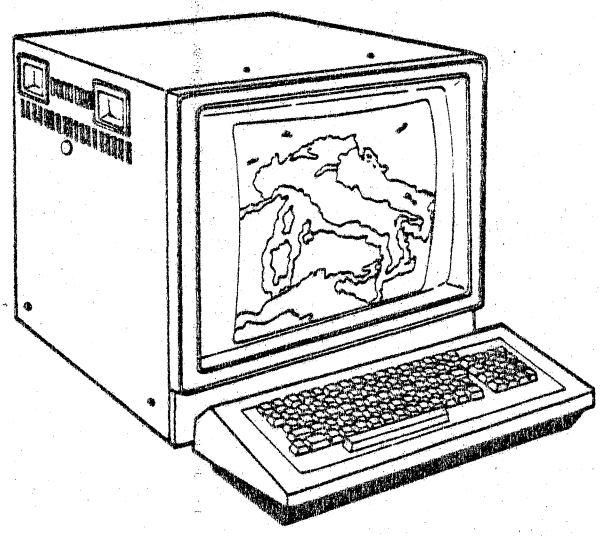
The 8X00 series GRAPHIC 8 features a dynamic display update via a double refresh buffer memory technique. From one up to four CRT display monitors are supported by the 8X00 series configurations. Resolutions of 512 x 512, 640 x 480, 1024 x 768 (interlaced) or 1024 x 1024 (interlaced) are available. Both color and monochrome versions are offered with up to 8 bits per pixel to provide as many as 256 simultaneous colors or monochrome intensities (or 128 plus blink).

The GRAPHIC 8 display processor is a general purpose digital computer with a set of over 400 instructions that controls a variety of functions, which reduce the loading on the host computer. In combination with the host computer, the GRAPHIC 8 system permits the user to display digital data in a visual format on the CRT display monitor and to interact with the displayed image by means of keyboards, forcesticks, trackballs, and data tablets. Its high performance and intelligence make it well suited to a variety of applications, such as, CAD/CAM, simulation and training, command and control, cartography, and many others.

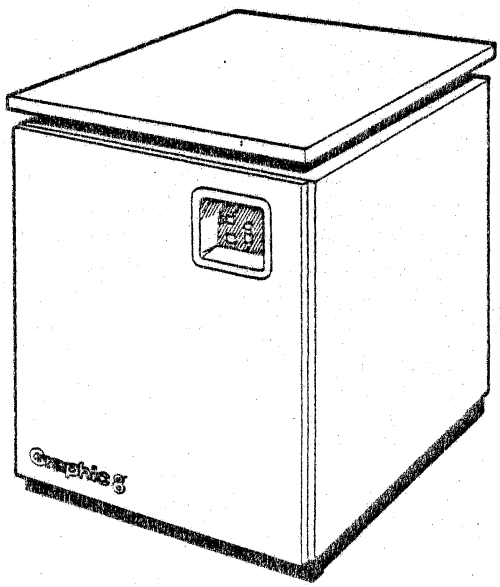
1.2 COMPONENT DESCRIPTION

The basic GRAPHIC 8 system consists of a terminal controller and a monitor. The basic system can be expanded to include a wide variety of options and enhancements.

™GRAPHIC 8 and GRAPHIC 7 are trademarks of Sanders Associates, Inc.



COLOR DISPLAY MONITOR
AND KEYBOARD

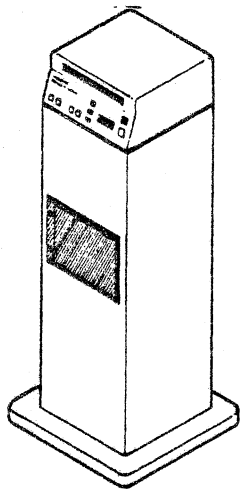


TERMINAL CONTROLLER

Figure 1-1. GRAPHIC 8 System Components

DESCRIPTION

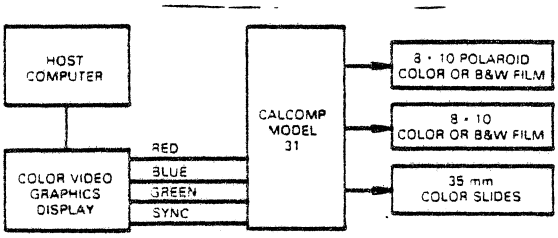
The CalComp Model 31 is a stand-alone recording system for making 8 × 10 inch color or black and white prints and 35 mm color slides of any data presented on its self-contained, high resolution raster scan CRT. With Polaroid Type 808 Polarcolor 2 Land film, the Model 31 makes color prints instantly available.



FEATURES

- Exceptional image resolution with accurate rendition of color hue, saturation and lightness
- High throughput (at least thirty 8 × 10 prints or over one hundred 35 mm slides per hour)
- For use with all raster scan computer graphics and digital image processing applications
- Microprocessor controlled to assure consistent exposure, reliable operation
- Separation mode for automatic three-color separation exposures
- Switchable raster blending for high color saturation prints
- Fully automated 35 mm slide capability with remote control (optional)

SYSTEMS INTERCONNECT



SPECIFICATIONS AND CHARACTERISTICS

Video Monitor:
Nominal resolution of 1400 lines center screen at 100 cd/m² (30 fL) on flat-face CRT.
Pixel position error is<0.5% within a 9 cm circle, <1% at corners.

Speed of Operation:
8" by 10" Polaroid—less than 60 seconds per exposure
8" by 10" Transparency—less than 40 seconds per exposure
35 mm—6 seconds per exposure

Film Type:
8" × 10" (20.32 cm × 25.40 cm)—color or black and white. Must be loadable in cassette format.
With optional auxiliary camera, 35 mm sprocket feed film—color only, in cartridge.

Film Speed:
15 DIN (ASA25) to 24 DIN (ASA200)

Physical Specifications:

Width	38.7 cm (15.25 in.)
Depth	38.7 cm (15.25 in.)
Height	113.2 cm (44.75 in.)
Base	50.4 cm × 50.4 cm (20 in. × 20 in.)
Weight	40 kg (88 lbs.)

Power Requirements: (all units single phase, line to neutral):

Standard	120 VAC ± 10%, 50/60 Hz, 0.8 amps, 110 watts (max)
Optional	100 VAC ± 10%, 50/60 Hz, 1.0 amps, 220 or 240 VAC ± 10%, 50/60 Hz, 0.5 amps

Video Input:
Separate Red, Green, Blue, and Sync video signals of 0.35 to 2.0 peak to peak voltage required.
Standard—Handles horizontal line rate of 500 to 650.
Optional-High Line— Handles horizontal line rate of 800 to 1100 at 60 Hz or 1300 at 50 Hz.

Operating Environment:
Temperature—20°C ± 10°C
Relative Humidity—15% to 90% non-condensing
Altitude—Sea level to 4500 meters
Operating Noise—Negligible

Cables:
Power: 3 meter power cord.
Interface: Four 74 Ohm, RG-59 coaxial cables of 3 meters each. BNC Plug on each end will mate to BNC Bulkhead Receptacle.

Figure 1-2. CalComp Model 31 Color Graphics System Specifications

1.2.1 TERMINAL CONTROLLER. The GRAPHIC 8 system contains a terminal controller which consists of a rack mountable card cage and a power supply. As shown in figure 1-3, the cards are interconnected via either a processor bus or a graphic bus. The size of controller selected is based on the four following major considerations:

1. Color or monochrome

2. Number of simultaneous colors or intensities

3. Resolution of the display image

4. Number of display stations per controller

The GRAPHIC 8 terminal controller accommodates 17 cards and a power supply. The controller consists of a card cage with slots for 17 cards. Six of the slots are for the processor cards, one slot is for the digital graphics controller, one slot is for the timing module, and the remaining nine slots are for the mapping memory and video controller cards.

1.2.1.1 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-11/34® 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). An 8-bit configuration switch is program readable (used by GCP) from octal location 177774.

1.2.1.2 Read/Write Memory. Locations in the read/write memory are assigned addresses 000000_8 through 777777_8 and are accessed by means of a 18-bit address on the processor bus and by a 16-bit address and mapping registers on the memory card. The 18-bit address can be used to access the location of a word (16 bits) or of an individual byte (8 bits) as required. Refer to figure 1-4 for a GRAPHIC 8 system memory map.

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

®PDP and DEC are registered trademarks of Digital Equipment Corporation.

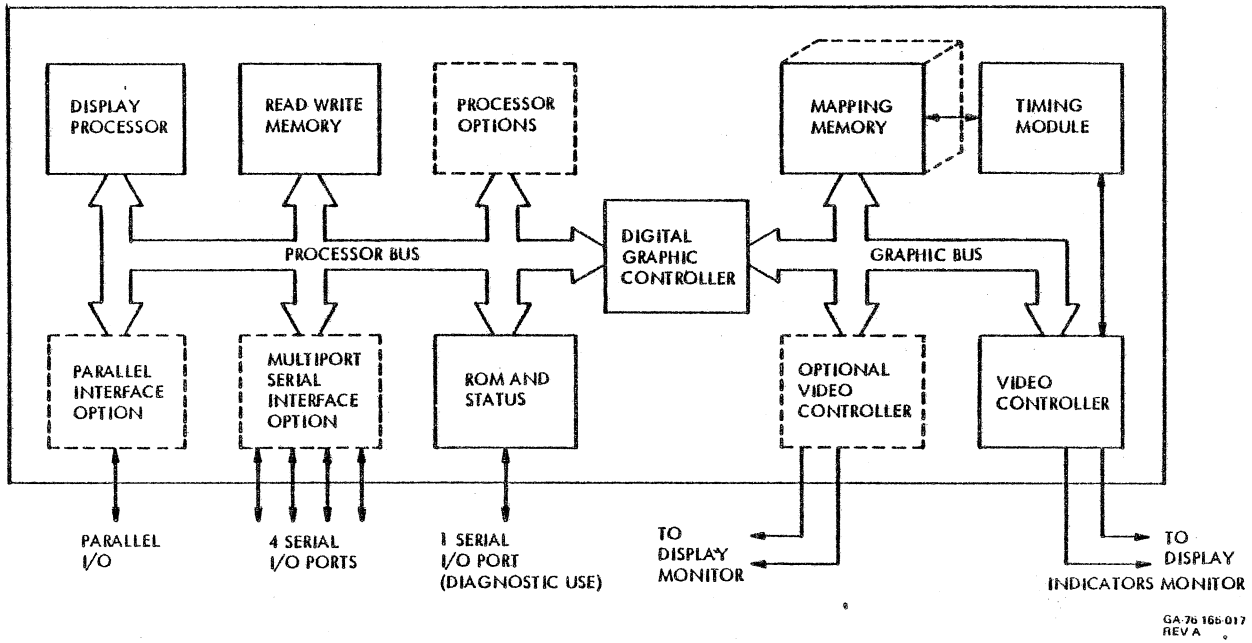


Figure 1-3. GRAPHIC 8 Terminal Controller Functional Block Diagram

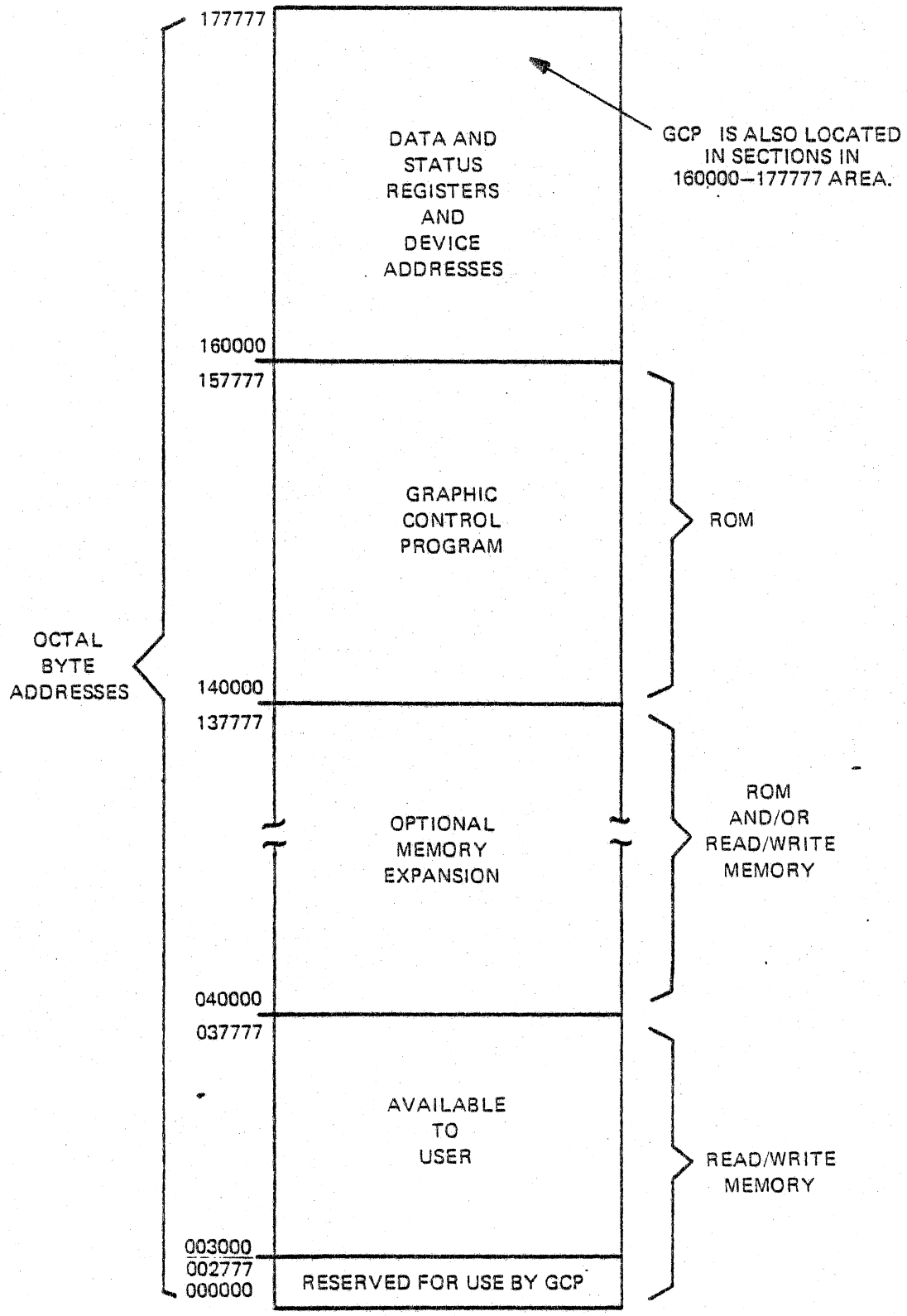


Figure 1-4. GRAPHIC 8 System Memory Map

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.

1.2.1.3 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 firmware. 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.

1.2.1.4 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.

Three 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.

1.2.1.5 Parallel Interface. An optional GRAPHIC 8 parallel interface allows high-speed communications with handshaking and is intended for applications where the Graphic 8 is located in proximity to the host. All parallel interface signals are TTL-compatible. Under program control, the interface operates in either an interrupt driven or a DMA mode. In the latter mode, the interface operates at speeds up to 500,000 16-bit words/sec. 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, two parallel interface cards may be installed, but are not supported by the standard Graphic Control Program.

1.2.1.6 Digital Graphic Controller. The digital graphics controller is a microprocessor with more than 50 instructions committed to ROM. It retrieves display update instructions from R/W memory, generates vectors, characters, conics, point plots, fills and stores these in mapping memory in raster-scan format. It executes all 40 display instructions of Sanders' stroke-writing (random position) product line graphics terminal, the GRAPHIC 7. Additional display instructions have been added to implement features unique to digital TV.

These digital 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 digital graphic controller may be considered as a device on the processor bus of the terminal controller. It contains its one set of registers that maintain instruction address, control fetch operations, 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 digital 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 Video Controller. The video controller obtains data from the mapping memory and formats it for presentation on the display monitor(s). Outputs are provided as either RGB color or monochrome and as composite video.

External video may be accepted by the video controller and logically ORed with internally-generated video. A single video controller can accommodate up to eight bits per pixel.

The video controller generates one non-destructive, full-screen, crosshair cursor and contains the cursor address registers which are accessible to the user. It controls the split screen function which allows the user to divide the display face into up to three variable-height horizontal bands and fill these bands with data from anywhere in addressable mapping memory. This feature allows the user to simultaneously view up to three separate areas of mapping memory which are not necessarily contiguous.

The video controller contains a 256 x 8-bit word RAM look-up-table (LUT) which allows pseudo-color or pseudo-gray level transformation to be made.

1.2.1.8 Mapping Memory. The mapping memory contains pixel data in a format which allows display refresh in a raster scan mode. The mapping memory may be configured for various resolutions up to 1024 x 1024 and for interlace or non-interlace refresh. A single memory board can be supplied with a capacity of over four million bits. Up to eight bits can be combined per pixel to provide 256 possible colors or intensity levels.

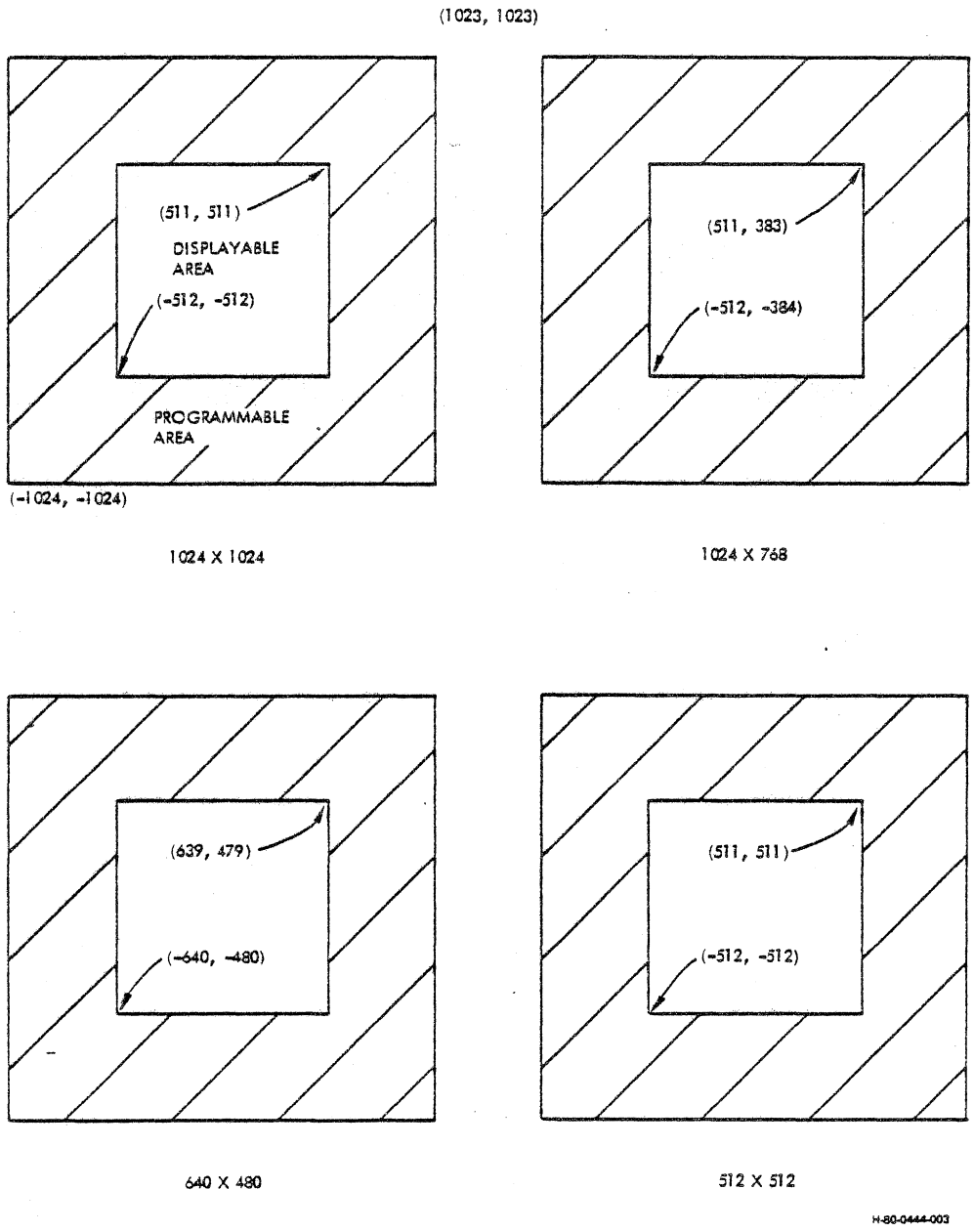


Figure 1-5. Addressable vs. Displayable Areas for the Four Screen Resolutions

A dual mapping memory configuration is for high-speed dynamic update of data. The dual memory concept allows the hardware to clear and update one memory while the second memory is refreshing the display. When the next update occurs, the roles of the two memories are reversed so that the previously updated memory now becomes the refresh memory.

1.2.1.9 Timing Module. The timing board generates all display-related timing signals as well as the necessary synchronization signals for the monitors. On-board switches allow selection for compatible operation with the possible resolutions and refresh frequencies.

1.2.1.10 Character Generation. Character generation is performed by the Digital Graphic Controller. The basic set of characters supplied is a standard set of 96 ASCII characters. When the ASCII code corresponding to the desired character is applied to the read-only memory, the character is drawn at the position determined by the current/position for X and Y.

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

Read-only memory 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 GRAPHIC 8.

1.2.1.11 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.

1.2.1.12 Data Converter. The model 5744 data 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 data 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 data 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.

The data converter is not supported by the standard Graphic Control Program.

1.2.1.13 EPROM Expansion Module. As options are added to the GRAPHIC 8, 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.

1.2.2 INPUT DEVICES. Optional data input devices for the GRAPHIC 8 give the operator two-way interaction with the display and processing system. Input devices available include two types of keyboards; a trackball, a forcestick, and a data tablet. The GCP in firmware can support up to eight keyboards, or eight 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 8 for the input of maintenance data.

1.2.2.1 Keyboards. Standard keyboards available for the GRAPHIC 8 are the Model 5783 and Model 5784 keyboards. The keyboards contain a main block of alphanumeric 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.

1.2.2.2 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.

1.2.2.3 Maintenance Data Input Devices. A teletypewriter and/or a paper tape reader can be connected to the GRAPHIC 8 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 8.

1.2.3 OUTPUT DEVICES. The standard output device for the GRAPHIC 8 is the CRT display monitor. A hard copy unit is available as an optional output device. Using the same signals that go to a standard display monitor, the hard copy unit can produce a duplicate on paper of any static image displayed on the CRT of the display monitor. Operation of the hard copy unit is controlled manually.

An optional hard copy multiplex switch is available. The multiplex switch is capable of interfacing up to four GRAPHIC 8 displays to a single hard copy unit.

1.2.3.1 Display Monitors. The GRAPHIC 8 offers the user a choice of configuration of eight CRT monitors (four monochrome and four color) to provide the right monitor for the intended application.

Positions on the 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 screen. Of the 2048 by 2048 addressable locations, the displayable area comprises the field of coordinates centered about the middle of the screen. Refer to figure 1-5 for different screen resolutions.

1.2.3.2 Hard Copy. Both monochrome and color hard copy devices are available for use with the GRAPHIC 8. Refer to figures 1-1 and 1-2.

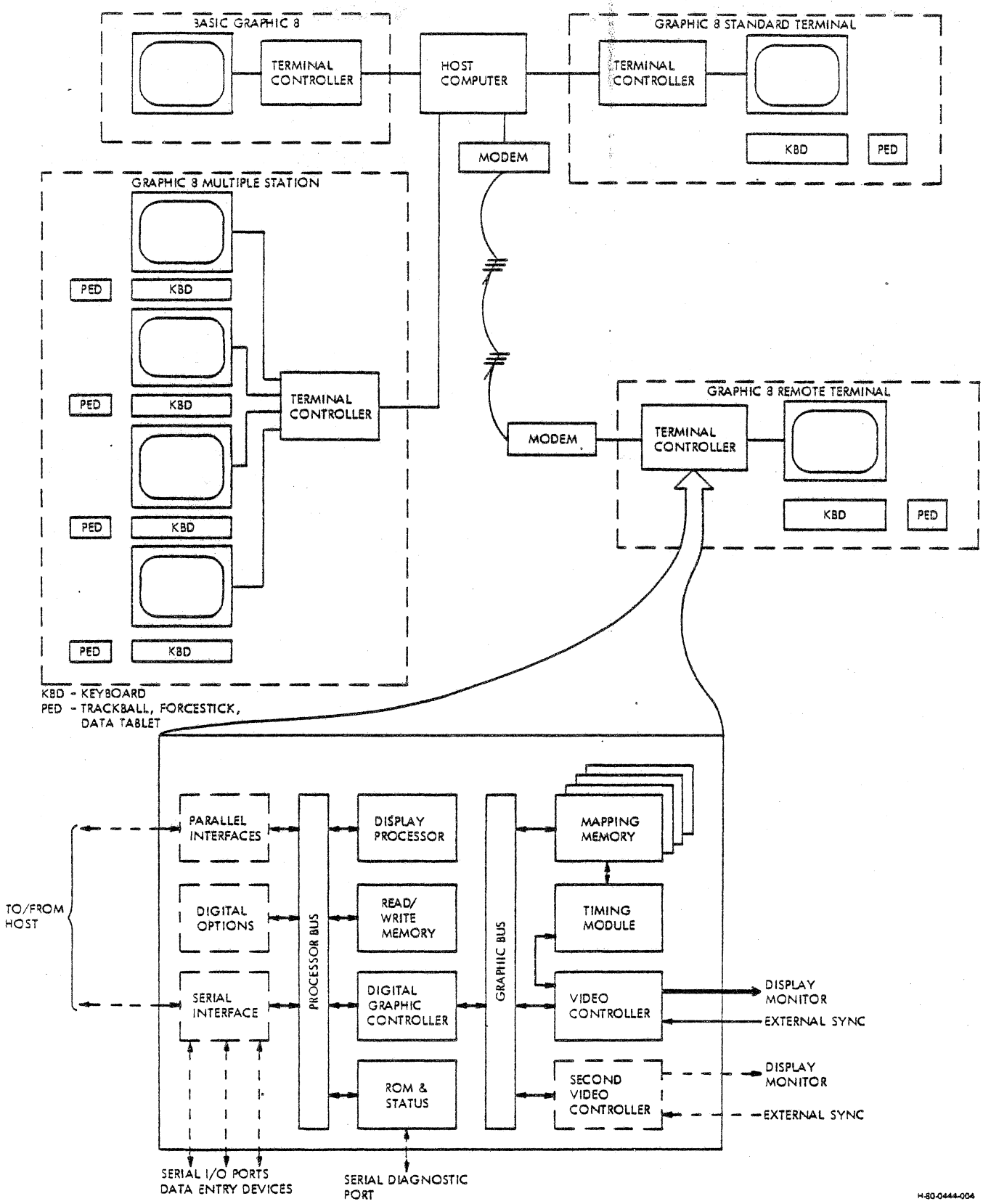


Figure 1-6. Representative GRAPHIC 8 System Configurations

SECTION 2

OPERATING MODES

2.1 GENERAL

The GRAPHIC 8 system can be operated in either the local or the system mode. In the local mode, the GRAPHIC 8 operates as a stand-alone system; in the system mode, the GRAPHIC 8 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 8, 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
MEMORY SYSTEM	0 0	0 0 0	0 0 0

Address 670 contains a copy of the system configuration register (see page page 4-22).

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)
0	0	0	0	0	0	0	0	0K
0	0	0	0	0	0	0	1	16K
0	0	0	0	0	0	1	1	32K
0	0	0	0	0	1	1	1	48K
0	0	0	0	1	1	1	1	64K
0	0	0	1	1	1	1	1	80K
0	0	1	1	1	1	1	1	96K
0	1	1	1	1	1	1	1	112K
1	1	1	1	1	1	1	1	128K
Banks		3		2		1		0

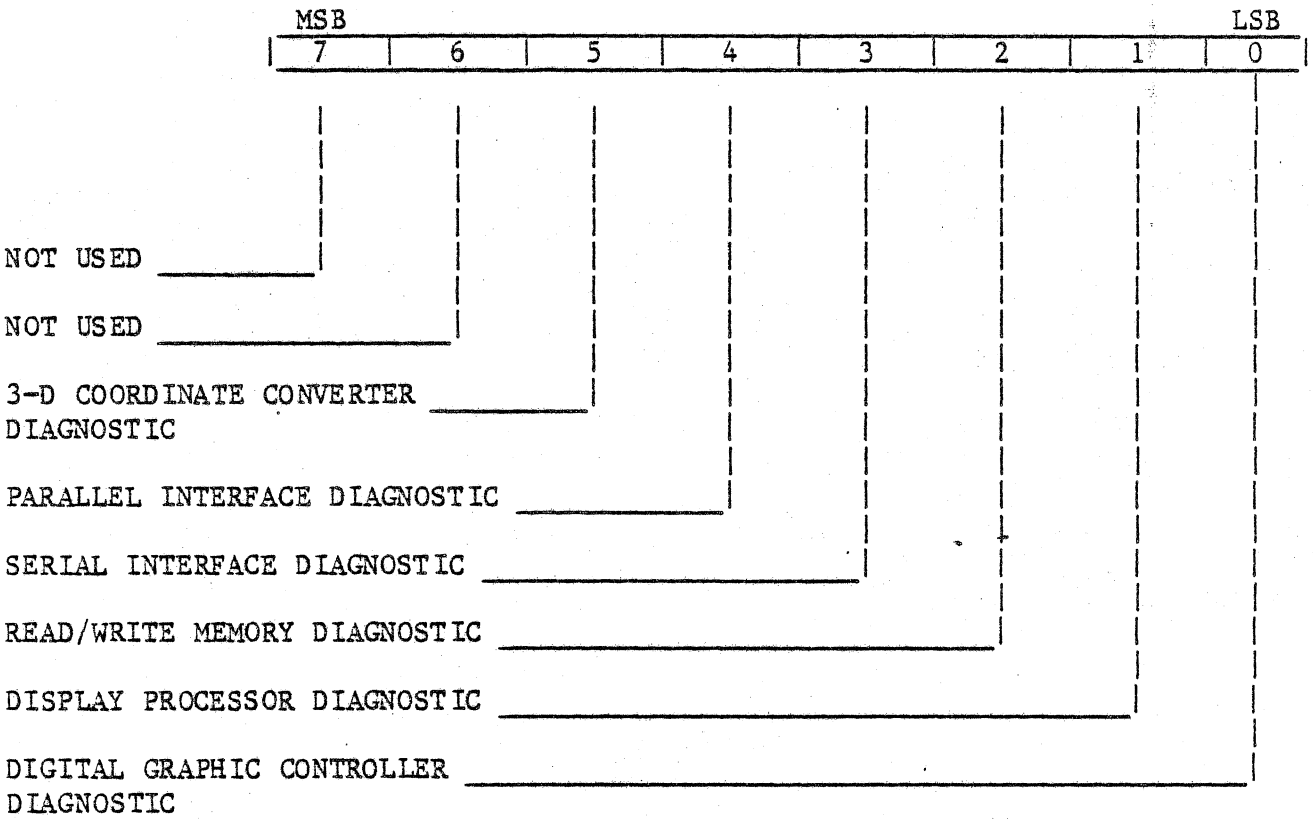
ROM CONFIGURATION WORD

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	ROM		0	0	0	0

B_5	B_4	
0	0	No optional ROM
1	0	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 monitor when the GRAPHIC 8 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:



A checksum calculation routine performed whenever the GRAPHIC 8 is initialized in the local mode is a checksum calculation based on all GCP stored in read only memory. The result is deposited in memory location - 500 (octal). Location 500 can also be examined as described in paragraph 2.2.2.1.

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.

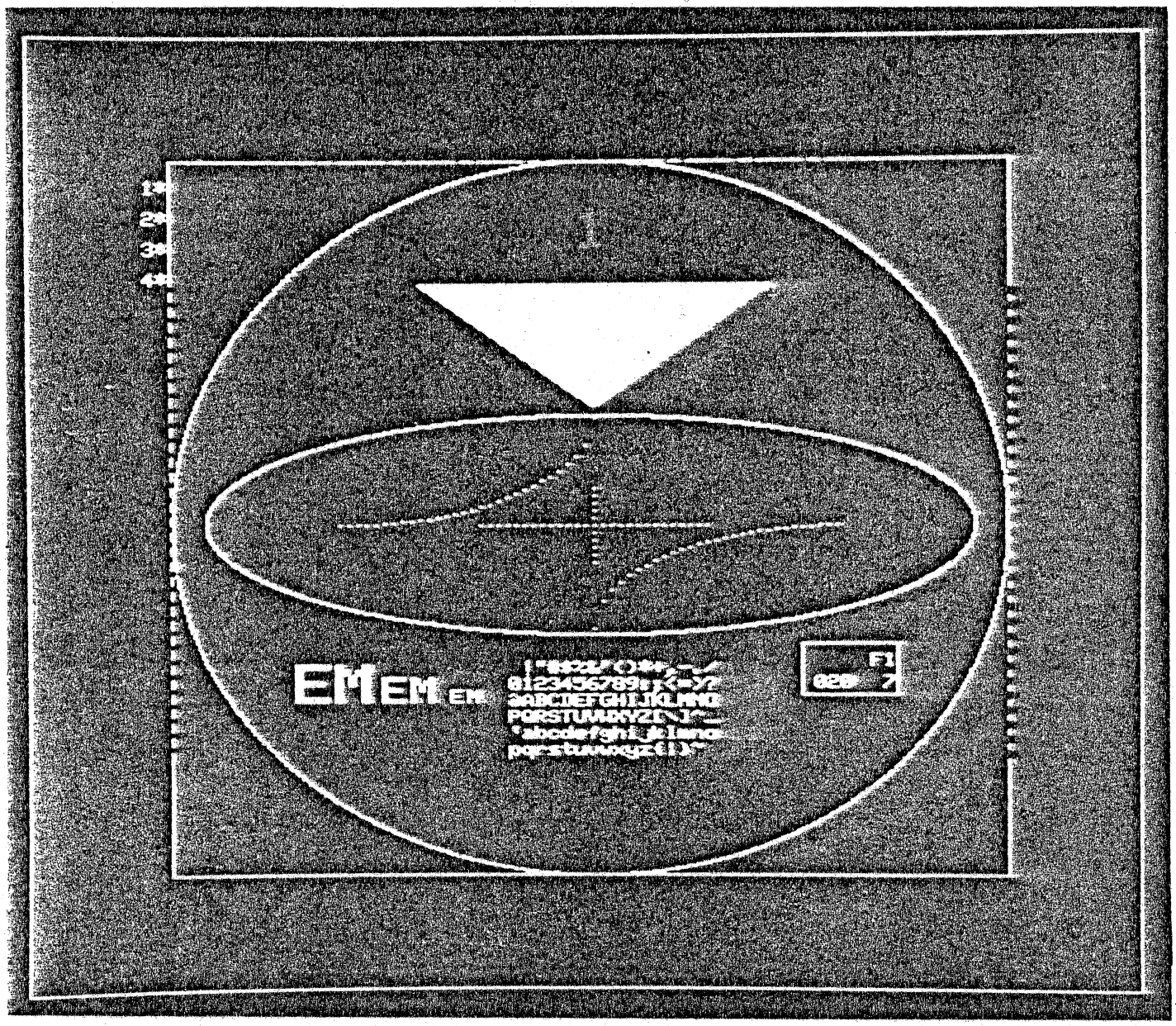


Figure 2-1. Verification Test Pattern

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.

Table 2-1. Serial Interface Port Codes

CODE	SERIAL INTERFACE PORT	DEVICE	ASSOCIATED CONNECTOR
TT	TTY	Console teletypewriter	J2 on ROM and Status card
F1	3	Function keyboard 1	J5 on multiport serial interface card #1
F2	7	Function keyboard 2	J5 on multiport serial interface card #2
F3	2	Function keyboard 3	J4 on multiport serial interface card #1
F4	9	Function keyboard 4	J3 on multiport serial interface card #3
S1	1	Serial host	J2 on multiport serial interface card #1
PED Indicators			
1*	4	PED #1	J6 on card #1
2*	8	PED #2	J6 on card #2
3*	6	PED #3	J4 on card #2
4*	10	PED #4	J4 on card #3

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 n* displayed on the test pattern are only those that are included in the system configuration (Display Processor II switch settings) regardless of whether the trackball or forcestick is physically 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.)

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 1-digit numeral in the upper center of the verification test pattern indicates the monitor (1-4).

A filled Polygon positioned beneath the blinking Monitor Number will display the default Colors/Gray levels. The number of Colors/Gray levels is a function of the configuration and can be 8, 16, 128, or 256.

2.2.1.1 Hardcopy Generation. A hardcopy of the terminal verification pattern can be generated by pressing function key F0 to freeze the pattern and then the hardcopy initiate button located on the hardcopy unit.

2.2.1.2 Data Tablet Testing. The data tablet can be tested by pressing function key F1. This causes the n* trackball/forcestick indicators to change to n#. 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 8 has been initialized in the local mode 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 "B0 M" are displayed on 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. (B0 0-32K; B1 32-64K; B2 64-96K; B3 96-128K; and B4 16-32K RAM).
S	Transfers GRAPHIC 8 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 nnnnnnn (octal).
nnnnnnG RETURN	Transfers control of display processor to program beginning at memory address nnnnn (octal).
Y RETURN or P RETURN	Calls teletypewriter emulation program. After entering emulation program, function key F0 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 8 to system operating mode. (Y = serial, P = parallel)
RUB OUT	Deletes last octal entry from keyboard.

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 arrow (↑ 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 B0, B1, B2, B3, or B4 causes the memory bank selection to be changed to bank 0, 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*	000000-177777	100000-177777	04-07

NOTE

*Addresses in the range of 140000-177777 (pages 4, 5, 6, and 7) for bank 0 correspond to ROM and I/O device registers. Addresses in the range of 140000-177777 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 0 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 8 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 monitor serves as the printout device. Scrolling of data on the display monitor 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.

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.

The emulation program is entered from the monitor level by typing the letter "Y" or "P" followed by RETURN.* Full-duplex of 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 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 F0 during teletypewriter emulation causes the 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 8 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.

2.2.2.6 Additional Local Mode Commands. Additional commands that can be used when the GRAPHIC 8 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.

*Y = serial, P = parallel

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 7777.

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 0 command is used to detect the presence and status of all loaded options. Typing 0 followed by RETURN causes the display of the first option loaded. Successively pressing the RETURN key causes the display of all 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.

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 screen) and then halts.
157720	PLUS	Maintenance routine. Graphic controller sets X and Y positions at maximum on-screen positions (upper and right corner) and then halts.
157730	MINUS	Maintenance routine. Graphic controller sets X and Y positions at minimum on-screen positions (lower left corner) and then halts.
157740	LOADER	Enables a file to be loaded into read/write memory from various input devices.
157750	MONITOR MODE	Enables local mode commands to be executed.
157760	SYSTEM MODE	Enables system mode.
157770	TEST PATTERN	Transfers to verification test pattern.

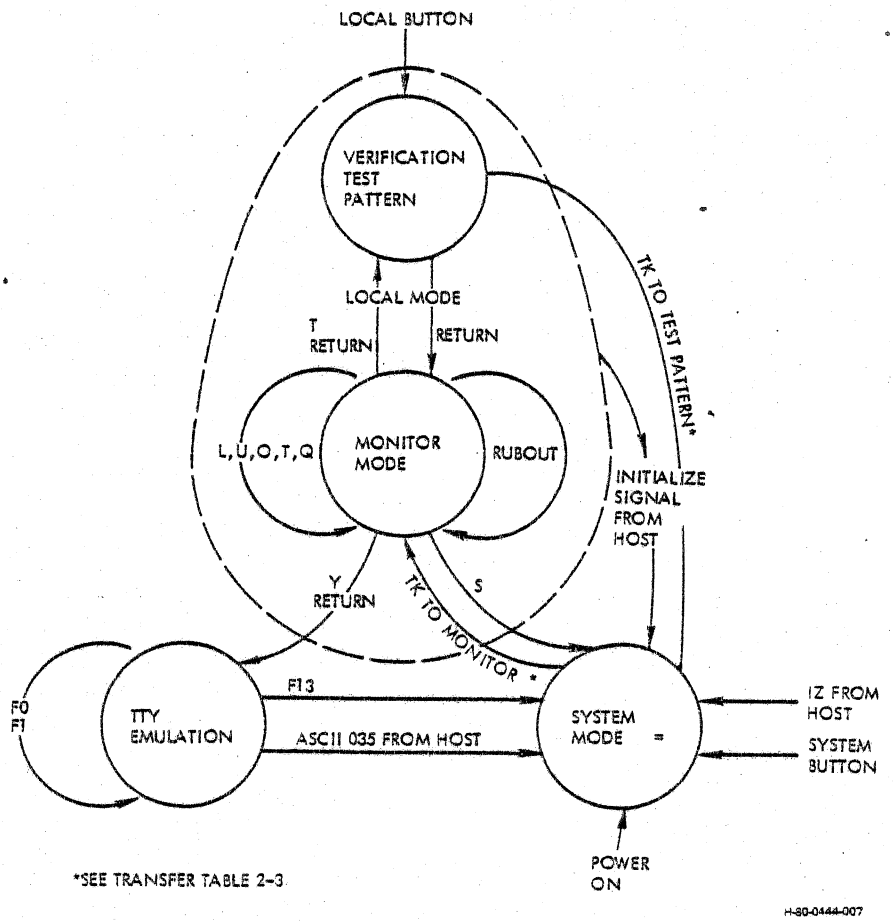


Figure 2-2. Summary of GRAPHIC 8 Operating Modes

SECTION 3

GRAPHIC 8 INSTRUCTIONS

3.1 GENERAL

Instructions used by the GRAPHIC 8 can be divided into two categories: those executed by the display processor and those executed by the digital 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 digital 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

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

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)

Condition Codes:

N: Set if value in Q reg < 0; cleared otherwise

Z: Set if value in Q reg = 0, cleared otherwise

V: Cleared

C: Not affected

3.3 DIGITAL/GRAPHIC CONTROLLER INSTRUCTIONS

The primary functions of the digital graphic controller are to retrieve the display instructions from the refresh file, calculate the addresses of the pixels that are to be illuminated and to write the pixel data into the mapping memory.

The digital graphic controller (DGC) instruction set comprises more than 60 instructions that are used to control the presentation of the image to be displayed. These instructions can be broken down into four basic categories: Pixel position instructions, sequence control instructions, register instructions, and display control instructions. Pixel position instructions determine which pixels will be displayed for the purpose of presenting 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 8 users. These macros can be used to assemble all the graphic controller instructions plus some useful instruction sequences. Refer to Appendix B for description of the macros.

The complete set of instructions and data that defines a particular display image is referred to as a refresh file. An Extended Instruction Mode (EIM) bit permits more instructions than would normally be possible with a given set of opcode bits. The EIM bit is set or cleared by the Load Mode bits instruction, MODE.

For example, 18 bit addressing is used for most sequence control instructions when the EIM bit is on.

An indicator (EIM) appears whenever the instruction being described is for EIM bit on.

Instructions are fetched by the digital 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 digital graphic controller instructions. A summary of the instructions is contained in Appendix A.

Instructions with bit = X means that the bit is ignored by the instruction processor. However, since future systems may utilize these X bits, never use an X bit for information storage.

3.3.1 PIXEL POSITION INSTRUCTIONS. Twenty-five instructions are used to determine the intensified Pixels in the display monitor. These instructions include load, move, draw, text, and conic instructions.

3.3.1.1 Load Instructions. Load instructions specify X-axis positions on the screen in terms of absolute data (specific coordinates) or relative data (lengths along X axis).

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

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

Bits 0-10 define absolute X-axis coordinate in two's complement form. These bits (sign extended) replace contents of X position register.

Operation: (DXR) ← X COORDINATE

|LDXR| LOAD X RELATIVE Octal code: 024ddd

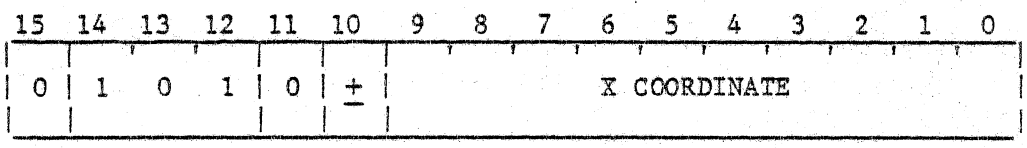
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	±	X INCREMENT									

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.

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

3.3.1.2 Move Instructions. Move instructions specify X- and/or Y-axis new pixel positions on the CRT screen in terms of absolute data (specific coordinates) or relative data (lengths along X and/or Y axis). 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.

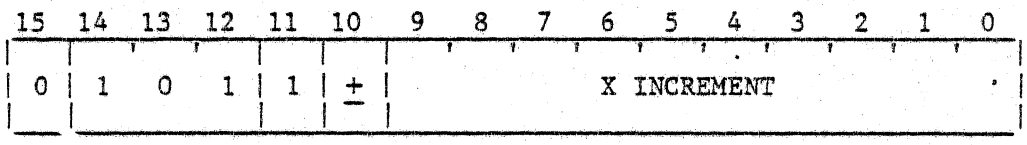
|MVXA| MOVE X ABSOLUTE Octal code: 05dddd



Bits 0-10 define absolute X-axis coordinate in two's complement form. These bits (sign extended) replace contents of X position register. NOTE: Mode 0 only

Operation: (DXR) ← X COORDINATE

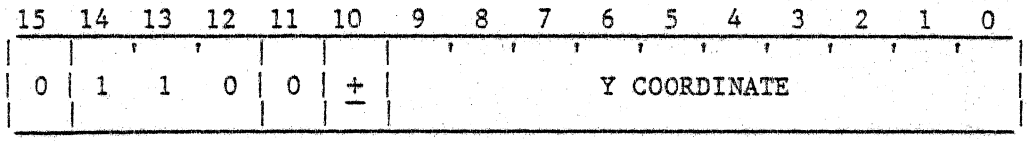
|MVXR| MOVE X RELATIVE Octal code: 054ddd



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. NOTE: Mode 0 only

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

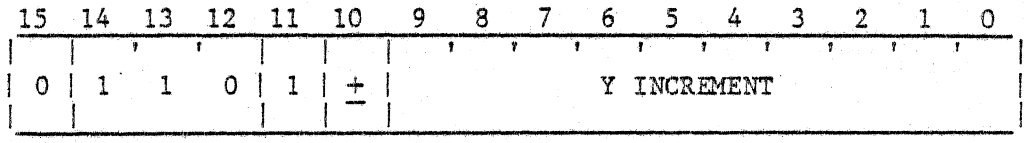
|MVYA| MOVE Y ABSOLUTE Octal code: 06dddd



Bits 0-10 define absolute Y-axis coordinates in two's complement form. These bits (sign extended) replace contents of Y position register.

Operation: (DYR) ← Y COORDINATE

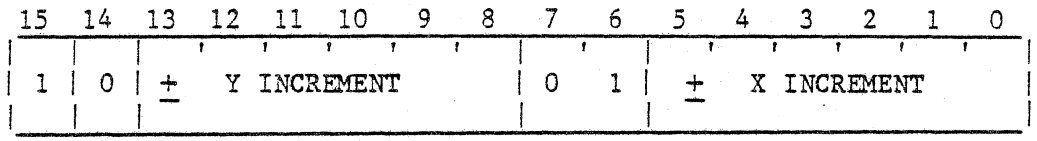
|MVYR| MOVE Y RELATIVE Octal code: 064ddd



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.

Operation: (DYR) ← Y INCREMENT

|MVSR| MOVE SHORT RELATIVE Octal code: 1dd1dd

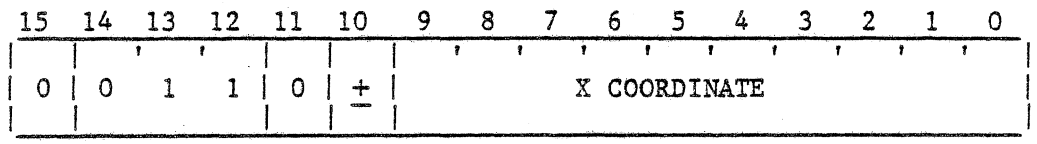


Bits 0-5 define increment in X-axis coordinates; bits 8-13 define increment in Y-axis coordinate. Both sets of bits are in two's complement form. These bits are added (sign extended) to contents of X and Y position registers, respectively.

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

3.3.1.3 Draw Instructions. Draw instructions are similar to move instructions (paragraph 3.3.1.2) except that they cause appropriate pixels to be illuminated from the current X and Y positions to the specified X and/or Y positions. A point plot relative instruction is also included in the draw instructions. This instruction defines a dot represented by an intensified pixel.

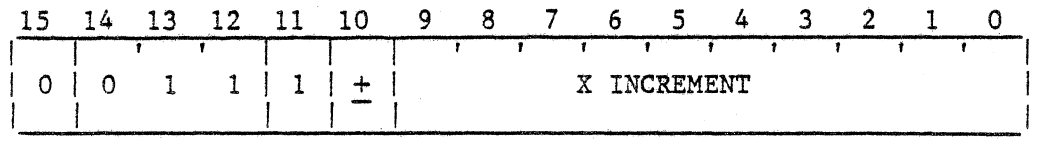
|DRXA| DRAW X ABSOLUTE Octal code: 03dddd



Bits 0-10 define absolute X-axis coordinate in two's complement form. These bits (sign extended) replace contents of X position register. NOTE: Mode 0 only

Operation: (DXR) ← X COORDINATE

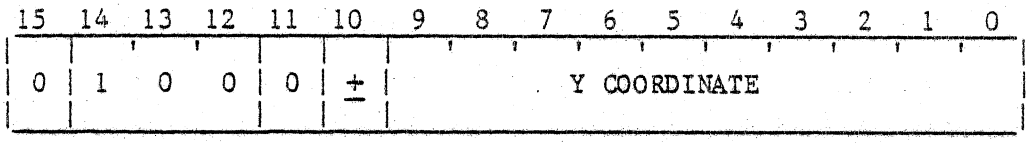
|DRXR| DRAW X RELATIVE Octal code: 034ddd



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. NOTE: Mode 0 only

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

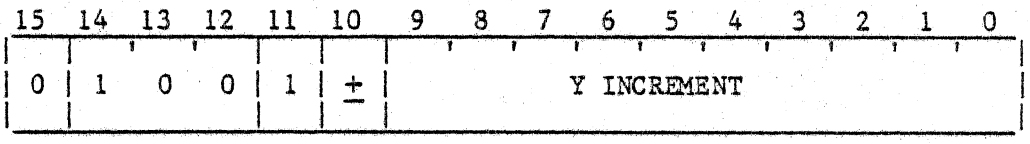
|DRYA| DRAW Y ABSOLUTE Octal code: 04dddd



Bits 0-10 define absolute Y-axis coordinate in two's complement form. These bits (sign extended) replace contents of Y position register.

Operation: (DYR) ← Y COORDINATE

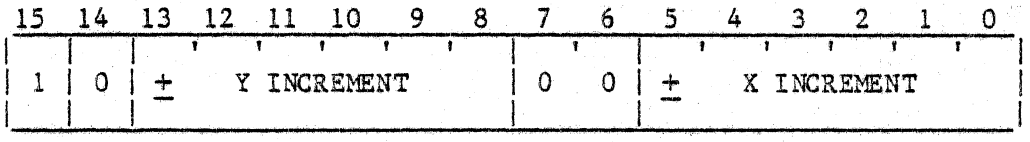
|DRYR| DRAW Y RELATIVE Octal code: 044ddd



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.

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

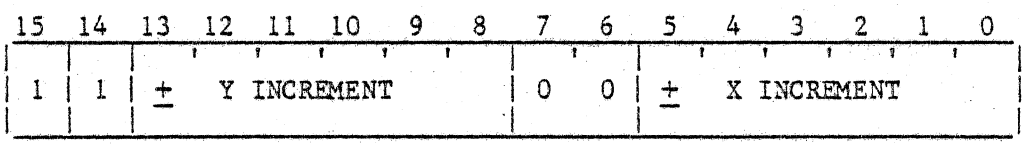
|DRSR| DRAW SHORT RELATIVE Octal code: 1dd0dd



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.

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

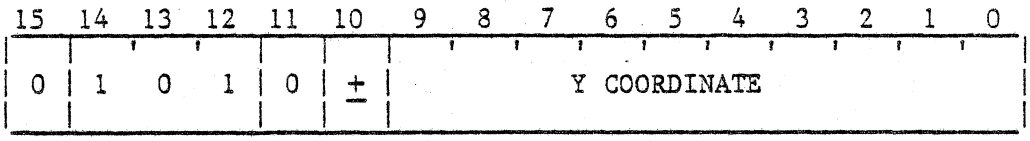
|PPLR| POINT PLOT RELATIVE Octal code: 1dd0dd



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. A dot is displayed as one intensified pixel.

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

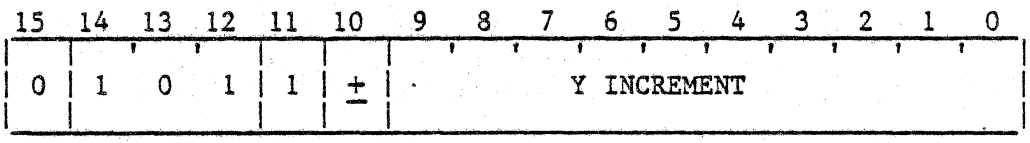
|PPYA| POINT PLOT Y ABSOLUTE (EIM) Octal code: 05dddd



The Point Plot Y Absolute instruction provides the ability to point plot along the Y direction without the need for the MVYA instruction. The Point Plot is displayed as one pixel. Normally, LDXA precedes this instruction.

Operation: (DYR) ← Y COORDINATE

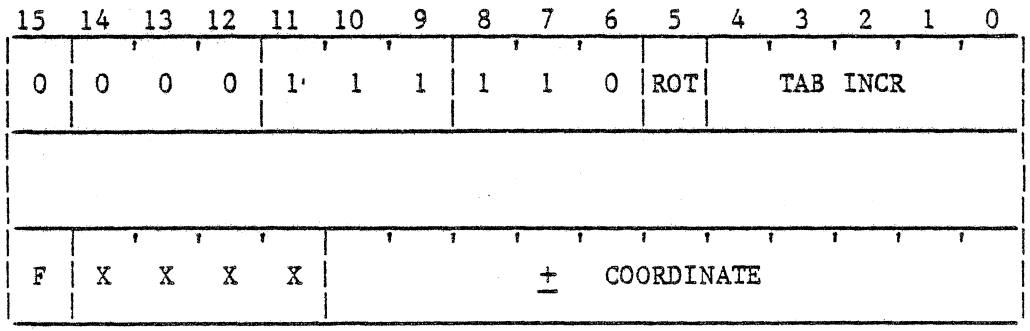
|PPYR| POINT PLOT Y RELATIVE (EIM) Octal code: 054ddd



The Point Plot Y Relative instruction provides the ability to point plot in the Y direction without the need for the MVYR instruction. The plotted point is displayed as one pixel.

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

|PPTA| POINT PLOT TABULAR ABSOLUTE (EIM) Octal code: 0076dd



The PPTA instruction specifies a sequence of X or Y coordinate depending on the orientation.

Bits 0-4 represents the X or Y Tabular Increment between successive points.

Bit 5 = 0 indicates horizontal orientation. Coordinates are Y values.
= 1 indicates vertical orientation. Coordinates are X values.

Bit 15 (in the data words)
= 0 indicates an intermediate point.
= 1 indicates the final plotted point and the end to the variable length instruction.

Operation:

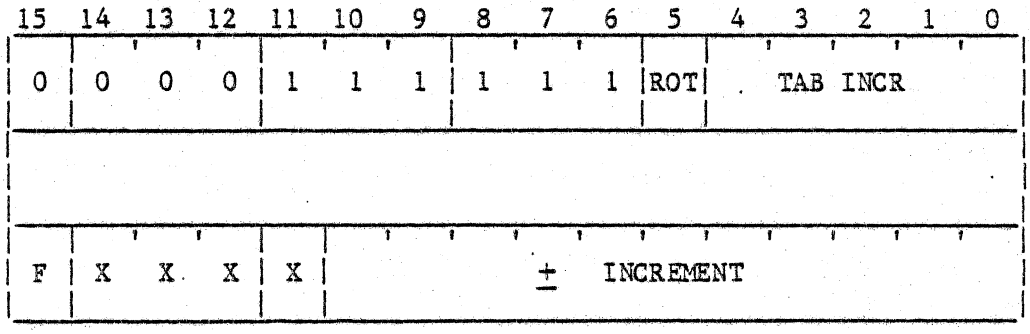
For horizontal orientation: (ROT = 0)

(DXR) ← (DXR) + tabular increment
(DYR) ← Coordinate

For vertical orientation: (ROT = 1)

(DYR) ← (DYR) + tabular increment
(DXR) ← Coordinate

PPTR POINT PLOT TABULAR RELATIVE (EIM) Octal code: 0077dd



The PPTA instruction specifies a sequence of X or Y increment depending on the orientation.

Bits 0-4 represents the X or Y Tabular Increment between successive points.

Bit 5 = 0 indicates horizontal orientation. Increments are Y values.
= 1 indicates vertical orientation. Increments are X values.

Bit 15 (in the data words)
= 0 indicates an intermediate point.
= 1 indicates the final plotted point and the end to the variable length instruction.

Operation:

For horizontal orientation: (ROT = 0):

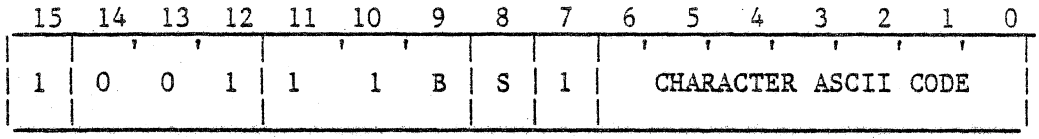
(DXR) ← (DXR) + tabular increment
(DYR) ← (DYR) + increment

For vertical orientation: (ROT = 1):

(DYR) ← (DYR) + tabular increment
(DXR) ← (DXR) + increment

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 position. The second instruction enables two characters to be drawn and the 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 X, Y position are determined by display control instructions (paragraph 3.2.4).

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



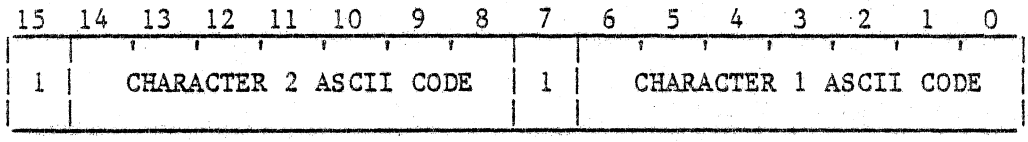
Bits 0-6 specify ASCII code of character to be drawn at location defined by contents of X and Y position registers. Bit 9 specifies whether the character is to blink or be steady (1 indicates steady). Bits 0-6 replace contents of character register. Bit 8, the shift bit, indicates either the standard character set (1) or the extended optional character set (0). This shift bit, applies only to the character specified in the CHAR instruction. It has no effect on subsequent text instructions. Bit 9 replaces blink bit in display Z register. After a character is presented, the starting location is returned in X and Y position registers and the previous state of the blink bit is restored in the display Z register.

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

NOTE

On customer systems containing special characters or special symbols, the shift out code (character code octal 16) can 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 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 the contents of character register. The character is presented. 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 the contents of text increment register. If ASCII code for a NULL character is specified, no character is drawn and the X (or Y) position register is not incremented.

Operation: Draw character 1 at position DXR, DYR

(DXR) ← (DXR) + (DTI) [(DYR) ← (DYR) + (DTI) for rotated text]

Repeated above for character 2

Three control characters are used in the CHAR or TXT instruction to determine text position on the screen (see table A-9).

The STX character defines the left margin of screen as the present X position (Y position for rotated text).

STX Operation: (LMR) ← (DXR)
[(LMR) ← (DYR) of rotated text]

The Carriage Return CR character sets the X position (Y position for rotated text) to the previously defined left margin.

CR Operation: (DXR) ← (LMR)
[(DYR) ← (LMR) for rotated text]

The Line Feed LF character decrements the current Y position (or increments the current X position for rotated text) by the amount previously defined as the line increment (see the LDTI instruction).

LF Operation: (DYR) ← (DYR) - (DLI)
[(DXR) ← (DXR) + (DLI) for rotated text]

The following information describes the RAM character generator.

Summary of features:

1. The font size of the standard character set determines the font size which must be used in the extended and RAM character sets. Vertical misalignment of the characters will result if this condition is not met.

2. In the RAM set, 32 character codes (000_8-037_8) are reserved for control functions; 96 character codes (040_8-177_8) are for displayable characters.

3. Multiple character tables are possible.

4. All standard character sizes and rotation apply to RAM characters.

5. ASCII code 033_8 is the control code used to enter the RAM character set.

6. The base address for the RAM table must be loaded prior to using RAM characters for the first time. Reloading is necessary only when multiple tables are in use.

7. The "building" of the RAM character table must conform to the guidelines that follow.

8. The character table must be based at 4K boundaries and may be placed in any bank of memory. Care must be taken not to violate any reserved areas of memory in bank 0.

9. RAM Control Characters

   NULL (000_8) performs same function as in standard and extended sets.

   SO (016_8) exits RAM character set and enters entended set.

   SI (017_8) exits RAM character set and enters standard set.

Guidelines

The RAM address of a specific character is determined by shifting the 7-bit ASCII code four places to the left, filling the four least significant bits with the LSB of the ASCII code, and "oring" this value with the base address loaded by the LDCG (load character generator) instruction (see page 3-23). The equation to calculate the RAM address of a character in decimal is:

$ASCII_{10}$ + base address$_{10}$ (+15 if $ASCII_{10}$ is odd) = address of RAM character

e.g.,

```
        1001000 = ASCII code
           ↓
0000010010000000        Modified ASCII code
           ↓
0010010010000000        RAM address assuming base address of 20000₈.

Base
address
```

The digital graphic controller microcode checks the LSB of the RAM address to determine the direction to proceed through the RAM. If the address is even, the digital graphic controller increments the RAM address by 1 to fetch the succeeding character data. If the address is odd, the digital graphic controller decrements the RAM address. This implies that the character definition is referenced to the initial character address calculated. An example of how the L and M characters would appear in a RAM table based at 20000_8 using their assigned ASCII codes is shown below.

1001100	=	L character ASCII code
0010010011000000	=	RAM address for L character 22300_8
1001101	=	M character ASCII code
0010010011011111	=	RAM address for M character 22337_8

The maximum width of a character is fixed at 7 pixels. Bit 0 of the 8-bit byte is a flag that informs the digital graphic controller that the character is complete. Character length may be up to 16 pixels and may be shifted as in the case of lower case characters.

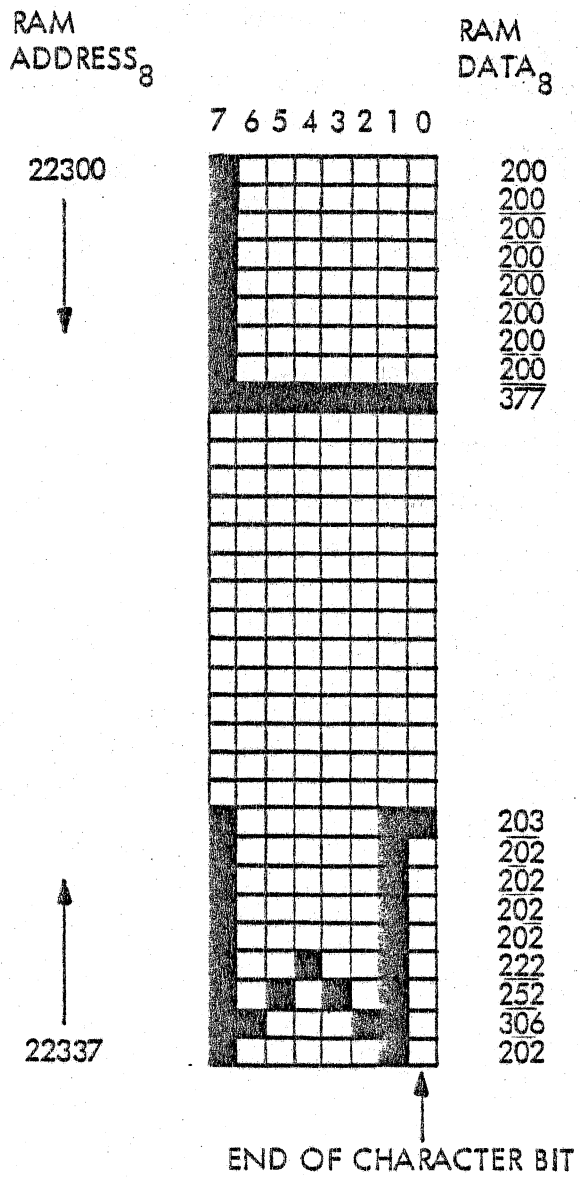
The load character generator address instruction loads the base address of the RAM character set. The character table must be based at 4K boundaries in memory.

Allowable character table base addresses (Bank Ø):

4000	44000	104000
10000	50000	110000
14000	54000	114000
20000	60000	120000
24000	64000	124000
30000	70000	130000
34000	74000	134000
40000	100000	

Example of a refresh file using the RAM character set:

```
        .
        .
        .
        .
   4700 .          Load character table address
  30000  .         Base address
 117601            Enter RAM character set
 140301 }
 174201 }          RAM characters
 100277 }
 117600            Exit RAM character set
        .
        .
        .
        .
 117601            Enter RAM character set
 112217 }
 130711 }          RAM characters
 141674 }
 117600            Exit RAM character set
        .
        .
        .
        .
```



Example of Two 7 X 9 Characters in RAM

3.3.1.5 Conic Instructions. Two conic instructions are used to specify 90-degree segments and axis lengths of ellipses to be displayed. One is a load instruction that specifies X-axis data. The second is a draw instruction that specifies Y-axis data and presents the ellipse. 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 alone 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 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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	QIII	QI	X SEMI-AXIS LENGTH								

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 unblanking (1 indicates unblank) in quadrants QI (upper right) and QIII (lower left), with respect to the major and minor axes (if rotated) of the ellipse. All of bits 0-10 replace contents of X conic register.

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

DRKY — DRAW CONIC Y — Octal code: 07dddd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	QIV	QII	Y SEMI-AXIS LENGTH (OR CIRCLE RADIUS)								

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 unblanking (1 indicates unblanking) in quadrants QII (upper left) and QIV (lower right), with respect to the major and minor axes (if the ellipse is rotated). All of bits 0-10 replace contents of conic Y register. The ellipse is drawn as defined in conic X and Y registers with ellipse center at location defined by X and Y position registers. The ellipse 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 created. Bits 9 and 10 then specify 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

3.3.2 SEQUENCE CONTROL INSTRUCTIONS. Thirteen instructions are used to control the sequence of program execution and timing by the digital graphic controller. These instructions include unconditional jump, conditional jump, subroutine, linkage, halt, wait, and update instructions.

The JUMP, JMPZ, JPRZ, CALL, CALR and RTRN utilize an 18-bit address if the EIM bit is on. In this case it is recommended that you use the extended form of the mnemonic for clarity.

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(E) JUMP Octal code: 0010Xd 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	A17	A16
I (A15)	A14	A13	.	.	JUMP ADDRESS						.	.	A2	A1	A0

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 (for EIM = 0) to which program control is to be transferred. The JUMPE, jump extended address instruction, utilizes bits 0-1 of the first word as bits A16-A17 of the extended 18 bit address. Also, when in extended mode bit 15 represents address bit 15 (A15) not the indirect bit I.

The specified address may be the address of any even-numbered byte from 00000 to 777776 (octal). If bit 15 of the second word is set to 0 (direct addressing) and EIM = 0, control is transferred to the address specified by bits A0-A14. If bit 15 is a 1 (indirect addressing) and EIM = 0, bits A0-A14 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 777776 (octal). Multi-level indirect addressing may be used.

Operation: Direct: (DPC) <— JUMP ADDRESS
Indirect: (DPC) <— (JUMP ADDRESS)

|JRMP(E)| JUMP RELATIVE Octal code: 0011XX 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	1	X	X	X	X	X	X
±	A14	.	.	JUMP INCREMENT (IN EVEN BYTES)									.	.	A0

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 digital graphic controller. Relative jumps from -100000 to 77776 (octal) bytes can be accomplished using this instruction.

If EIM = 1, the JRMP(E) instruction utilizes 18-bit addressing. If the increment causes the 16-bit program counter to overflow, the bank register PGR is incremented or decremented appropriately.

Operation: (DPC) <— (DPC) + JUMP INCREMENT

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

or

|NOOP|

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

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 digital graphic controller. Relative jumps from -400 to 376 (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(E) JUMP IF DISPLAY REGISTER 0 CONTENTS ≠0 Octal code: 0012X 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	A17	A16
I (A15)	A14	A13	.	.	JUMP ADDRESS						.	.	A2	A1	A0

JMPZ is a two-word instruction used for the conditional transfer (direct or indirect) or 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 JMPZE, conditional jump extended address instruction, uses bits 0-1 of the first word as bits A16-A17, respectively, of the extended 18 bit address. Note that for EIM = 1 bit 15 represents the address bit (A15) not the indirect bit I.

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 and EIM = 0 (direct addressing), control is transferred to the address specified by bits A0-A14. If bit 15 is a 1 and EIM = 0 (indirect addressing), bits A0-A14 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(E) instruction (this is the address to which the program counter is pointing). Note that direct addressing cannot be used for addresses greater than (7) 77776 (octal). Multilevel indirect addresssing 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(E)| 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
±	A14	.	.	JUMP INCREMENT (IN EVEN BYTES)									.	.	A1	A0

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 (DR0) 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 (DR0) ≠ 0. When (DR0) ≠ 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 (DR0) = 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 from -100000 to 77776 (octal) bytes can be accomplished using this instruction.

If EIM = 1, then the JPRZE instruction utilizes 18-bit addressing. If the increment causes the 16-bit program counter to overflow, the bank register PGR is incremented or decremented appropriately.

Operation: (DR0) ≠ 0: (DPC) ← (DPC) + JUMP INCREMENT
(DR0) = 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). The digital graphic controller is capable of calling subroutines between different memory banks by the use of the extended address instructions CALLE. A jump-and-mark instruction is also included which permits direct or indirect calls to be made to subroutines at specific memory locations.

|CALL(E)| CALL SUBROUTINE Octal code: 0021Xd 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	A17	A16
A15	A14	.	.	SUBROUTINE ADDRESS									.	.	A2	A1	A0

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 CALLE, call extended subroutine address instruction, utilizes bits 0-1 of the first word as bits A16-A17, respectively, of the extended 18 bit subroutine address.

The specified address may be the address of any even-numbered byte from 000000 to 777776 (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. Bits A16-A17 are loaded into the bits 14 and 15, respectively, of the bank register PGR. 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 digital graphic controller.

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

IF EIM = 0, then:
     (DPC) ← SUBROUTINE ADDRESS

ELSE [EIM = 1]
     (DSP) ← (DSP) - 2
     (Top stack location) ← (PGR)
     (DPC) ← SUBROUTINE ADDRESS
END
```

CALR(E) CALL RELATIVE Octal code: 0022XX dddddd

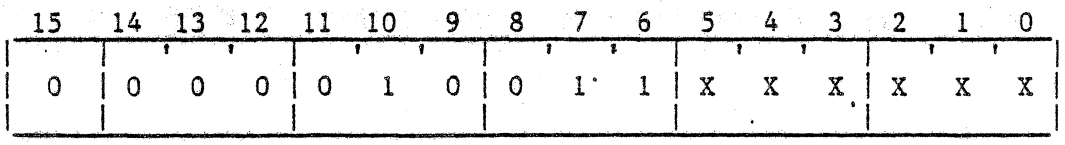
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	1	0	X	X	X	X	X	X
±	A14	.	.	SUBROUTINE INCREMENT (IN EVEN BYTES)							.	.	A1	A0	

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 -100000 to 77776 (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 digital 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.

If EIM = 1, then the CALRE instruction utilizes 18-bit addressing. If the increment causes the 16-bit program counter to overflow, the bank register PGR is incremented or decremented appropriately.

Operation:
```
IF (EIM = 1) then:
     (DSP) ← (DSP) - 2
     (Top stack location) ← (PGR)
     (DSP) ← (DSP) - 2
     (Top stack location) ← (DPC)
     (DPC) ← (DPC) + SUBROUTINE INCREMENT
ELSE [EIM = Ø]
     (DSP) ← (DSP) - 2
     (Top stack location) ← (DPC)
     (DPC) ← (DPC) + SUBROUTINE INCREMENT
END
```

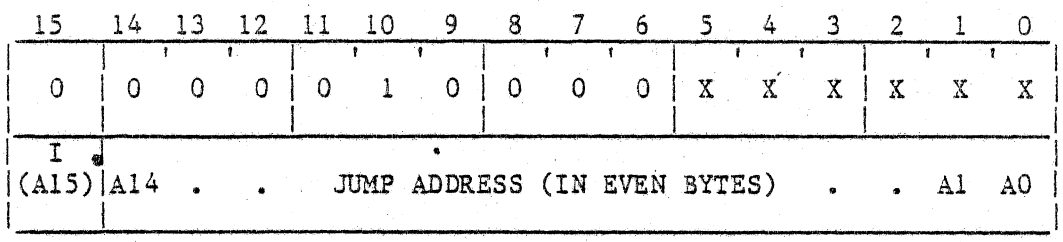
RTRN(E) RETURN Octal code: 0023XX



A RTRN instruction is normally the last instruction of a subroutine called by a CALL(E) 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 digital 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 digital graphic controller.

Operation:
```
IF (EIM = 1) then:
     (PGR) ← (Top stack location)
     (DSP) ← (DSP) + 2
     (DPC) ← (Top stack location)
     (DSP) ← (DSP) + 2
ELSE [EIM = Ø]
     (DPC) ← (Top stack location)
     (DSP) ← (DSP) + 2
END
```

JMPM(E) JUMP AND MARK Octal code: 0020XX dddddd



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 and EIM = 0 (direct addressing), the return address is stored in the location specified by bits 0-14. If bit 15 is a 1 and EIM = 0 (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 followng 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.

Note that for EIM = 1 bit 15 of the second word represents jump address bit (A15) rather than the indirect bit I.

Operation: Direct:

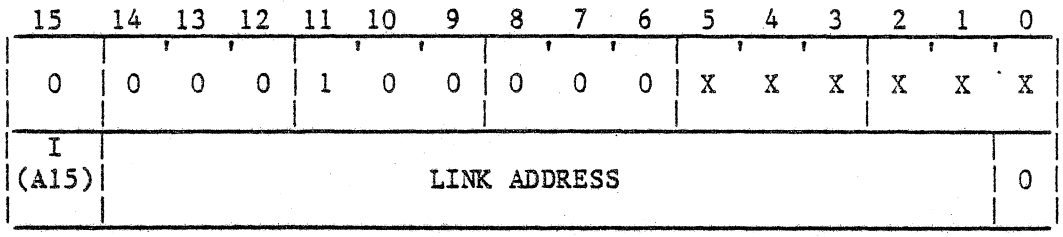
(JUMP ADDRESS) ← (DPC)
(DPC) ← JUMP ADDRESS + 2

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 digital 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(E) SYNCHRONIZED LINKAGE Octal code: 0040XX dddddd



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 bank defined by the bank register. (See PGR in Section 4.) The direct link address is accessed in bank 0.

Note that for EIM = 1 bit 15 of the second word represents the LINK address bit (A15) rather than indirect bit I.

Operation: Direct:

(LINK ADDRESS) ← (DPC)

Indirect:

(Address contained in LINK ADDRESS) ← (DPC)

3.3.2.5 Halt and Wait Instructions. One halt, one wait and update instruction are included in the digital 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 update of mapping memory.

|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 digital graphic controller is controlled by a command from the display processor as described in Section 7.

Operation: Digital 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 control the displaying of a refresh file each time a frame sync pulse is generated. One and only one WATE instruction is used in each refresh file. The mapping memory is cleared each time the WATE instruction is accessed by the digital graphic controller. The mapping memory being displayed is swapped and then the second (non- displayed) mapping memory is cleared when WATE is encountered. The second mapping memory then becomes the displayed memory. This process is repeated each time the WATE instruction is accessed by the digital graphic controller.

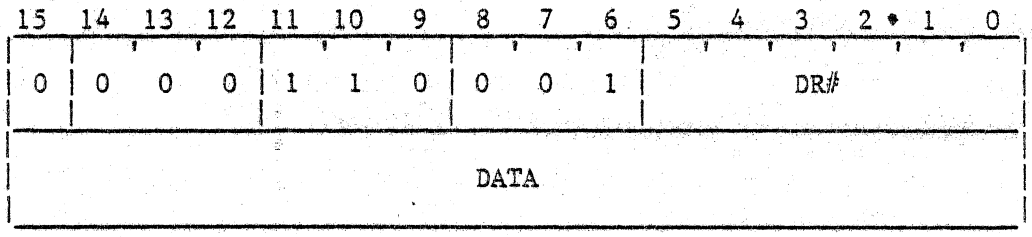
|UPDT| UPDATE VIDEO CONTROLLER REGISTERS (EIM) Octal code: 0073XX

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

The UPDT instruction is used to update the selected video controller registers from the corresponding RAM register without effecting pixel memory. It is used, for example, to display the full screen cross hair cursor, display a split screen, or change video controller status.

3.3.3 REGISTER INSTRUCTIONS. Register instructions are used to modify the contents of general purpose registers of the digital graphic controller (display registers) and to control graphic controller stack operations.

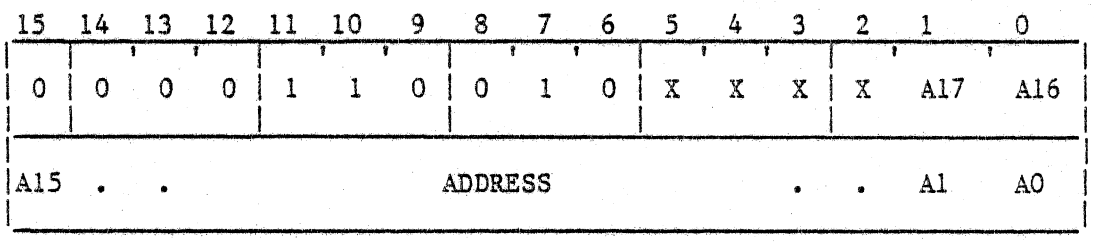
|LDDI| LOAD DISPLAY REGISTER IMMEDIATE Octal code: 0061dd dddddd



LDDI is a two-word instruction used to load data into one of the display registers of the digital 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 to 5 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. See table A-6 for the complete list of registers that can be loaded.

Operation: (DR#) ← DATA

|LDSP(E)| LOAD STACK POINTER Octal code: 0062Xd dddddd



LDSP is a two-word instruction used to load an address into the digital graphic controller stack pointer. The first word identifies the instruction. The second word contains the memory address to be loaded into the digital graphic controller stack pointer.

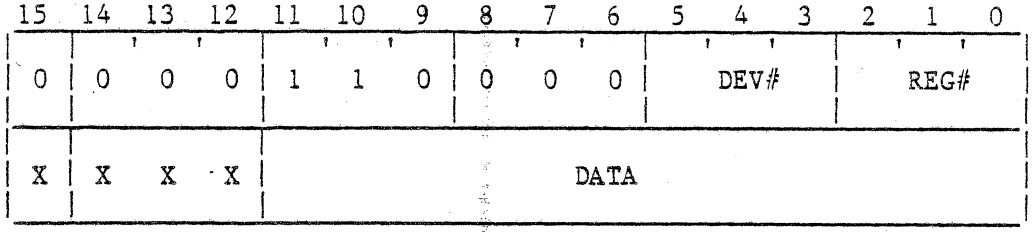
The LDSPE, Load Stack Pointer Extended instruction, utilizes bits 0 and 1 of the first word as bits A16 and A17, respectively, of the 18 bit extended address to be loaded into the stack pointer (for EIM bit on). The graphic controller stack is accessed in the bank defined by the bank register. (See PGR in Section 4.)

Operation: (DSP) ← ADDRESS

NOTE

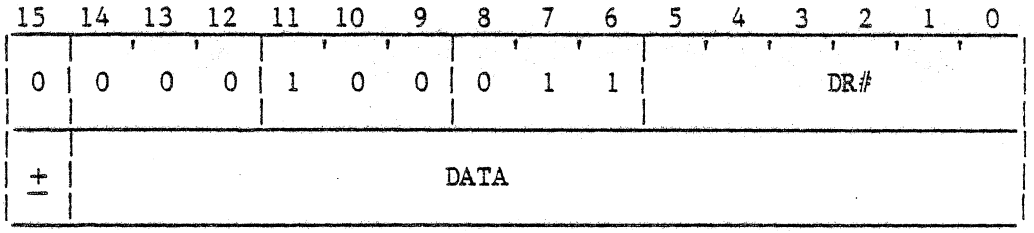
The beginning of stack may be defined in any bank.
The stack must not transcend bank boundaries.

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



LDRI is functionally equivalent to NOOP. It is included for compatibility with the GRAPHIC 7 system.

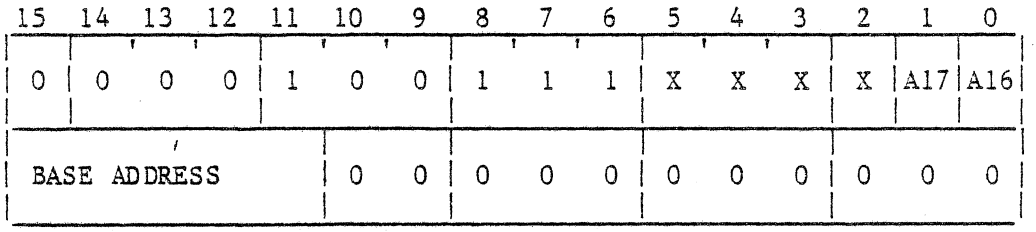
|ADDI| ADD TO DISPLAY REGISTER IMMEDIATE Octal code: 0043dd dddddd



ADDI is a two-word instruction that enables a numerical value to be added to or subtracted from the contents of a display register. The first word 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 0-5 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. See table A-6 for a complete list of display registers.

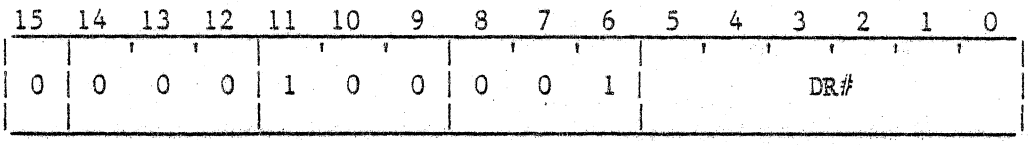
Operation: (DR#) <— (DR#) + DATA

|LDCG| LOAD CHARACTER GENERATOR Octal code: 0047dd ddd000



LDCG is a two-word instruction that loads the base address of the RAM character set. The character table must be based at 4K boundaries in memory.

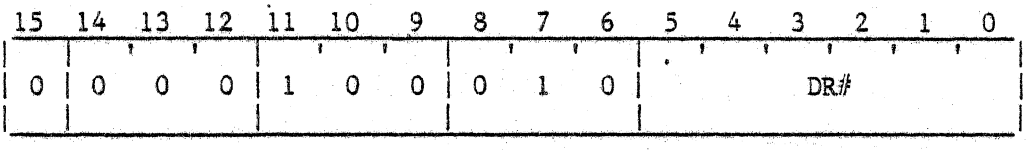
SAVD(E) SAVE DISPLAY REGISTER Octal code: 0041dd



The SAVD instruction causes the contents of the display register identified by bits 0-5 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 0-5 identify the number of the register in binary form. General purpose registers DR0 through DR3 are designated by 00, 01, 10, and 11, respectively. If EIM = 1, then an 18-bit stack address is used.

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

RESD(E) RESTORE DISPLAY REGISTER Octal code: 0042dd

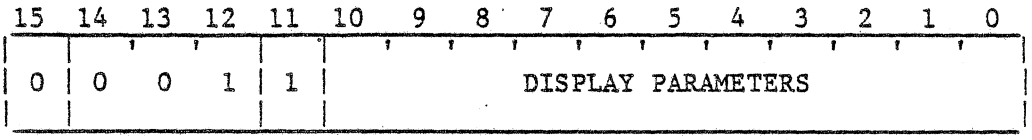


The RESD instruction causes the contents of the top of the graphic controller stack to be popped and placed into the display register identified by bits 0-5. Following the pop operation, the graphic controller stack pointer is incremented by two. Bits 0-5 identify the number of the register in binary form. General purpose registers DR0 through DR3 are designated by 00, 01, 10, and 11, respectively. If EIM = 1, then an 18-bit stack address is used.

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 initialized 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

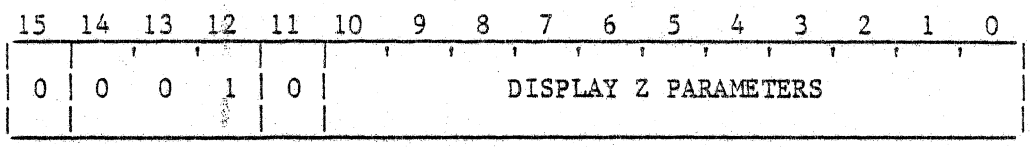


LDDP is used to modify the contents of the display parameter register. The action of individual bits is as follows:

Bit(s)	Action
0,1	Character size:* 1 0 0 0 = size 0 (smallest) 0 1 = size 1 (same as 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
4-10	Not used but are always zero

*See table A-12, page A-70 for character size descriptions.

|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 (mode 0 only; not mode 1) 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)

The gray level is also written into the Pixel Data Register (PDR) (bits 1-7).

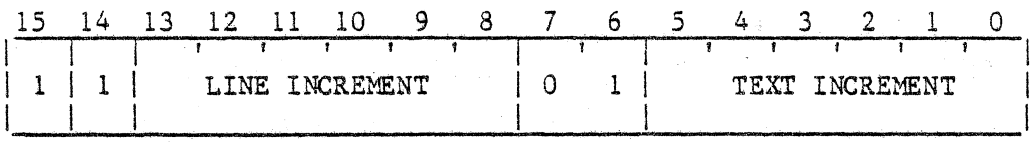
3,4	Line structure select: 4 3 Coordinate units on/off 0 0 = solid vector 0 1 = dotted vector (3 on, 5 off) 1 0 = dashed vector (5 on, 3 off) 1 1 = dot-dashed vector (4 on, 4 off, 20 on, 4 off)
5	Blink select: 0 = steady 1 = blink

The blink bit is also written into the MSB of the Pixel Data Register. The blink rate is 1.5 hertz.

6-9	Display select: 9 8 7 6 1 X X X = Display no. 1 enabled X 1 X X = Display no. 2 enabled X X 1 X = Display no. 3 enabled X X X 1 = Display no. 4 enabled
10	Display select change enable: 0 = no change 1 = change select status to that indicated by bits 6-9

Operation: (DZR) ← DISPLAY Z PARAMETERS

|LDTI| LOAD TEXT INCREMENT REGISTERS Octal code: 1401dd



LDTI is used in conjunction with the TXT (draw two tabular characters) to specify the amount by which each line and each character is incremented. 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 text increment register. The line increment is specified in bits 8 through 13. See table A-12 for the recommended line and text increments for character sizes 1 through 3.

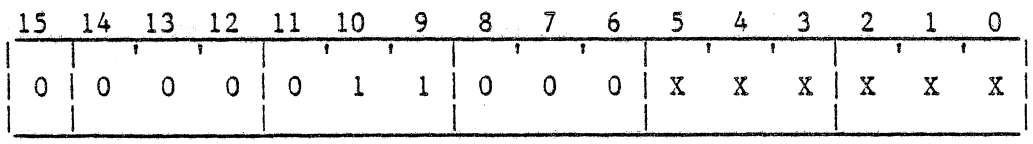
Operation: Normal characters:

(DTI) <- TEXT INCREMENT
(DLI) <- LINE INCREMENT
(DXR) <- (DXR) + TEXT INCREMENT
(followng display of each character)

Rotated characters:

(DTI) <- TEXT INCREMENT
(DLI) <- LINE INCREMENT
(DYR) <- (DYR) + TEXT INCREMENT
(following display or each character)

|IZPR| INITIALIZE Octal code: 0030XX



IZPR is functionally equivalent to a NOOP. It is included for compatibility with the GRAPHIC 7.

|LDPD| LOAD PIXEL DATA REGISTER (EIM) Octal code: 034Xddd

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

The LDPD instruction is used to modify the Pixel Data Register (bits 0-7). See Section 4.3.2 for a description of the Pixel Data Register (PDR).

|MODE| LOAD INSTRUCTION MODE Octal code: 0072dd

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

The MODE is used to modify the Extended Instruction Mode (EIM) bits 0-5. The EIM determines which set of instructions will be recognized by the digital graphic controller with the same opcode. When mode = 0, the standard instruction set is used. When mode = 1, the Extended Instruction Set is used. If bits 1-5 are set, no mode change results. See table A-2 for a summary of these instruction sets.

|INIT| INITIALIZE (EIM) Octal code: 0071XX

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	1	X	X	X	X	X	X

INIT is used to restore the digital graphic controller to power up conditions.

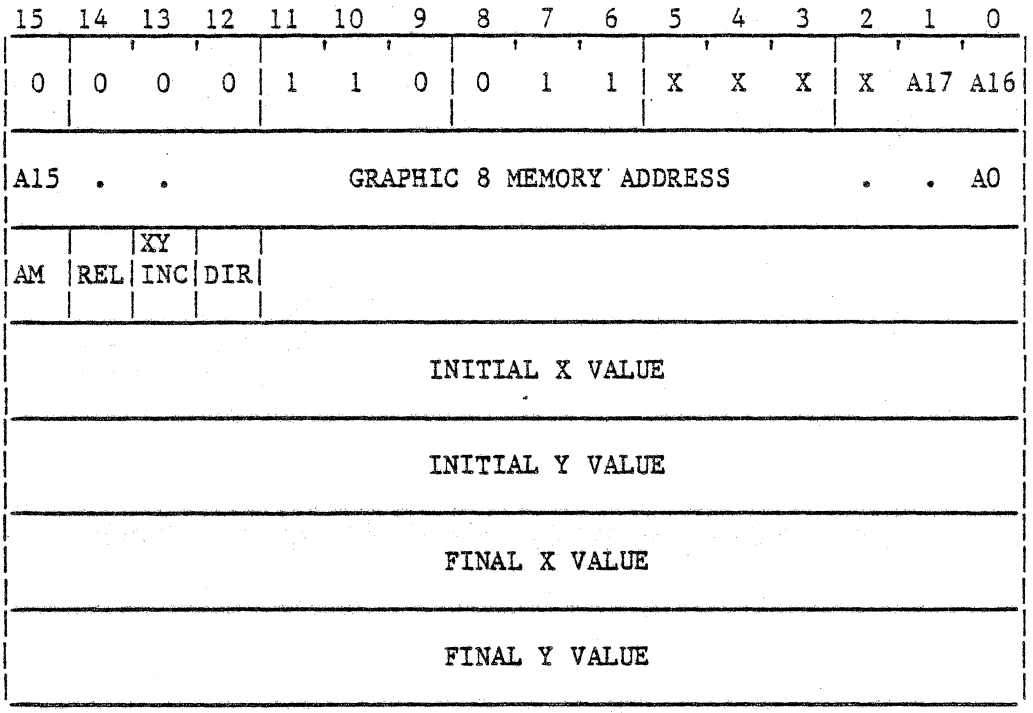
INIT resets the split screen function and crosshair cursor on the video controller, loads the look-up table to initial values (see page 7-21), and selects the color white.

|CLRM| CLEAR MAPPING MEMORY (EIM) Octal code: 067XX

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

CLRM clears the selected mapping memory.

MVPD — MOVE PIXEL DATA (EIM) Octal code: 063dd



MVPD controls the transfer of data between the selected mapping (Pixel) memory and the display processor memory. The data is defined by the GRAPHIC 8 memory address (bits A0 to A17). The Initial X and Y values and the Final X and Y values represent the rectangular array of pixels involved in the transfer.

Word 1	Bits	Description
	0,1	GRAPHIC 8 memory address bits A16, A17
Word 3	12	= 0 Data transfer occurs from mapping memory to GRAPHIC 8 memory
		= 1 Data transfer occurs from Graphic 8 memory to mapping memory

Word 3	Bits	Description
	13	= 0 The XY pixel scan is left to right moving from bottom to top.
		= 1 The XY pixel scan is bottom to top moving across from left to right.
	14	= 0 Initial X and Y values are absolute pixel addresses of the lower left corner of the rectangular pixel array. The final X and Y values are absolute pixel addresses of the upper right corner of the rectangular pixel array.
		= 1 Initial X and Y values represent pixel increments from the current X,Y pixel address to the lower left corner of the pixel array. The final X and Y values are the pixel increments from the current X,Y pixel address to the upper right corner of the rectangular pixel array.
	15	Addressing mode
		= 0 GRAPHIC 8 absolute address
		= 1 GRAPHIC 8 memory displacement from the display program counter (1st word after word 7)

Each pixel array corresponds to an array of consecutive n bit bytes in the GRAPHIC 8 memory. n (= 4,8) is the number of bits per pixel including the MSB or blink bit. The value in the n bit byte represents the gray level of the corresponding pixel.

MDLU — MODIFY LOOK-UP TABLE (EIM) Octal code: 066Xd

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	1	0	X	X	X	X	A17	A16
A15	.	.	GRAPHIC 8 MEMORY ADDRESS										.	.	A0
AM	NUMBER OF GRAPHIC 8 8-BIT BYTES														
0	0	0	0	0	0	0	0	0	0	0	0	VIDEO CONTROLLERS			
0	0	0	0	0	0	0	1	LOOKUP TABLE ADDRESS							

MDLU is used to modify the video controller look-up table. The look-up table defines the manner in which the data in mapping memory is presented to the display without modifying the refresh data itself.

Each video controller contains a 256 x 8-bit word RAM look-up table (LUT) (starting at video controller memory 400 (octal)) which permits pseudo-color or gray level transformations.

The GRAPHIC 8 memory address specifies the beginning of n consecutive 8-bit bytes (where n is the number of GRAPHIC 8 bytes, word 3). If the addressing mode (AM) bit is 0, then this address is absolute. If the (AM) bit is 1, then the value represents a displacement from the program counter (1st word after word 5). The contents of each byte is the desired new gray level (or color) for each corresponding gray level (or color) in mapping memory.

The Video Controller number is assigned as follows:

Controller Number	Word 4, value for bit: 3	2	1	0
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Any combination of controllers can be assigned with the same MDLU instruction by setting any combination of these bits.

The LUT beginning video controller address is 400_8. Addresses greater than 400_8 but less than or equal to 777_8 may be specified to modify a portion of the LUT.

The useable portion of the LUT is a function of the number of bits per pixel and the blink capability of the configuration. If blink is not enabled, then all bits per pixel (number P) are used for pseudo gray level (or color). Therefore, with no blink, the useable portion of the LUT is from 400_8 to $400_8 + 2^P - 1$.

If the configuration has blink, then the MSB is used so only P-1 bits are used for the pseudo gray level (or color). Therefore, with blink, the useable portion of the LUT is from 400_8 to $400_8 + 2^{P-1} - 1$.

For example, suppose the configuration has 4 bits per pixel with blink for video controller number 1. Initially, the LUT looks like:

Video Controller Address (octal)	Contents
400	0
401	1
402	2
---	---
406	6
407	7

We wish to modify the LUT to reverse the roles of gray levels 0 and 7 (reverse video). The MDLU instruction is:

```
.WORD 6600              ; the MDLU opcode
.WORD #ADR              ; G8 memory address or displacement (if AM = 1)
10                      ; # of 8-bit bytes
 1                      ; controller #
 400                    ; LUT address in video controller RAM
```

where

```
ADR:  .BYTE 7, 1, 2, 3, 4, 5, 6, 0
```

NOTE

Location 0 of the LUT defines the background color. Blinking is between the defined color and black, regardless of the background color.

FLPG — FILL A CONVEX POLYGON (EIM) Octal code: 0075XX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	X	X	X	X	A17	A16
A15	.	.	ADDRESS FOR LIST OF VERTICES											.	. A0
AM	REL	NUMBER OF VERTICES													

where the address contains:

X1		ΔX1
Y1		ΔY1
.		.
	OR	
.		.
Xn		ΔXn
Yn		ΔYn

Third word bit 15 = 0 for absolute address of list
= 1 for displacement from the program counter (first word after word 3)

and

bit 14 = 0 for absolute vertices

bit 14 = 1 for relative vertices. Each vertex is relative to the current X and Y position, not to other vertices.

The adjacent vertices are listed in either the clockwise or counterclockwise direction. The last vertex is adjacent to the first in the list. A previous LDDZ or LDPD instruction is used to specify the gray or color level for the filling algorithm.

The fill algorithm is based on the geometric shape of the convex polygon. A convex polygon has all its interior angles less than 180°.

There is no limit to the number of vertices that can be specified except the size of read/write memory. If a non-convex polygon is specified unpredictable results will occur. Maximum execution speed is obtained if the vertices are specified in the clockwise direction starting with Y maximum.

NOTE

The DXR and DYR remain unchanged. The beginning of the vertex list can be in any bank. However, the vertex list must not transcend bank boundaries.

SECTION 4

GRAPHIC 8 REGISTERS

4.1 GENERAL

GRAPHIC 8 registers fall into three major categories: display processor registers, digital 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.

An 8-bit switch register is program readable (used by GCP) from octal location 177774.

4.3 DIGITAL GRAPHIC CONTROLLER REGISTERS

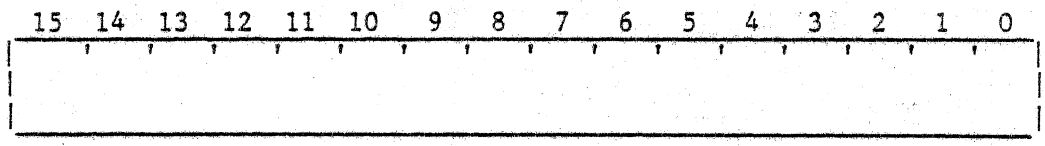
Digital graphic controller registers can be divided into six groups: processor registers, function registers, sense and mask registers, function control registers, display control registers, and configuration register. The following paragraphs provide details concerning the application, format, and address of each register of each group. The RAM register address, RRn, refers to a RAM address in the digital graphic controller. The register is accessible by program control only if it is also assigned an octal memory address: nnnnnn. Refer to Appendix A for a summary of the data applicable to digital graphic controller registers and their display register numbers (DRn).

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



Each of the four general purpose registers (display registers) in the digital graphic controller is a 16-bit register that 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.

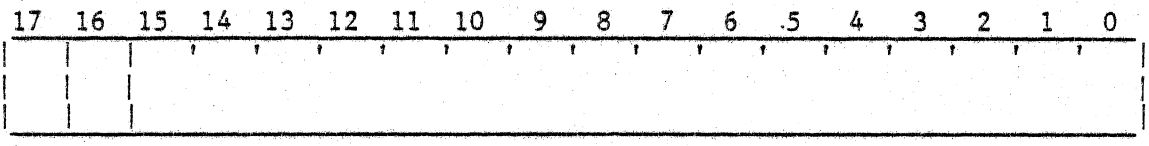
Associated instructions: DR0 thru DR3:

LDDI SAVD
ADDI RESD

DR0 only:

JMPZ JPRZ

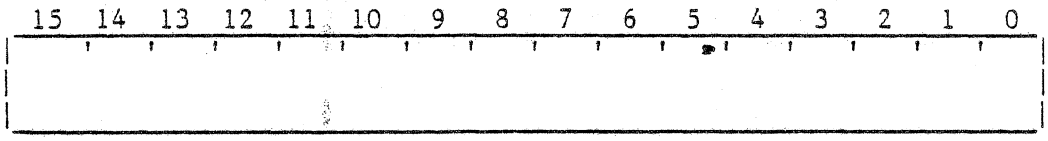
|DSP| STACK POINTER Octal address: 165000



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(E) instruction. When a SAVD(E) 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(E) instruction is used to pop data from the stack, the contents of the stack pointer if 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

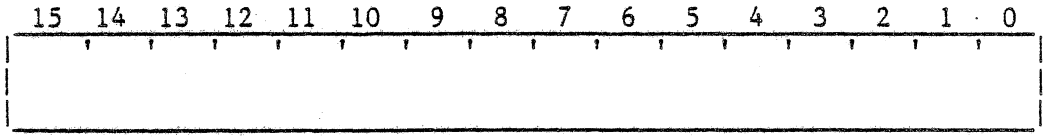
|DPC| PROGRAM COUNTER Octal address: 165006



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 digital 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

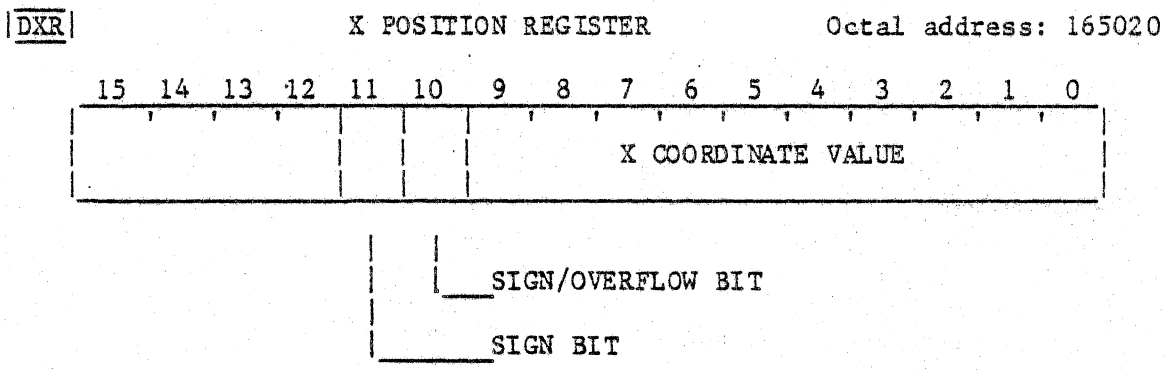
|DIR| DISPLAY INSTRUCTION REGISTER Octal code: 165010



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

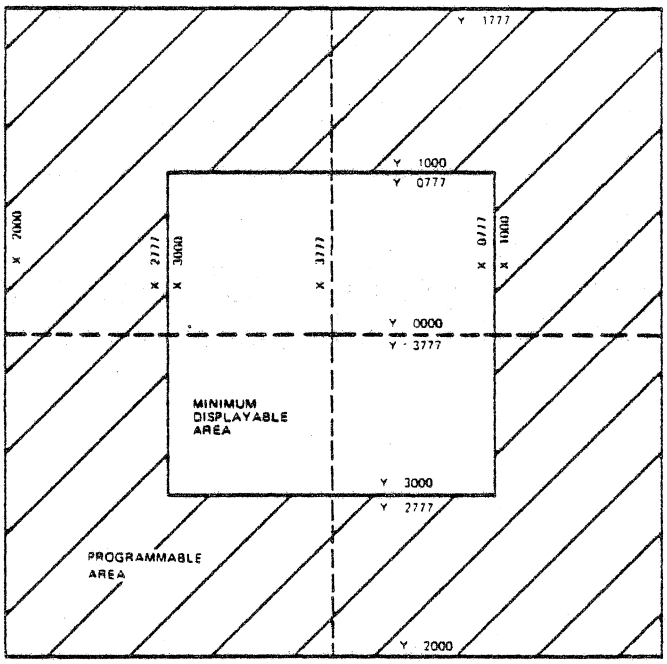
Associated instructions: All

4.3.2 FUNCTION REGISTERS. These function registers are loaded by the digital graphic controller instructions as required to control the functions performed by the microcode. Each function register has an octal address so that, when the digital 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 screen position. When a digital 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 and the interrupt is enabled, 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 screen will be blanked until bits 10 and 11 are no longer different. Coordinate values are expressed to 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 display to be blanked (see figure 4-1).

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



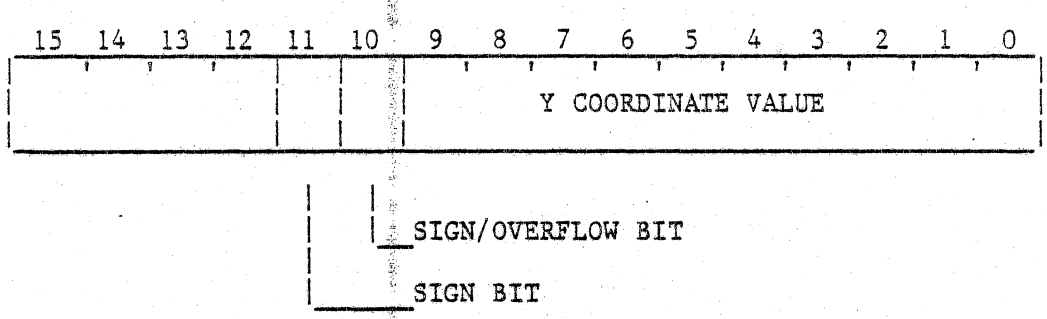
H-80-0483-022

NOTE

Coordinate designations are in octal format.

Figure 4-1. Addressable vs. Displayable Mapping Memory Areas for 1024 x 1024 Screen

DYR — Y POSITION REGISTER — Octal address: 165022

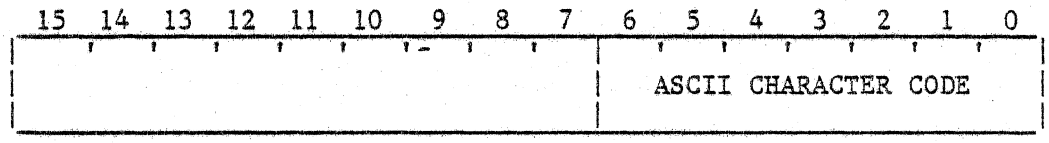


The Y position register is a 12-bit register that contains the value of the Y coordinate of the screen position. This register is identical to the X position register and performs the same functions for Y coordinate data that the X position register performs for X coordinate data. The X/Y overflow bit 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	PPTA	PPTR
DRYR	MVSR	PPYA	PPYR
MVYA	PPLR		
MVYR	TXT (for rotated characters)		

DCR — DISPLAY CHARACTER REGISTER — Octal address: 165024

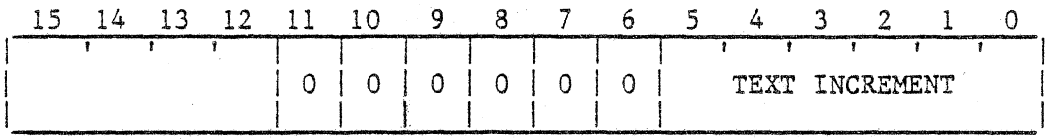


The display character register is a seven-bit register that contains the code of the character or symbol to be displayed. ASCII codes are used for standard and optional characters and symbols as shown in Appendix A.

Associated instructions:

TXT
CHAR

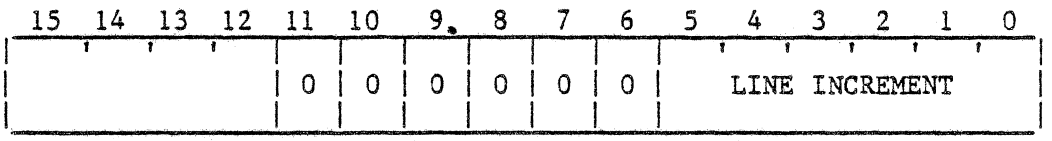
|DTI| TEXT INCREMENT REGISTER Octal address: 165012



The text increment register is a 12-bit register that contains the value by which the screen 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 screen position occurs when the CHAR (draw single character) is used.

Associated instructions: TXT LDTI

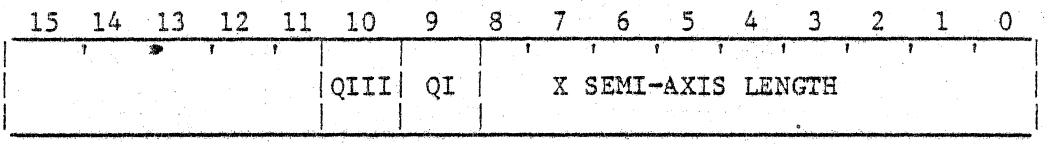
|DLI| LINE INCREMENT REGISTER Ram address: RR126



The line increment register is a 12-bit Ram Register that contains the number screen coordinate units that are advanced after the control character, Line Feed, is encountered in the TXT or CHAR instruction. For horizontally oriented characters, the line increment value is decremented from the Y position register. For vertically oriented characters, the line increment value is incremented to the X position register.

Associated instructions: TXT CHAR LDTI

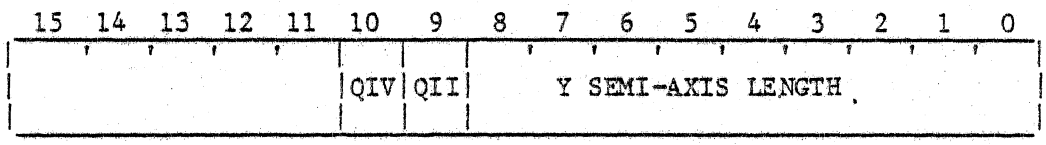
|KXR| CONIC X DATA REGISTER (OPTIONAL) Octal code: 165026



The conic X data register 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 unblanking for quadrants I (upper right) and III (lower left). A zero indicates the image in the quadrant is to be blanked while a one indicates the image is to be unblanked. Loading this register does not change the current screen position. 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.

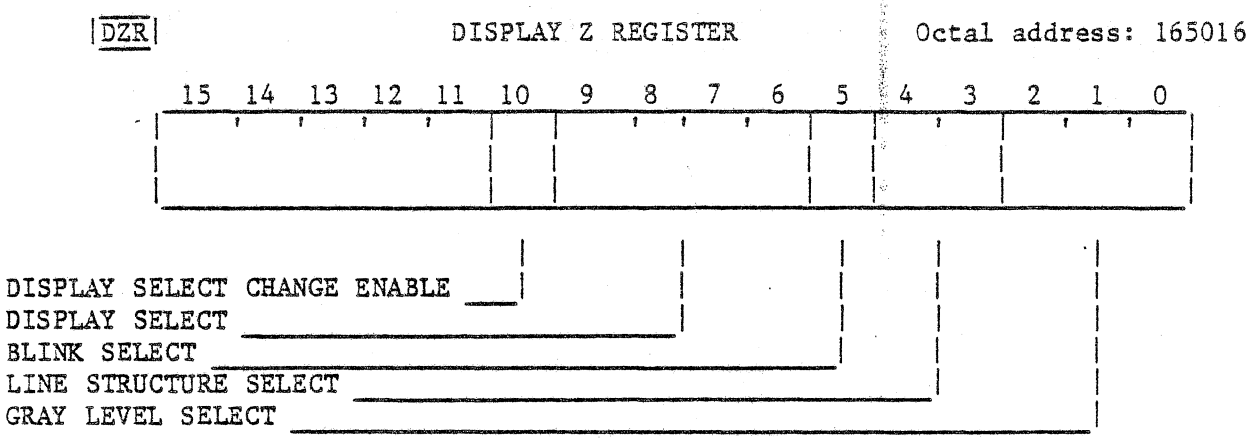
Associated instructions: LDKX
DRKY (when not preceded by LDKX)

|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 unblanking for quadrants II (upper left) and IV (lower right). A zero indicates blanking while a one indicates unblanking. Loading this register changes the current screen position 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



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) This gray level is also written into bits 1 to 7 in the Pixel Data Register (PDR). See the Note.
3,4	Line structure select: 4 3 0 0 = solid vector 0 1 = dotted vector 1 0 = dashed vector 1 1 = dot-dashed vector Screen Coordinates On/Off* 3 ON, 5 OFF 5 ON, 3 OFF 4 ON, 4 OFF, 20 ON, 4 OFF
5	Blink select: 0 = steady 1 = blink This blink bit is also written into the Pixel Data Register as the MSB. The blink rate is 1.5 hertz.

*For a 512 x 512 screen, one pixel represents two screen coordinate units.

Bit(s)	Action
6-9	Display select: 9 8 7 6 1 X X X = Display No. 1 enabled X 1 X X = Display No. 2 enabled X X 1 X = Display No. 3 enabled X X X 1 = Display No. 4 enabled The Display Select bits 9-6 are mapped to bits 0-3 of the Display Select Register (DSR).
10	Display select change enable: 0 = no change 1 = change display select status to that indicated by bits 6-9

Associated instructions: LDDZ

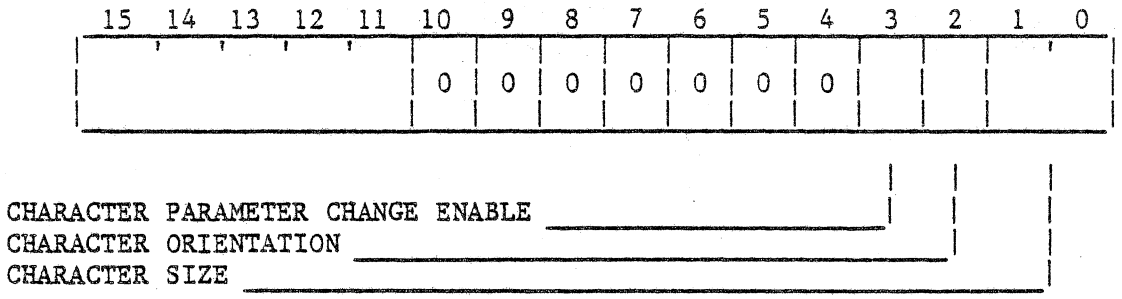
NOTE

The DZR bit 5 (blink) and bits 0-2 are mapped to the PDR register as follows:

Bits per Pixel	7	6	5	4	3	2	1	0
4					5	2	1	0
8	5					2	1	0

In Mode 0 the intensity bits are written through the DZR for compatibility with the GRAPHIC 7 systems.
In Mode 1 the LDPD instruction is used to specify intensities.

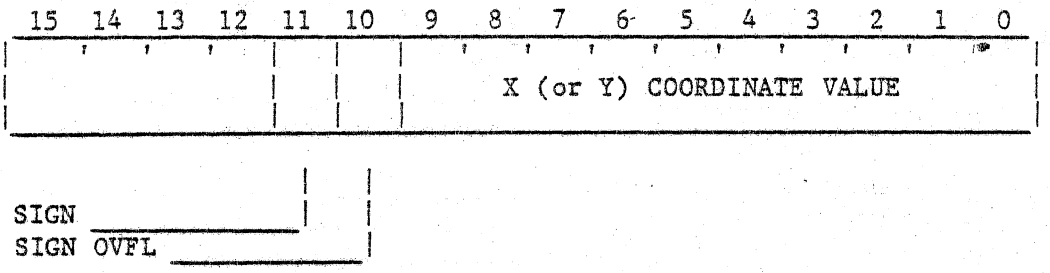
DPR — DISPLAY PARAMETER REGISTER — Octal code: 165014



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: (Also see table A-12) 1 0 0 0 = size 0 (smallest) 0 1 = size 1 (same as 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
4-10	Not used, but are always zero

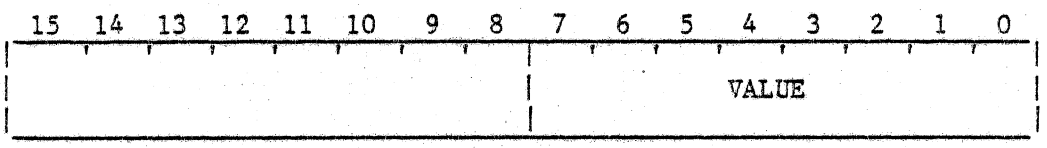
|LMR| LEFT MARGIN REGISTER Ram address: RR108



The left margin register contains the X position value (or Y position value if rotated text) that defines the position at which the next character will be presented whenever a CR control character is encountered. The margin value is stored when an STX character is specified in a CHAR or TXT instruction as the current DXR (or DYR).

Associated instructions: CHAR TXT

|PDR| PIXEL DATA REGISTER Octal address: 165044



The Pixel data register is an eight-bit register that contains a value that specifies an index into the lookup table and the blink status. If blink is enabled, the MSB of this value represents the blink bit. The remaining bits are the index into the lookup table. Bits 8 to 15 are not used. The blink bit is set by the LDDZ instruction only.

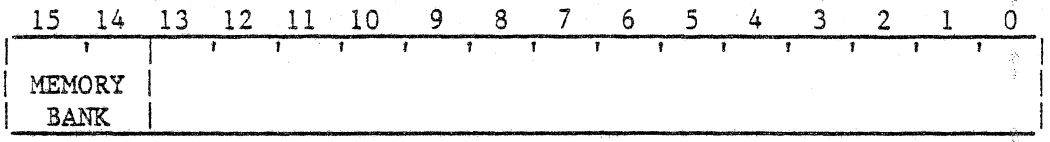
Associated instructions: LDPR

NOTE

The following summarizes the use of the PDR for various bits per pixel and blink (*).

		PDR Bits							
Blink	Bits per Pixel	7	6	5	4	3	2	1	0
Yes	3	.	.	.	.	*	2	1	0
No	4	.	.	.	.	3	2	1	0
Yes	7	*	6	5	4	3	2	1	0
No	8	7	6	5	4	3	2	1	0

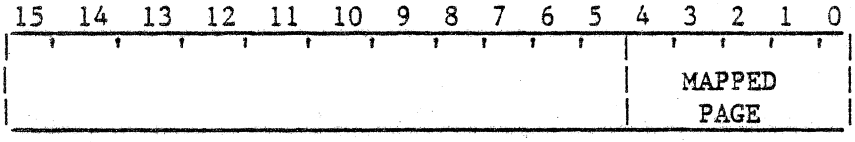
|PGR| GRAPHIC CONTROLLER BANK REGISTER Octal address: 165014



The graphic controller bank 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

|PR1| Octal address: 172342



|PR2| Octal address: 172344

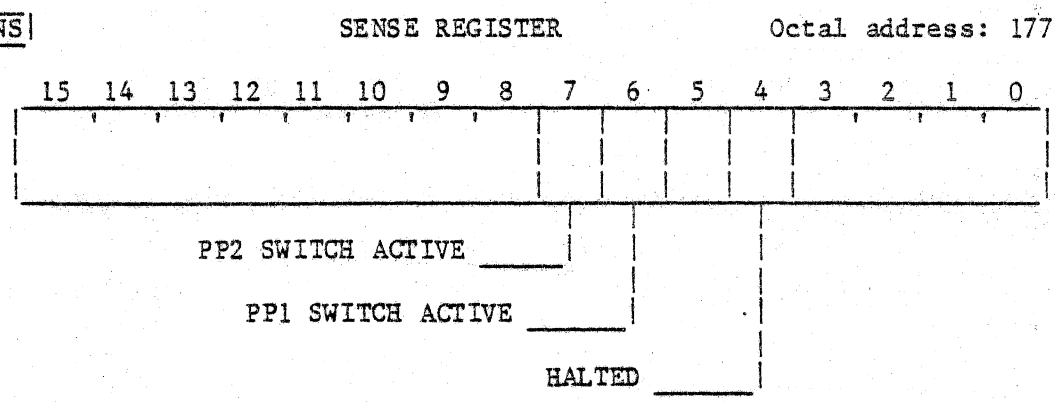
|PR3| Octal address: 172346

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 8.

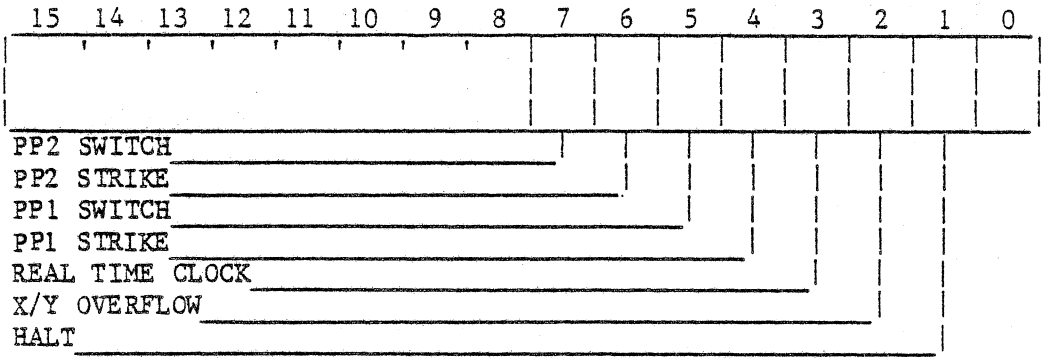
4.3.3 SENSE AND MASK REGISTERS. One sense and one mask register are associated with the digital graphic controller. Both registers are assigned octal addresses and may be read at any time by the display processor using programmed data transfers. Programmed data transfers may also be used by the display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



The sense register is a one-bit register that indicates the halt status of the digital graphic controller. The remaining bits are not used. The condition indicated by bit 4 when it is set to 1 is as follows:

Bit	Condition Indicated When Set to 1
4	Digital 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) Bus timeout (memory fails to respond to a fetch command) 6) 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)
6	PHOTOPEN switch 1 is active
7	PHOTOPEN switch 2 is active

MKR | MASK REGISTER | Octal address: 177662



The mask register is a seven-bit register on the ROM and Status card that enables the digital 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 by HREF instruction	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	PHOTOPEN 1 switch activated	000160
6	PHOTOPEN 2 strike	000154
7	PHOTOPEN 2 switch activated	

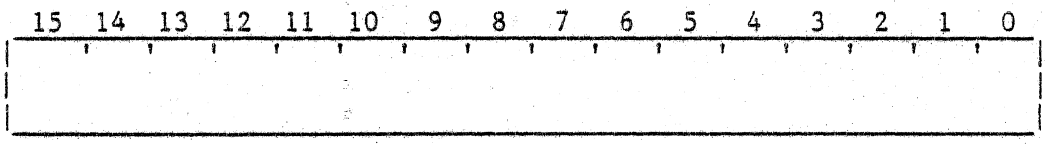
NOTE

Refer to H-82-1319 (Vistagraphic Series PICK Routine User's Manual) for PHOTOPEN usage.

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

4.3.4 FUNCTION CONTROL REGISTERS. Two function control registers are associated with the digital 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.

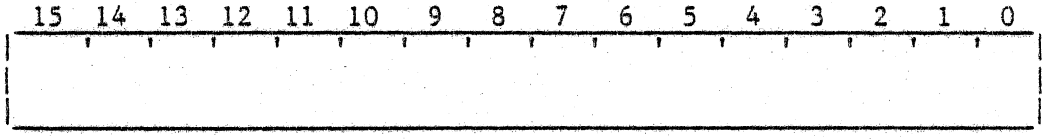
|FUNS| FUNCTION CONTROL STOP REGISTER Octal address: 165040



The function control stop register is a function control register used to halt the digital graphic controller. The HALT bit 4 in the sense register SENS must be checked for the digital graphic controller being halted before proceeding with code that assumes the controller is halted. 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



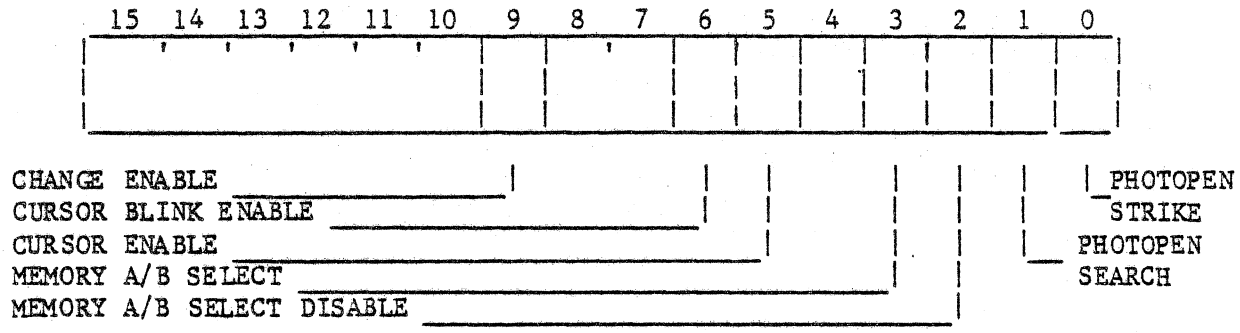
The function control continue register is a function control register used to restart the graphic controller after it has been halted. The HALT bit 4 in the sense register SENS must be checked whether the digital graphic controller has started before proceeding with code that assumes the graphic controller is running. 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

4.3.5 DISPLAY CONTROL REGISTERS. Any combination of four video controllers may be selected by the LDDZ instruction. Twelve display control registers are associated with the selected video controllers. The first register allows blink, cursor, and/or memory to be enabled. Nine registers control split screen definition. Two registers specify the position of the non-destructive crosshair cursor. All these registers are set by the LDDI instruction.

|STAn| VIDEO CONTROLLER STATUS REGISTER

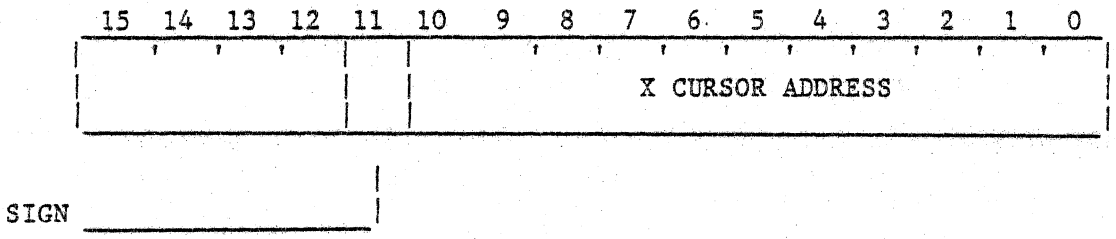
no.	Octal DR #
0	4
1	23
2	42
3	61



The Video Controller Status register contains data that controls the presentation of screen data.

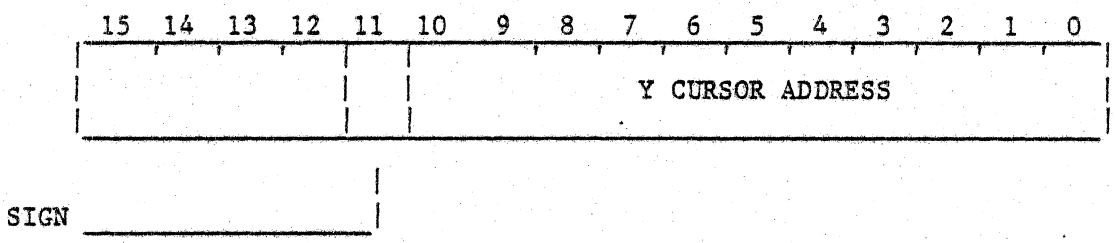
Bit(s)	Value	Action
0	0	No PHOTOPEN strike in last frame
	1	PHOTOPEN strike occurred in last frame
1	0	Set video to reference black
	1	Set video to reference white
2		Disable selection of memory A or B
3	0	Select memory A if bit 2 = 0
	1	Select memory B if bit 2 = 0
4	0	Unused
5	0	Disable crosshair cursor
	1	Enable crosshair cursor
6	0	Disable cursor blink if bit 5 set
	1	Enable cursor blink if bit 5 set Cursor blink rate is 7.5 hertz
7-8		Unused
9	0	Allow only bits 2 and 3 to be written
	1	Allow only bits 5 and 6 to be written

|XCRN| X CURSOR ADDRESS REGISTER Octal DR #21
40
57
76



The X cursor address register defines the X coordinate of the vertical crosshair of the enabled cursor.

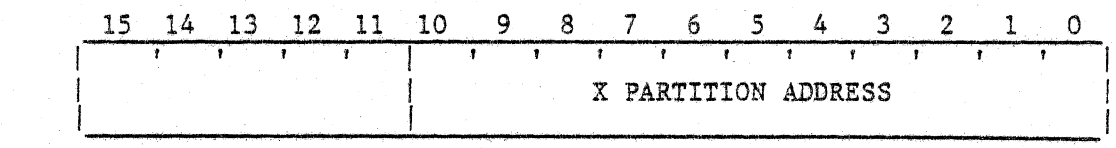
|YCRn| Y CURSOR ADDRESS REGISTER Octal DR #22
41
60
77



The Y cursor address register defines the Y coordinate of the horizontal crosshair of the enabled cursor.

The position of the cursor (intersection of the crosshair) is controlled by the contents of the X Cursor Register XCR and the Y Cursor Register YCR. If the intersection of the crosshair is off screen, but either X or Y is on screen, than the corresponding Y or X crosshair will be displayed on the screen.

|SXns| START X REGISTER Octal DR #: see table below

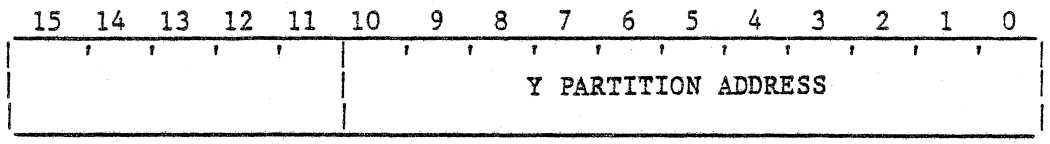


The start X register is an 11-bit register that contains the starting address for the upper left corner of section number(s) for selected video controller (n). The display register numbers (DR #) are defined in the following table.

OCTAL VALUES OF DR # FOR
SELECTED VIDEO CONTROLLER n

		1	2	3	4
screen	1	6	25	44	63
section s	2	12	31	50	67
	3	16	35	54	73

SYns — START Y REGISTER — Octal DR #: see table below

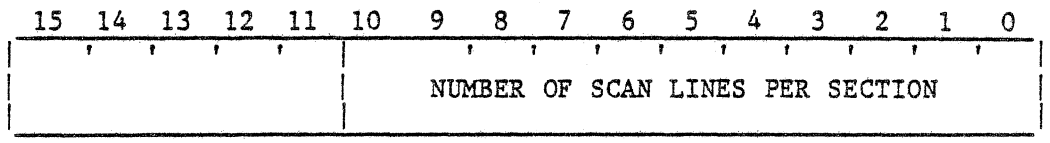


The start Y register is an 11-bit register that contains the starting address for the upper left corner of section (s) for selected video controller number (n). The display register numbers (DR #) are given in the following table.

OCTAL VALUES OF DR # FOR
SELECT VIDEO CONTROLLER n

		1	2	3	4
screen	1	5	24	43	62
section s	2	11	30	47	66
	3	15	34	53	72

LNns — LINE REGISTER — Octal DR #: see table below



The line register is a 11-bit register that contains the number of scan lines per section (s) for selected video controller (n). The display register numbers (DR #) assigned are in the following table. For example, the line register for video controller 3 and screen partition 2 is LN32 with display register DR51.

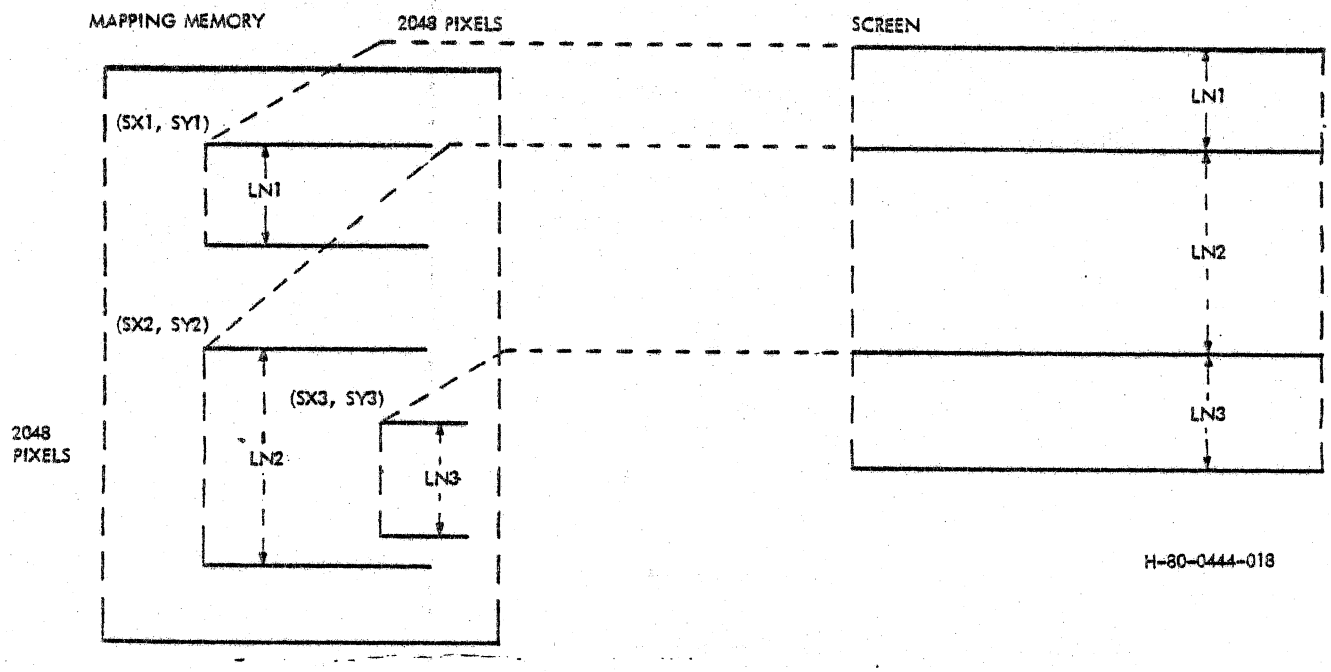
		1	2	3	4
screen	1	7	26	45	64
position s	2	13	32	51	70
	3	17	36	55	74

The split screen function allows the user to partition the display face into as many as three variable height horizontal bands. The bands contain data from anywhere in the addressable mapping memory. Up to three simultaneous views can be presented from three areas of mapping memory which are not necessarily contiguous.

The band defined by (SX_s, SY_s, LN_s) will wrap around to the beginning of mapping memory if the SX_s is closer to the "right edge" of mapping memory than the horizontal resolutions of the screen. If the sum of LN_1, LN_2, and LN_3 is less than the vertical screen resolution, then the split screen 1, 2, 3 will wrap around in that order.

The cursor will appear in each split screen section in which either an X or Y coordinate of the cursor is in that section.

The following depicts the split screen function:

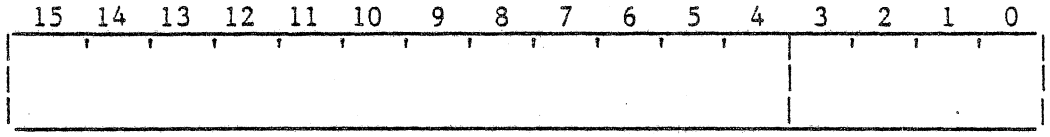


NOTE

The system sets up SX1, SY1, LN1 to display the complete screen when split screen is not being used.

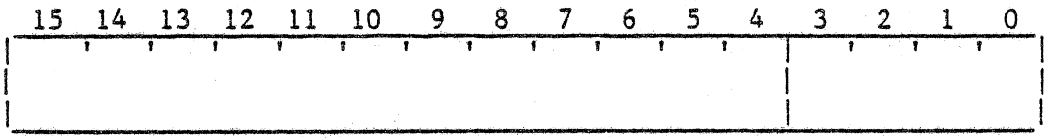
4.3.6 CONFIGURATION REGISTERS. Three types of registers are available to determine the configuration of the present GRAPHIC 8 installation.

|DCF| DISPLAY CONFIGURATION REGISTER Octal address: 165052



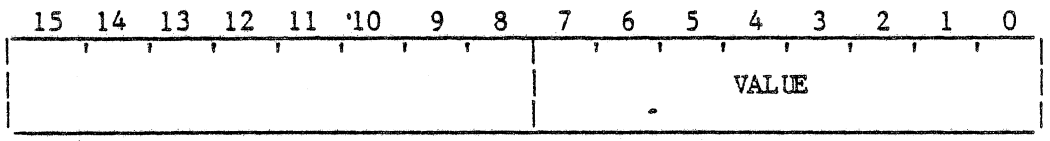
Display monitor number n (1-4) is configured into the system when bit n-1 of the Display Configuration register is set.

|VCF| VIDEO CONTROLLER DIRECTORY REGISTER Octal address: 165046



Video controller number n (1-4) is configured into the system when bit n-1 is set in the Video Controller Directory register.

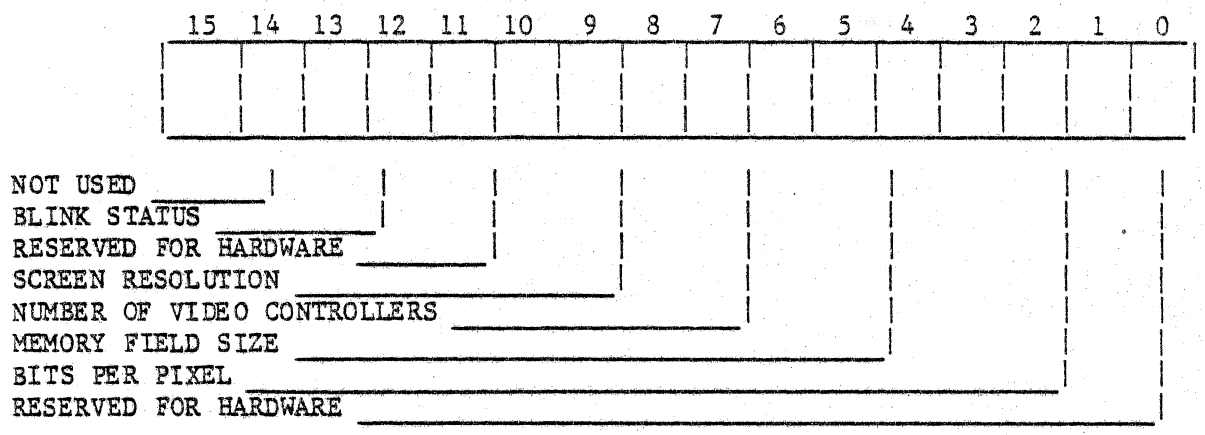
|CFR| CHARACTER FONT REGISTER Octal address: 165070



This register contains a value indicating the character font present in the current configuration.

Value = 377 for 7 x 9 font
= 371 for 5 x 7 font

SCF System Configuration Register Octal address: 165050



BIT(S)	VALUE
1-2	Bits per pixel:

2	1	
0	1	= 2 bits per pixel
1	0	= 4 bits per pixel
1	1	= 8 bits per pixel

3-5 Memory field size:

5	4	3		X	Y
0	0	1	=	2048	x 2048
0	1	0	=	1024	x 2048
0	1	1	=	2048	x 1024
1	0	0	=	1024	x 1024
1	0	1	=	512	x 1024
1	1	0	=	1024	x 512
1	1	1	=	512	x 512

6-7 Number of Video Controller

7	6	
0	0	= 1
0	1	= 2
1	0	= 3
1	1	= 4

8-9 Screen Resolution:

9	8	
0	0	= 512 x 512
0	1	= 640 x 480
1	0	= 1024 x 768
1	1	= 1024 x 1024

Blink/No Blink:

12	
0	= MSB or PDR is blink bit
1	= MSD or PDR is part of pixel data.

4.4 INTERFACE REGISTERS

Interface registers are associated with the various serial and parallel interface ports of the GRAPHIC 8. 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 thirteen serial interface ports are available for external devices to communicate with the GRAPHIC 8. One is located on the ROM and status logic card and four are located on each multiport serial interface card (three multiport serial interface cards may be installed in the terminal controller). The designations of these ports and the associated devices for 4 keyboards and 4 PEDs are as follows (note that ports 1, 5 and 9 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	Alphanumeric/Function Keyboard no. 3	
3	Alphanumeric/Function Keyboard no. 1	
4	PED no. 1	
5 (RS-232C)	Unused	Multiport serial interface card no. 2
6	PED no. 3	
7	Alphanumeric/Function Keyboard no. 2	
8	PED no. 2	
9 (RS-232C)	Alphanumeric/Function Keyboard no. 4	Multiport serial interface card no. 3
10	PED no. 4	
11	Unused	
12	Unused	
TTY	Teletypewriter	ROM and status logic card

Bits 0-3 of the Display Processor's 8-bit switch register (octal location 177774) represent the port assignments for keyboards and PEDS. When LOCAL or system mode is entered, GCP connects the corresponding device interrupt routines to the proper ports.

The following describes the assignment of serial interface ports 2, 3, 4, 6, 7, 8, 9 and 10 for different GRAPHIC 8 configurations. Port 1 is reserved for the host. Port 5 is unused.

Value in bits 0-3 of the switch register represent the number of keyboards in the system.	Ports left for PED no.							
	8	7	6	5	4	3	2	1
0	3	7	2	9	10	6	8	4
1	3	7	2	9	10	6	8	4
2	3	7	2	9	10	6	8	4
3	3	7	2	9	10	6	8	4
4	3	7	2	9	10	6	8	4
5	3	7	2	9	10	6	8	4
6	3	7	2	9	10	6	8	4
7	3	7	2	9	10	6	8	4
8	3	7	2	9	10	6	8	4
	1	2	3	4	5	6	7	8
	Ports for Keyboard No.							

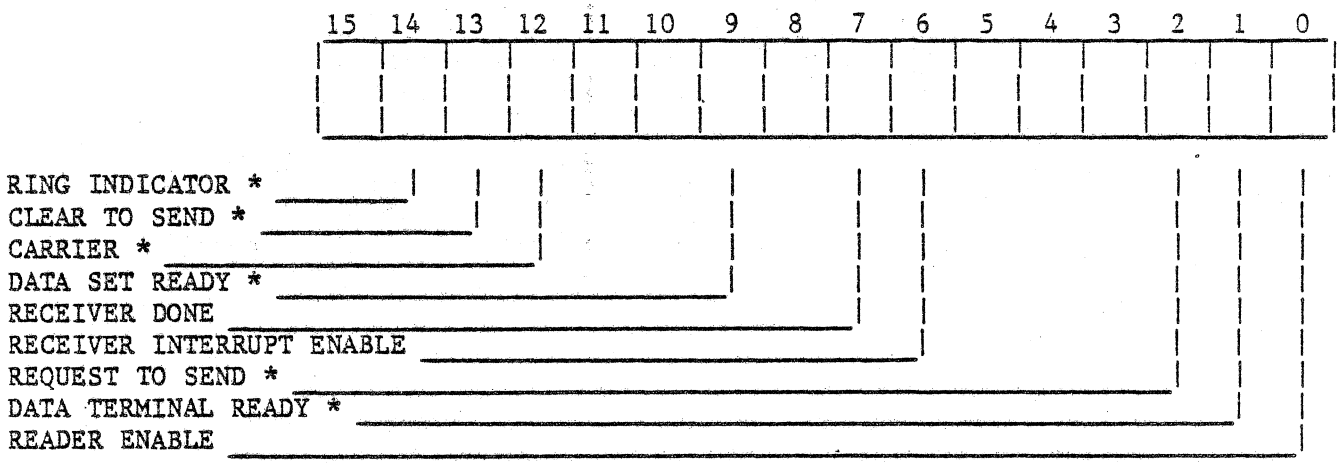
Normally, ports 1, 2, 3, and 4 are on multiport serial interface card no. 1. However, each port (1-8) can be "dialed" so to physically exist on any serial interface card (1 or 2). For example, two keyboards (ports 3 and 7) and two PEDS (ports 4 and 8) with no serial interface to the host computer could all be converted to multiport serial interface card no. 1.

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)
176600 (RSR9)
176610 (RSR10)
176560 (TTYRSR)



NOTES

1. Unidentified bits are not used.
2. Bits marked with an asterisk (*) are used on full RS-232C interface ports (ports 1, 5, and 9) 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

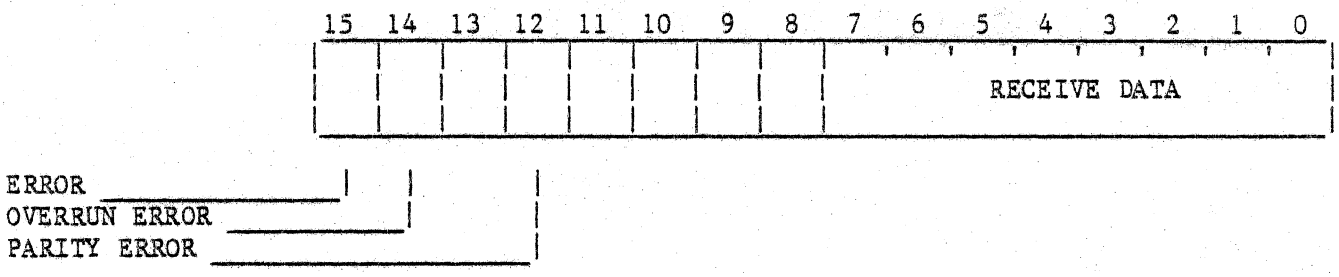
Bit	Function	Remarks
7	Data ready	1. Program read only 2. Cleared by controller bus reset 3. Set when receiver has transferred a character into associated receive data buffer (RDBn) 4. Cleared by setting reader enable (bit 0) or by reading RDBn 5. If receiver interrupt enable (bit 6) is set, setting this bit causes display processor interrupt to be generated 6. Interrupt trap addresses (octal) for each register are: RSR1 - 000300 RSR6 - 000350 RSR2 - 000310 RSR7 - 000360 RSR3 - 000320 RSR8 - 000370 RSR4 - 000330 RSR9 - 000400 RSR5 - 000340 RSR10 - 000410 TTYRSR - 00060

The following receive status register bits are used on full RS-232C interface ports (ports 1, 5, and 9) 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

Bit	Function	Remarks
12	Carrier	1. Program read only 2. Status of this bit reflects level at pin 16 of 26-pin I/O connector
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 8 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)
176602 (RDB9)
176612 (RDB10)
177562 (TTYRDB)



NOTES

1. Unidentified bits are not used.
2. Parity error (bit 12) is used on full RS-232C interface ports (ports 1, 5 and 9) only.

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

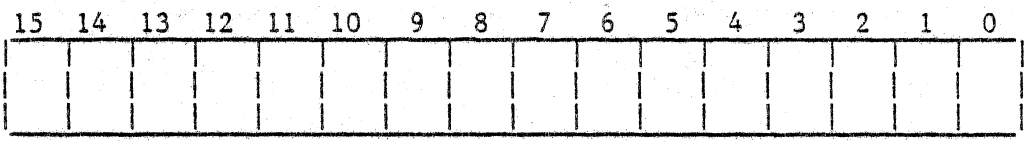
Bit(s)	Function	Remarks
0 thru 7	Receive data	1. Program read only 2. These bits contain last serial character received 3. If character is less than 8 characters, unused high-order bits will be zeros
14	Overrun error	1. Program read only 2. Cleared by controller bus reset 3. Updated each time a character 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, 5, and 9) 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) 176604 (TSR9) 176614 (TSR10) 177564 (TTYTSR)



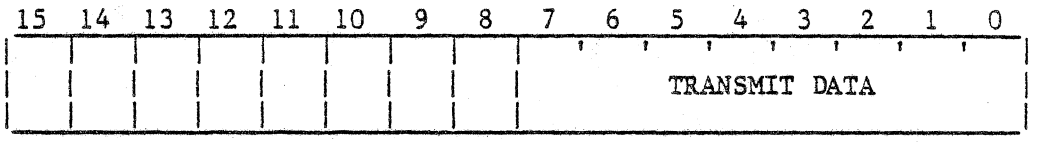
NOTE

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, TSR2 - 000314, TSR3 - 000324, TSR4 - 000334, TSR5 - 000344, TSR6 - 000354, TSR7 - 000364, TSR8 - 000374, TSR9 - 000404, TSR10 - 000414, 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 the 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) 176606 (TDB9) 176616 (TDB10) 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.

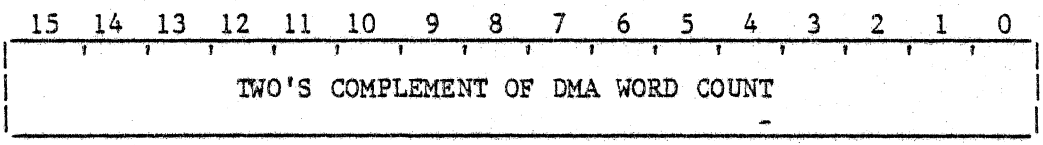
4.4.2 PARALLEL INTERFACE REGISTER. Up to two parallel interfaces can be used by external devices to communicate with the GRAPHIC 8; a separate card is required for each. Four registers are associated with each parallel interface. These are a word count register, a memory address register, a status register, 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 two parallel interface cards may be installed.

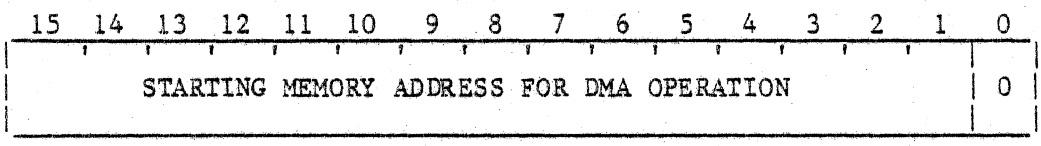
With respect to parallel interface registers, input data is data sent from the GRAPHIC 8 to the host computer; output data is data sent from the host computer to the GRAPHIC 8.

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



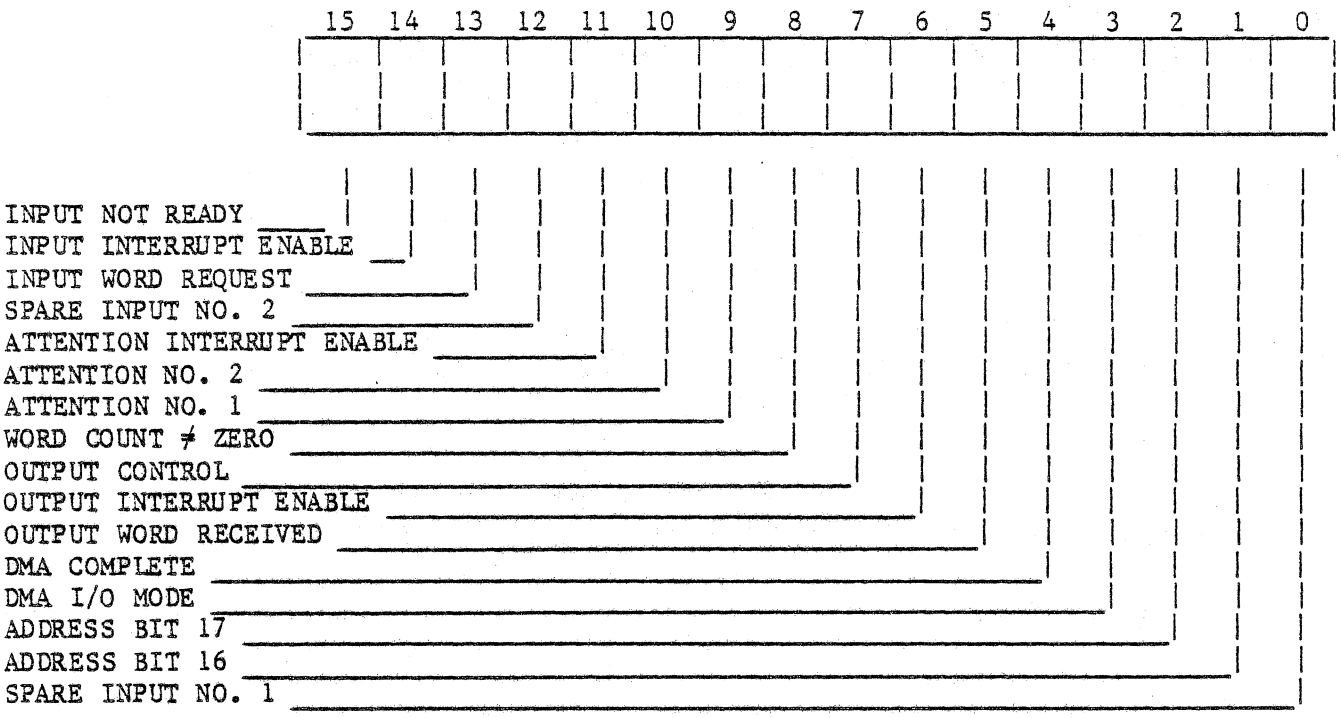
The word count register is a program read/write register used for direct memory access (DMA) operations. It is cleared by a controller bus reset. To initiate a DMA operation, the program writes into the word count register the two's complement of the number of memory words to be transferred between the GRAPHIC 8 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)



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 8 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)



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

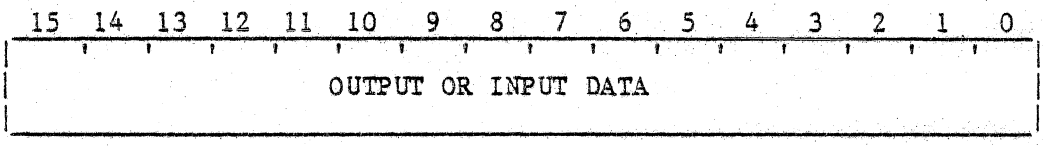
Bit	Function	Remarks
2	Address bit 17	1. Program read/write 2. Cleared by controller bus reset 3. This bit and address bit 16 (bit 1) are used in conjunction with the address in the memory address register (MARn) to expand the DMA addressing capability to 128K words
3	DMA I/O mode	1. Program read/write 2. Cleared by controller bus reset 3. When set, indicates DMA input operation (transfer of words from GRAPHIC 8 to host computer). When cleared, indicates DMA output operating (transfer of words from host computer to GRAPHIC 8) 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 STR2 - Unassigned
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 STR2 - Unassigned
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 - Unassigned
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)
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 8 and as an input data register (IDRn) when data is being transferred from the GRAPHIC 8 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 8) the interface loads the ODRn contents into the GRAPHIC 8 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 8 to the host computer) the interface loads the IDRn with data from the GRAPHIC 8 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 Graphic Control Program (GCP), a program in read only memory (ROM), is the central intelligence of the GRAPHIC 8. GCP allows the user to easily control the interactions between the human and Graphic system responses. This program handles all the tasks for the GRAPHIC 8 that must normally be programmed for other display systems. The software engineer, 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:

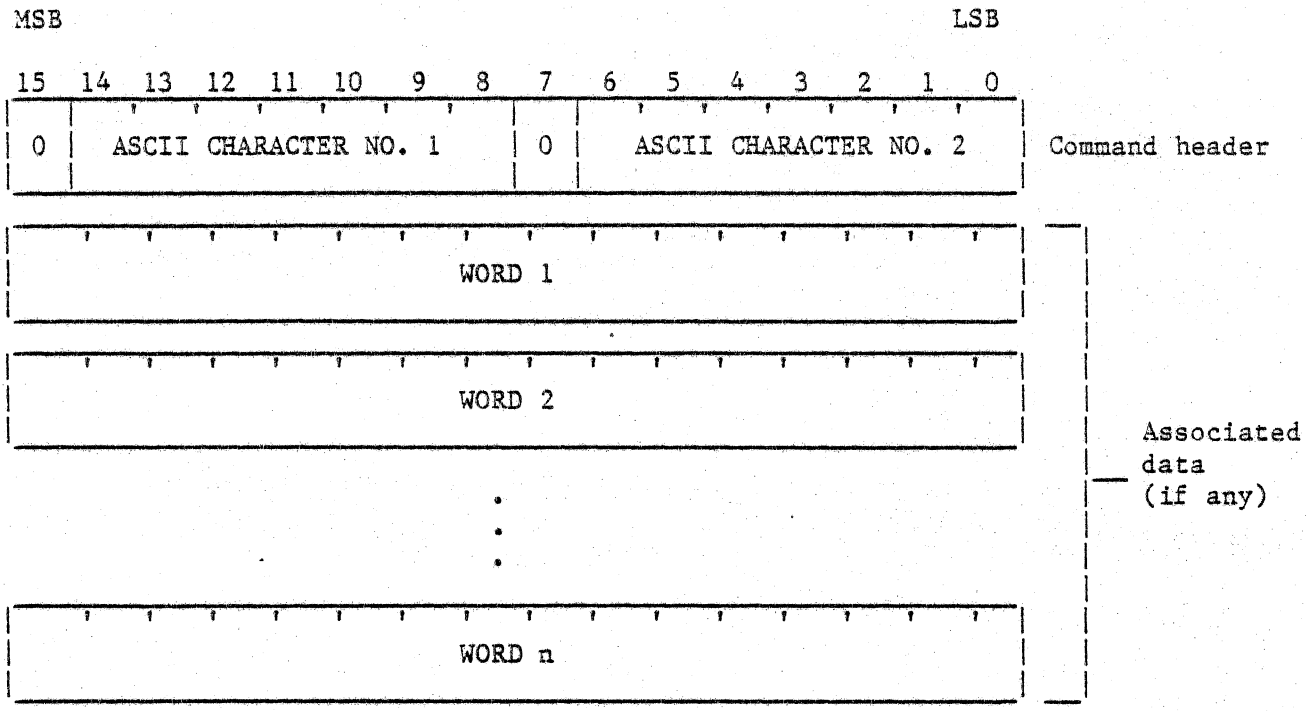
- Routine housekeeping
- Handling of all operator inputs
- Handling of trackball/forcestick or data tablet manipulations
- Handling of all graphic controller interrupts
- Insertion of keyboard data directly into a refresh file
- Formatting of messages for GRAPHIC 8-to-host communications

When the GRAPHIC 8 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 8 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 8 will be processed and the type of data that will be returned in messages to the GRAPHIC 8. For controlling these operations, the application programmer has full access to all control registers of the terminal controller.

Generations 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 8. 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 suport packages and data statements. A package of this type available as an option for the GRAPHIC 8 is the host-based FORTRAN support package (FSP).

5.2 HOST/GRAPHIC 8 COMMUNICATIONS

All communications between the host computer and the GRAPHIC 8 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 8-to-host messages, the translation is accomplished by GCP. For host-to-GRAPHIC 8 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 8 message GI 013700_8 000746_8. This message instructs the GRAPHIC 8 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:

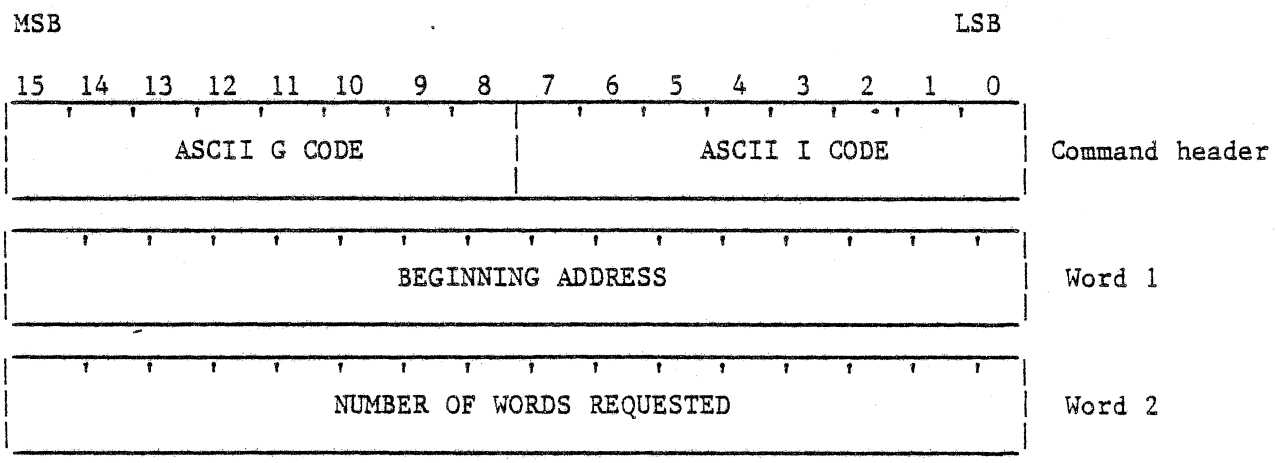


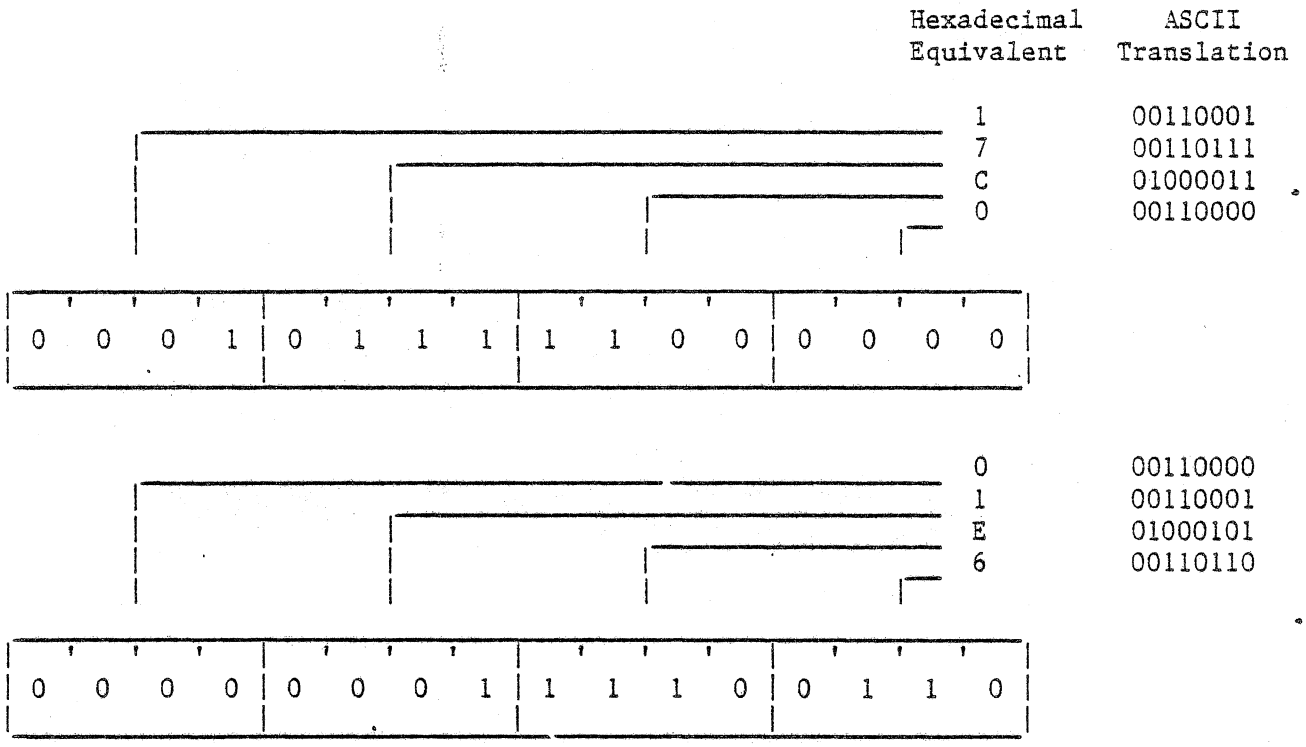
Table 5-1. Data Word Translation Codes

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:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	1	0	0	0	1	1	1	0	1	0	0	1	0	0	1	GI
0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	013700_8
0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	0	000746_8

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	
	ASCII 0 CODE	0 0 1 1 0 0 0 0
Ending address data word	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 8 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 8 MESSAGES

There are nine different groups of messages transmitted between the host computer and the GRAPHIC 8. 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. FORTRAN support (FSP) messages
8. Option messages
9. 3D coordinate converter messages

5.3.1 INITIALIZE AND ERROR MESSAGES. The initialize and error group consist of the following messages:

HOST-to-GRAPHIC 8 (H->G8)

IZ Initialize

GRAPHIC 8-to-HOST (G8->H)

XX

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

|IZ| (H→G8) INITIALIZE Command header code (octal): 044532

15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0	
0	ASCII I CODE	0	ASCII Z CODE	Command header

The initialize message is a single-word message that causes the GRAPHIC 8 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.

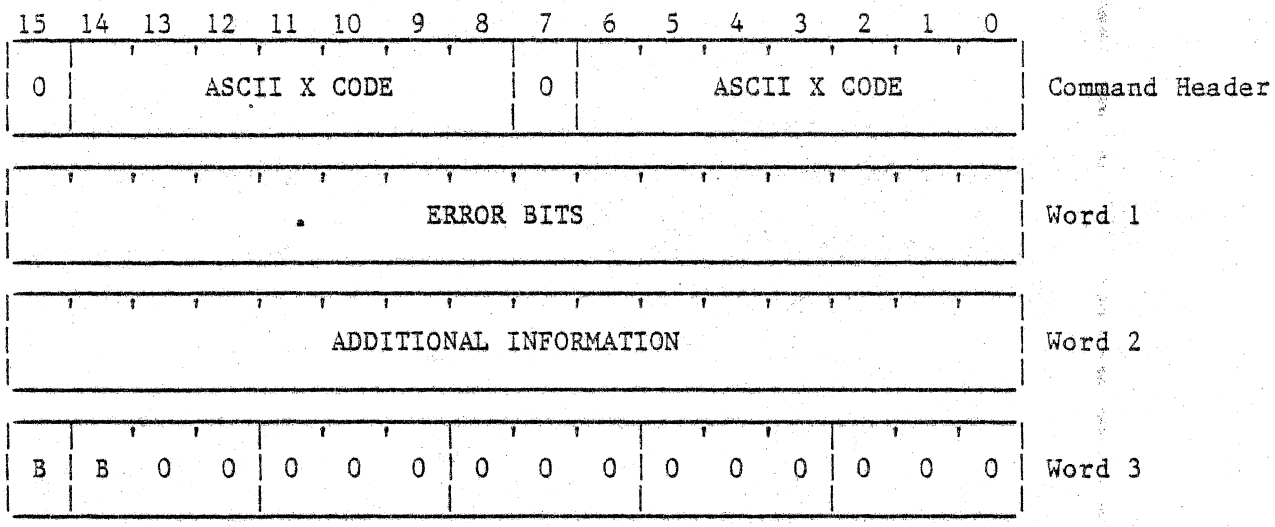
NOTES

1. An IZ message is recognized by the GRAPHIC 8 only when the GRAPHIC 8 is operating in the system mode. If the GRAPHIC 8 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 8 front panel.

2. After an IZ message has been sent, no further message should be sent from the host computer to the GRAPHIC 8 until an XX message has been sent from the GRAPHIC 8 to the host computer.

3. When the GRAPHIC 8 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 8.

|XX| (G8—>H) ERROR STATUS Command header code (octal): 054130



Whenever the GRAPHIC 8 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 8 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:

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

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:

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

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

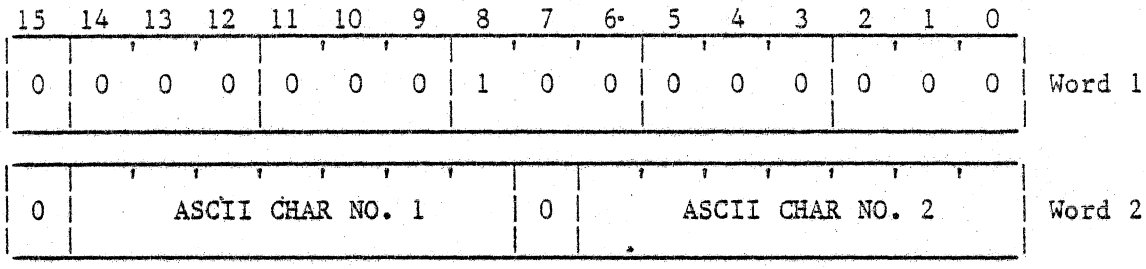
Bit 3 - incorrect message format sent by host computer
Bit 7 - unidentified internal interrupt detected by display processor
Bit 10 - GCP serial interface buffer is full
Bit 11 - GCP serial interface buffer is 7/8 full
Bit 14 - Command header not recognized by GCP

All other bits are always zero.

Any combination of errors can be indicated.

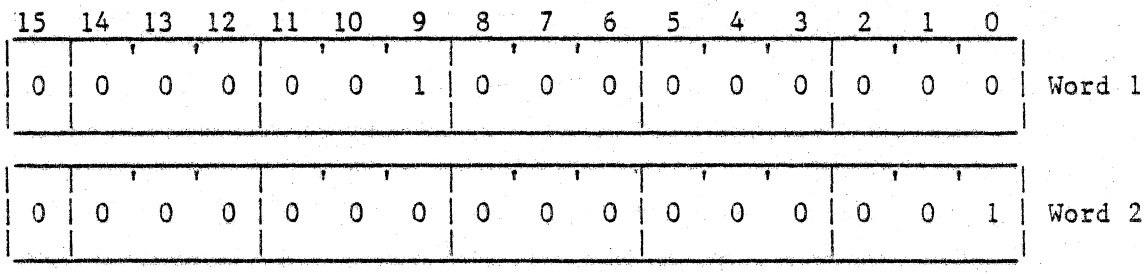
Word 2 contains all zeros for bit 3, 10, 11 and 14 type errors. For bit 7 errors, word 2 contains the address plus 4 bytes to identify the address of the unidentified internal interrupt.

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 8

IM	Initialize I/O message formats
PL	Poll GRAPHIC 8 for next message
NO	No operation

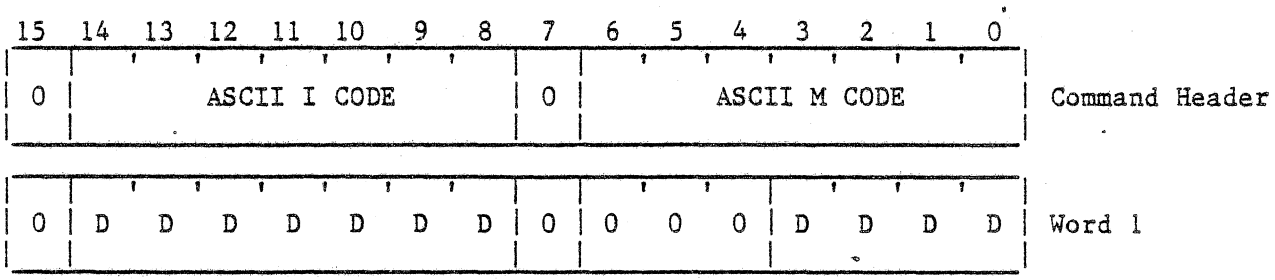
GRAPHIC 8-to-HOST

NM	No messages ready

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

IM (H->G8) INITIALIZE I/O MESSAGE FORMATS

Command header code (octal): 044515



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 8.)
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 8 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 8 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.

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 8 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 0.

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 0 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 0 has been set up. When bit 0 is set up for non-polling mode (bit 0 = 0), the value of bit 1 is ignored. When bit 0 is set up for polling mode (bit 0 = 1), GCP operates as follows:

1. When bit 1 is 0, 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 8 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 0 is 0, the special poll character is ignored by GCP. Setting bit 0 to a 1 activates the special poll character bits. If the special poll character is set up for 0 (i.e., bits 8 through 14 are 0), 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.

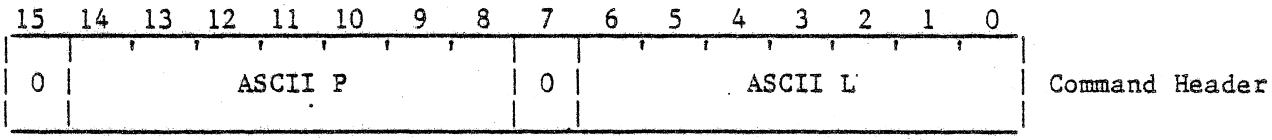
Bit 3 of word 1 of the IM message is used to activate the detection of command header errors. By default, GCP is initialized to ignore all command header errors. By setting bit 3 to a 1, GCP can be activated to validate all command headers. When GCP detects that the host computer has sent an invalid 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->G8) POLL GRAPHIC 8 FOR NEXT MESSAGE

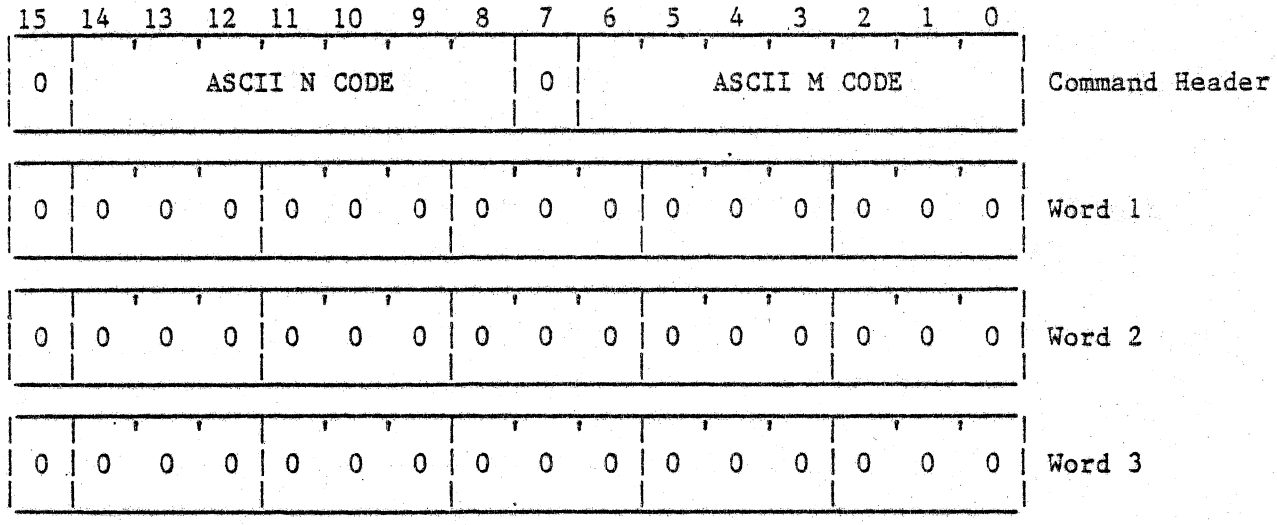
Command header code (octal): 050114



The poll message is sent by the Host to request that the GRAPHIC 8 send the next message. This command works in conjunction with the way the IM command has initialized GCP.

|NM| (G8→H) NO MESSAGES READY

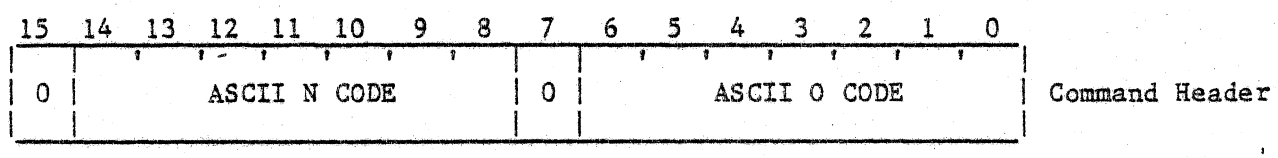
Command header code (octal): 047115



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→G8) 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 8 be of constant length.

5.3.3 MEMORY RELATED MESSAGES. The memory related messages consist of the following:

Host-to-GRAPHIC 8

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

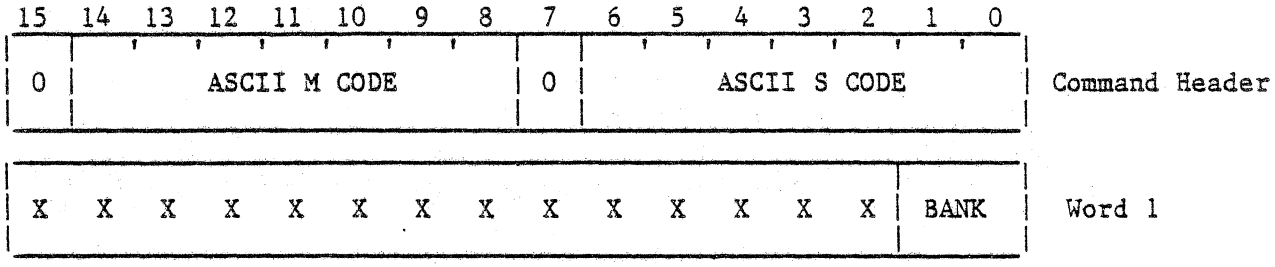
GRAPHIC 8-to-Host

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->G8) 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

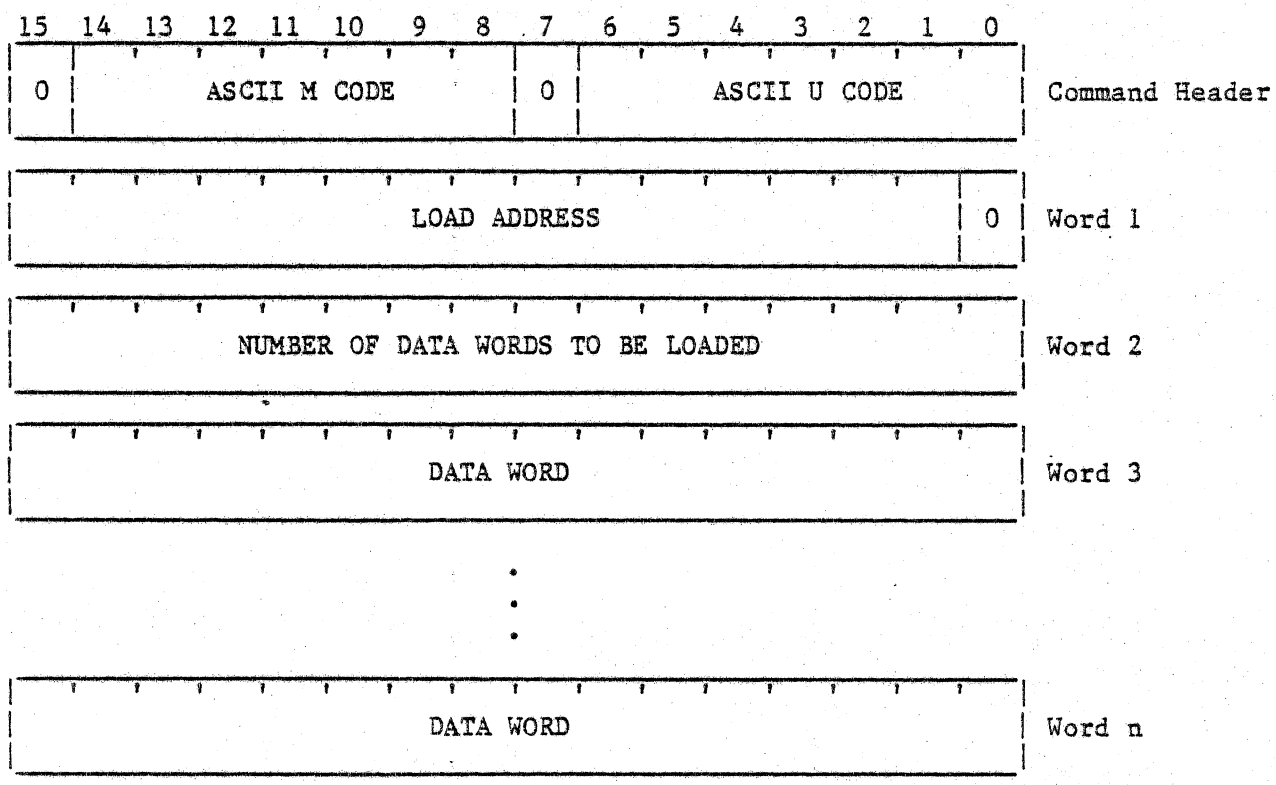
When GCP is initialized by default, bank 0 is selected. Memory bank selection should be taken into account for the following Host-to-GRAPHIC 8 messages:

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

Memory bank selection should be taken into account for the following GRAPHIC 8-to-Host messages:

RI, XX, HI, XI

|MU| (H→G8) MEMORY UPDATE Command header code (octal): 046525

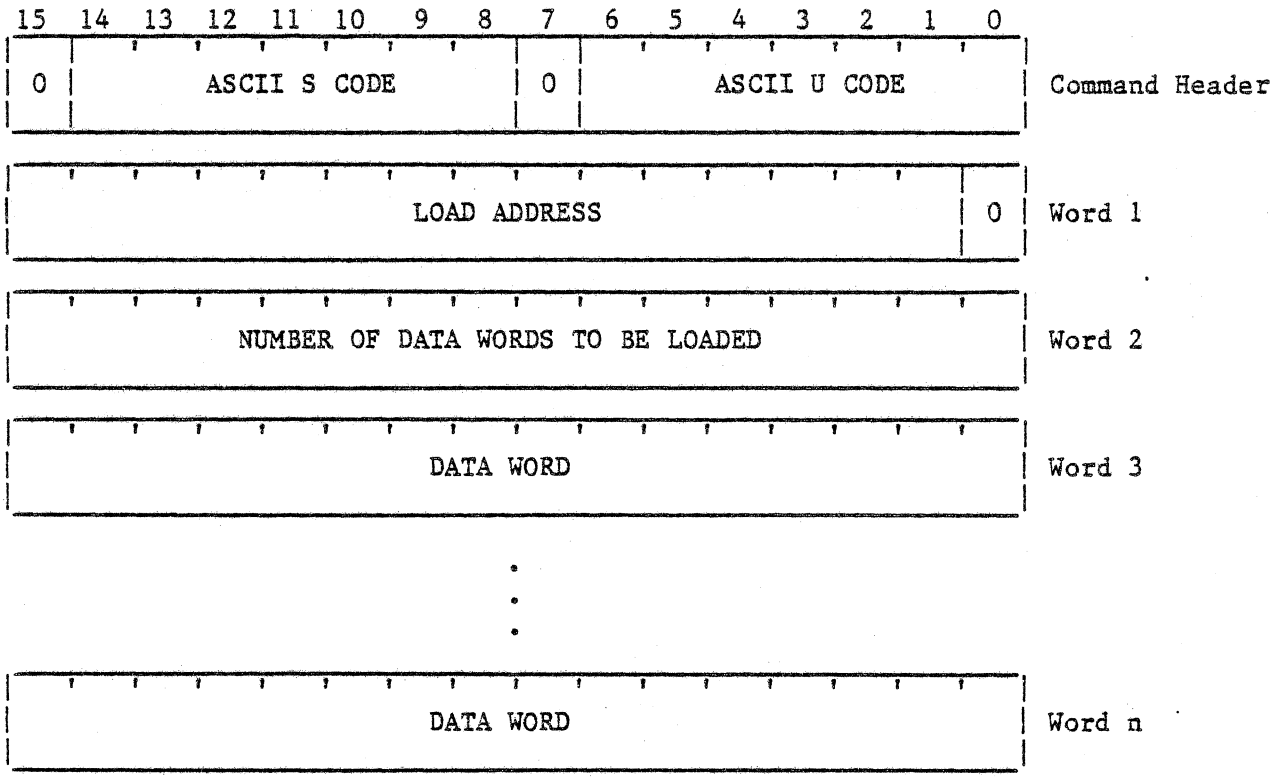


The MU message is a variable-length message used to load data into the read/write memory of the GRAPHIC 8. The load address specified in word 1 tells GCP the address at which loading of the data should begin. This address must be the address of an even-numbered byte. An odd address will result in an XX (error status) message being returned by the GRAPHIC 8 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 8, GCP halts the graphic controller (this blanks the display indicator). The GRAPHIC 8 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.

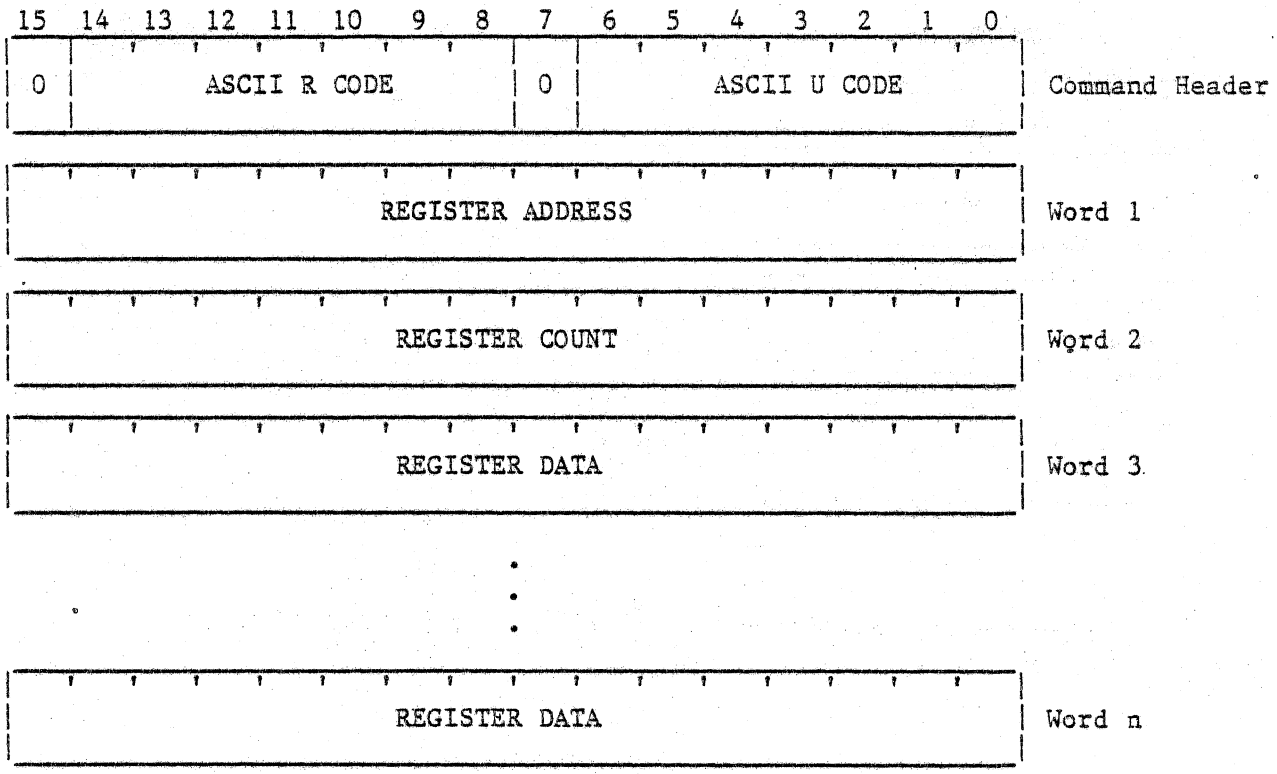
|SU| (H→G8) SELECTIVE UPDATE Command header code (octal): 051525



The SU message is a variable-length message that operates in exactly the same manner as the MU (memory update) message except that the digital 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→G8) REGISTER UPDATE Command header code (octal): 051125

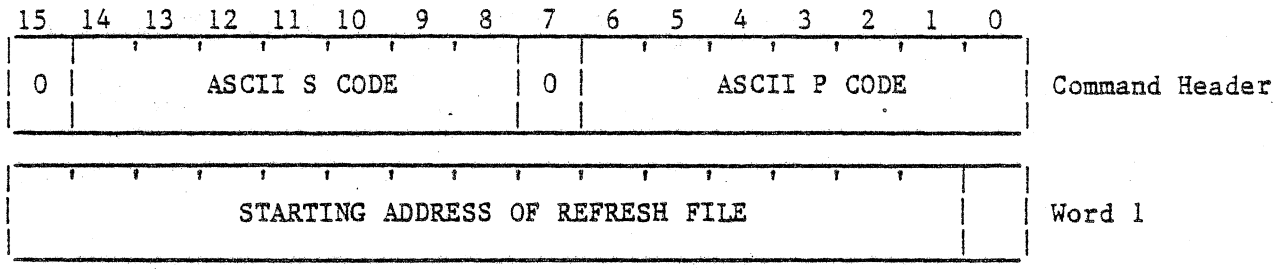


The RU message is a variable length message that is used to update a series of registers in the I/O address area of the hardware. Word 1 contains the address of the first register to be updated. Valid register addresses are in the range of 160000-177777 (octal). Word 2 contains the register count indicating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

NOTE

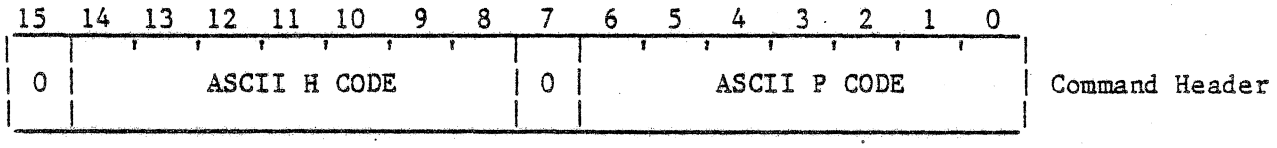
The RU message does not change the current memory bank selection. It is also possible to interpret register address as memory address in the above message. When updating memory address, the user must take into account memory mapping. Memory addresses in the range of 020000 to 077777 are subject to memory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

|SP| (H→G8) START PICTURE Command header code (octal): 051520



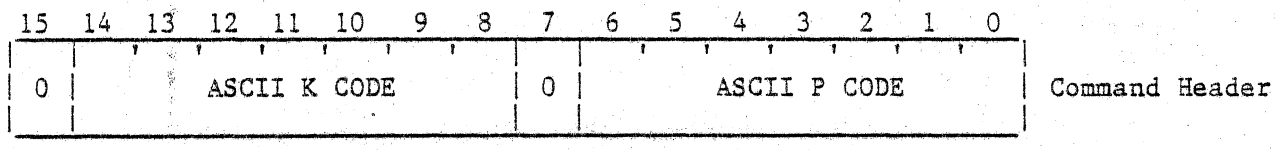
The SP message is a two-word message that causes the digital graphic controller to begin processing a refresh file starting at the address specified in word 1. The specified starting address should be the address of an even-numbered byte. If, however, an odd address is specified, the low order bit is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

|HP| (H→G8) HALT PICTURE Command header code (octal): 044120



The HP message is a single-word message that causes GCP to halt the digital graphic controller. 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 digital graphic controller remains halted until an SP (start picture) or KP (continue picture) message is sent by the host computer.

|KP| (H->G8) CONTINUE PICTURE Command header code (octal): 045520



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

a. Host computer sends HP (halt picture) message to GRAPHIC 8

b. Host computer sends MU (memory update) message to GRAPHIC 8

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

d. Display processor executes RESET instruction

e. HREF instruction executed by digital graphic controller

f. LINK instruction executed by digital graphic controller

g. Invalid instruction executed by digital graphic controller

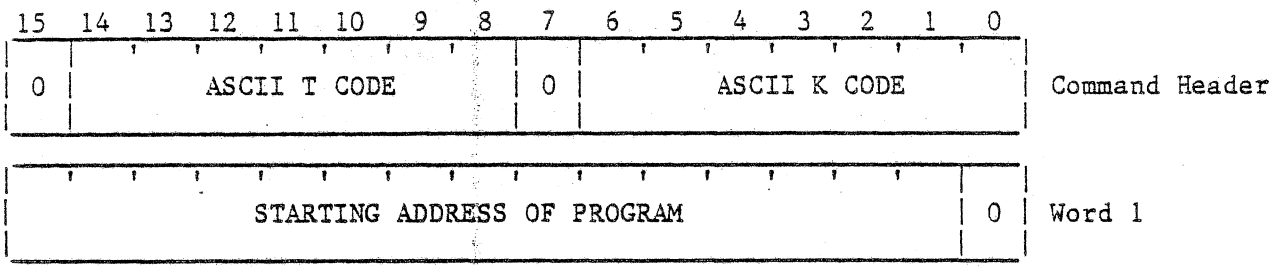
h. Bus timeout (memory fails to respond to a fetch command)

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

NOTE

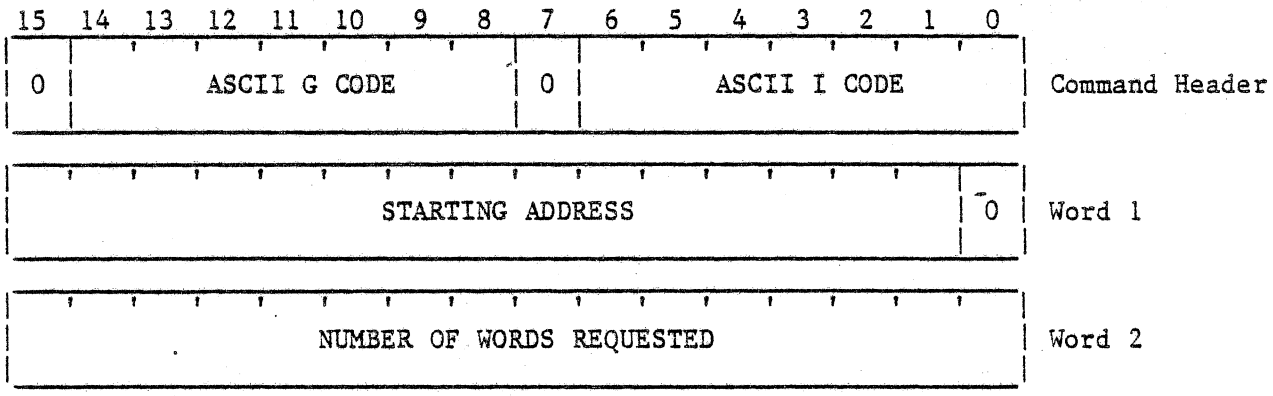
If the digital graphic controller is running when a KP message is sent by the host computer, GCP returns an XX (error status) message to the host computer.

|TK| (H→G8) 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 returns to GCP with an RTS PC instruction.

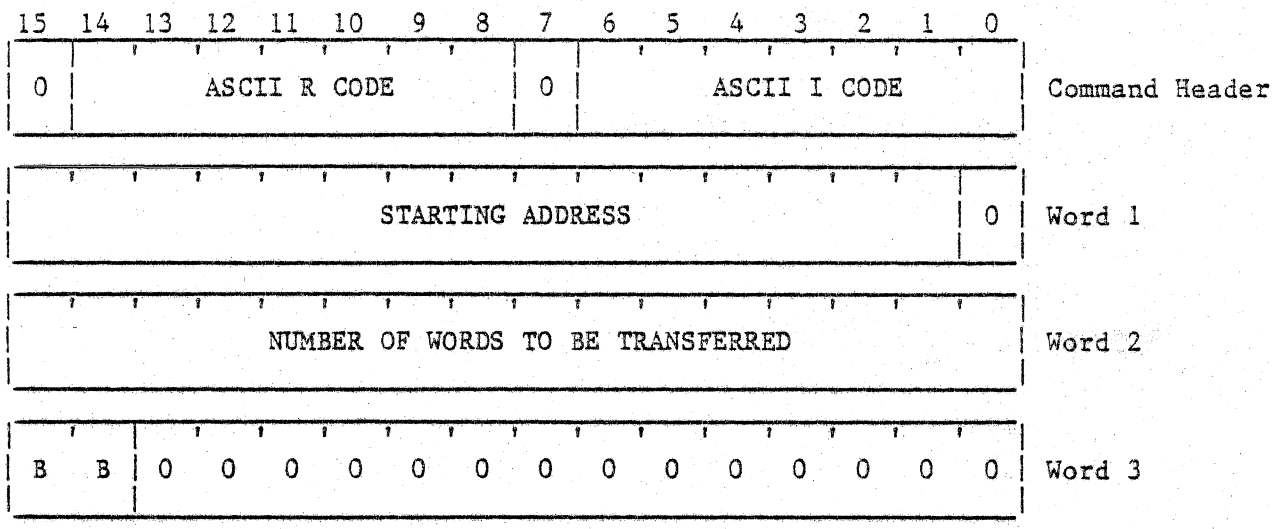
|GI| (H→G8) GIVE IMAGE Command header code (octal): 043511



The GI message is a three-word message that causes GCP to send back to the host computer the contents of the GRAPHIC 8 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.

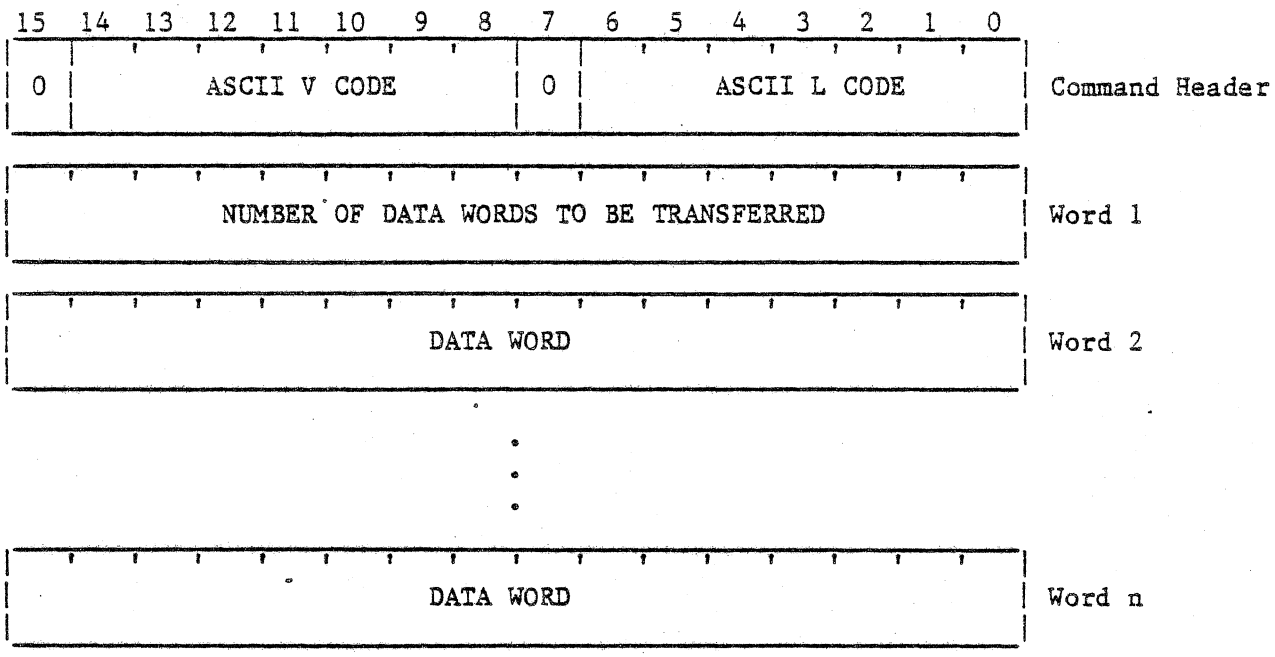
In response to a GI message, GCP sends an RI (return image) and a VL (variable length) message to the host computer. The RI message indicates the length of the VL message while the VL message contains the requested data.

|RI| (G8→H) RETURN IMAGE 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| (G8->H) VARIABLE LENGTH Command header code (octal): 053114

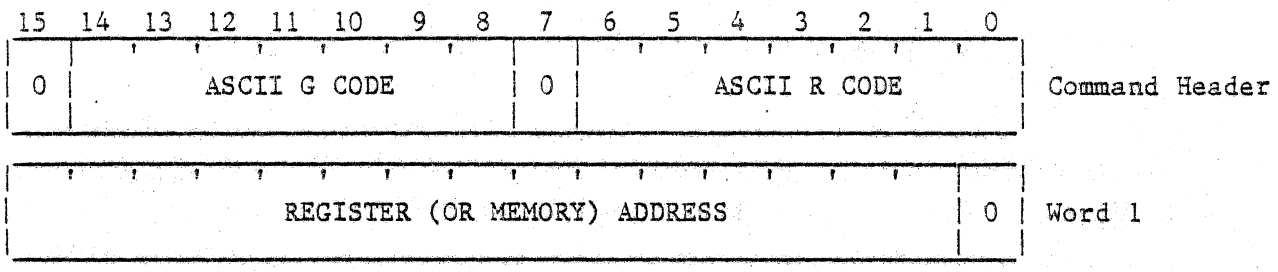


The VL message is the only GRAPHIC 8-to-host message that does not necessarily contain four words. Its length is determined by the number of data words to be transferred. In response to a GI (give image) message from the host computer, GCP returns an RI (return image) message that is immediately followed by a VL message. The RI message informs the host computer that a VL message is to follow. 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→G8) 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 8 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 8 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.

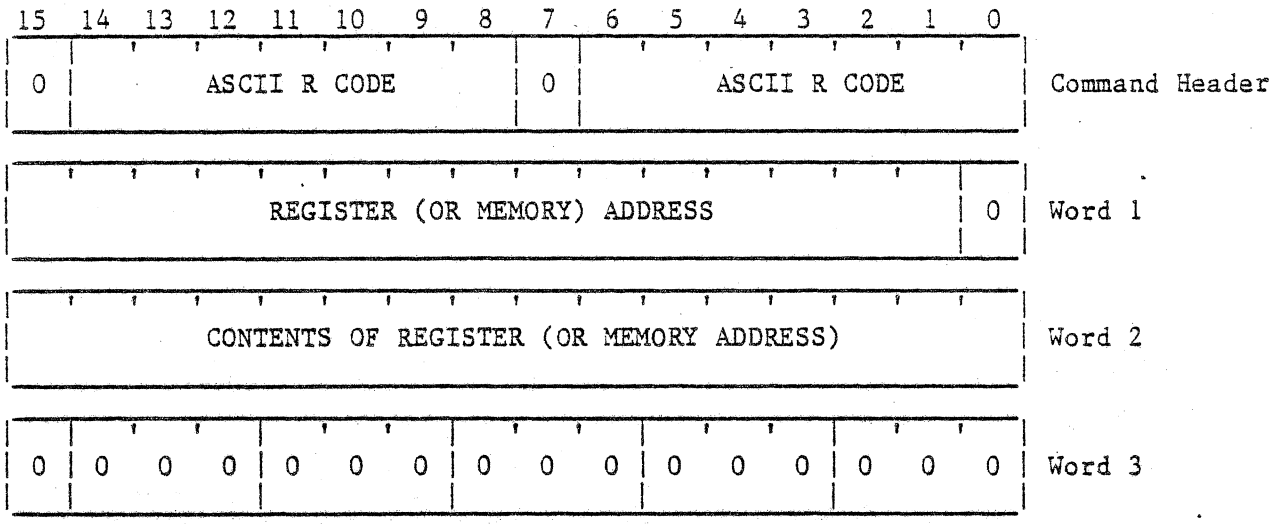
1. Addresses in the range of 000000-017776 are directly addressable.

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 (G8→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 8

IK Interrupt control

IS Enable selected interrupts

ZI Disable selected interrupts

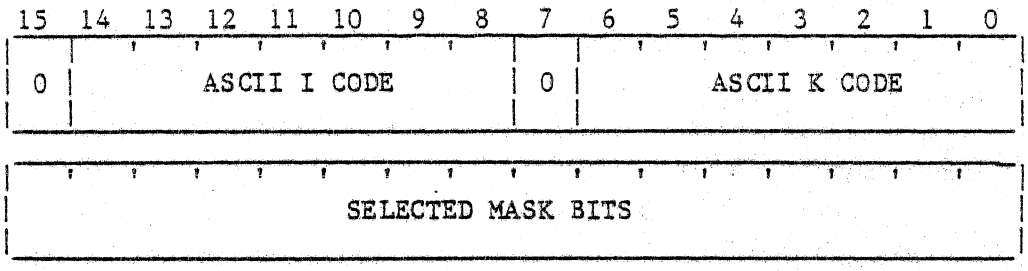
GRAPHIC 8-to-Host

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→G8) INTERRUPT CONTROL Command header code (octal): 044513



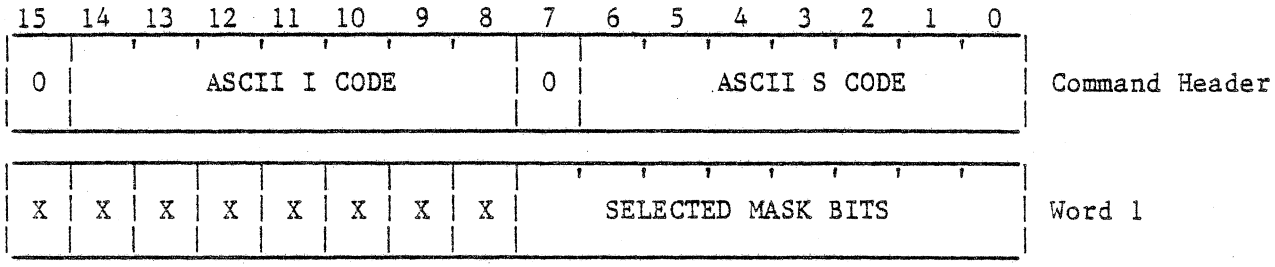
The IK message is a two-word message used to enable or disable certain GCP functions and to determine conditions under which the digital graphic controller can interrupt the display processor. When an IK message is sent, the contents of bits 1-3 of the word 1 directly replaces the contents of the graphic controller mask register (MKR). The remaining bits of the word are decoded and used to enable or disable associated interface ports. The function or interrupt condition associated with each bit is as follows:

BIT	FUNCTION OR INTERRUPT CONDITION	SERIAL INTERFACE PORT
0	PED no. 4	10
1	Digital Graphic Controller halt	N/A
2	X or Y position overflow	N/A
3	Real time clock	N/A
4	Unused	
5	Unused	
6	Unused	
7	Unused	
8	Unused	
9	Alphanumeric/function keyboard no. 4	9
10	PED no. 2	8
11	Alphanumeric/function keyboard no. 2	7
12	PED no. 3	6
13	PED no. 1	4
14	Alphanumeric/function keyboard no. 3	2
15	Alphanumeric/function keyboard no. 1	3

Note the serial interface ports to device assignments may vary for different configurations. See section 4.4.1 for more information on port assignments.

|IS| (H->G8) ENABLE SELECTED INTERRUPTS

Command header code (octal): 044523



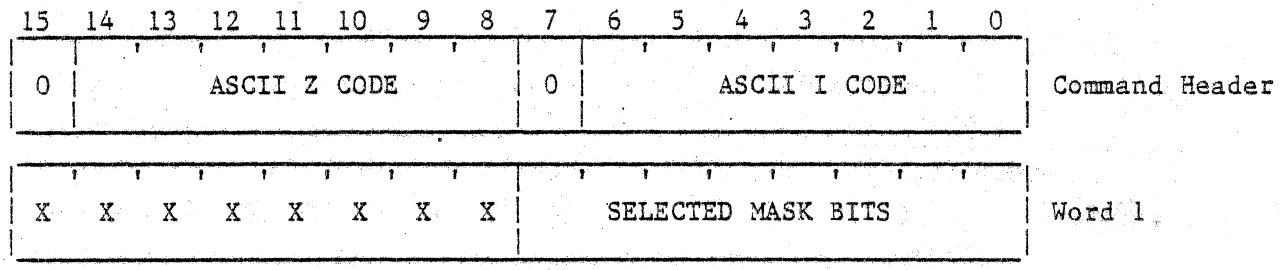
The IS message is a two word message used to selectively enable mask register (MKR) associated interrupts. If a mask bit is set to 1, then the interrupt associated with that bit is enabled. If a mask bit is set to 0, then the interrupt associated with that bit remains unchanged after the IS message is processed by GCP.

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

BIT	FUNCTION OR INTERRUPT CONDITION
0	Not used
1	Digital graphic controller halt
2	X or Y position overflow
3	Real time clock
4-15	These bits are ignored by GCP

|ZI| (H->G8) 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 0, then the interrupt associated with that bit remains unchanged after the ZI message is processed by GCP.

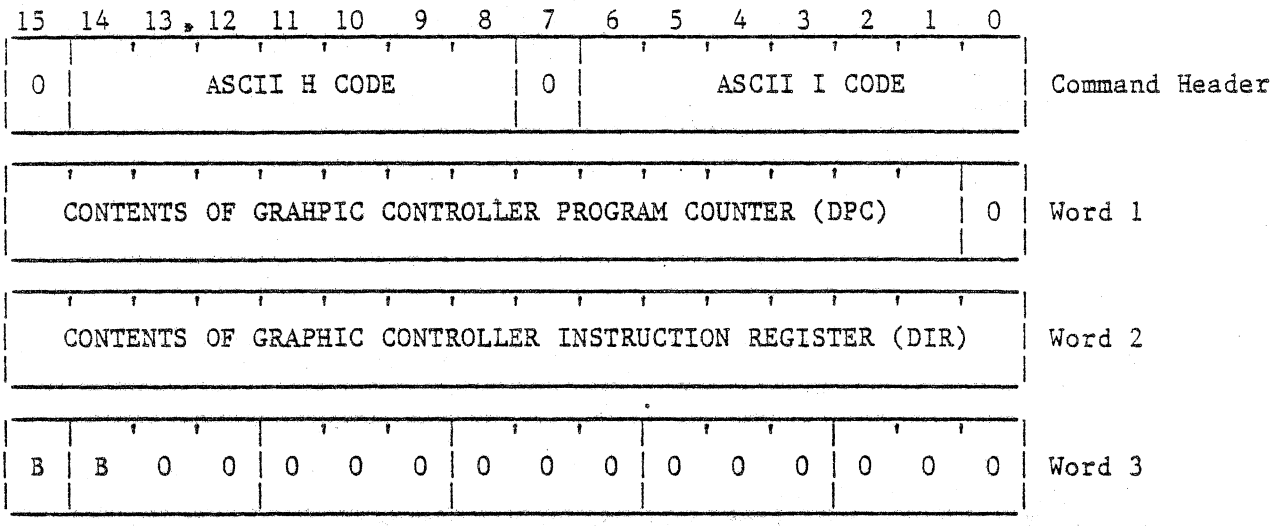
The function or interrupt associated with each bit is as follows:

BIT	FUNCTION OR INTERRUPT CONDITION
0	Not used
1	Digital graphic controller halt
2	X or Y position overflow
3	Real time clock
4-15	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.

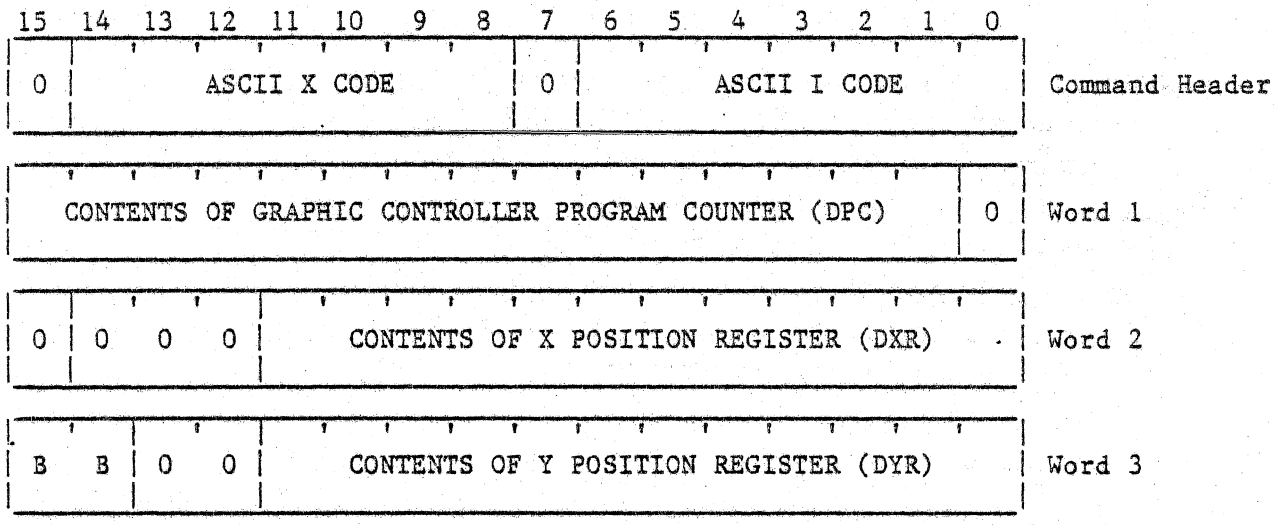
|HI| (G8→H) HALT INTERRUPT Command header code (octal): 044111



When the digital graphic controller halt interrupt to the display processor is enabled (by a host-to-GRAPHIC 8 IK or IS message), GCP sends an HI message to the host computer each time that a HREF instruction is executed by the digital 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| (G8→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 8 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 10 and 11 of the X position register or the Y 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 second instruction following the instruction that caused the interrupt. 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 bits 14 and 15 or word 3 contains 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 8

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 8-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

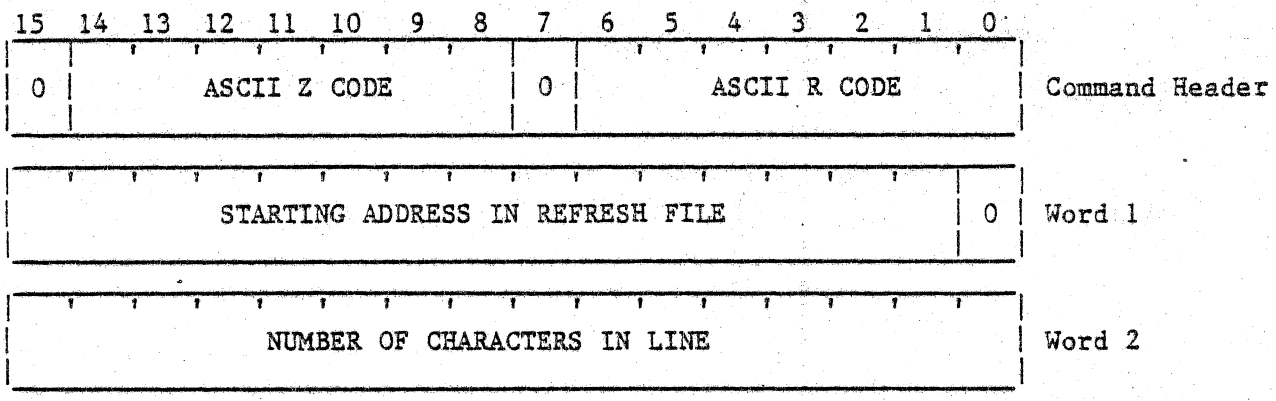
RL Function keyboard no. 2

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

(H→G8) INITIALIZE SCRATCHPAD FOR ALPHANUMERIC KEYBOARD

|ZR| no. 1 Command header code (octal): 055122

|ZT| no. 2 055124



This three word Initialize Scratchpad message is used to establish parameters for handling alphanumeric characters on a line basis from the keyboard. It is used in conjunction with a refresh file that contains an area reserved for storage of characters typed in by the operator. This area, called a scratchpad, typically consists of a sequence of TXT (draw two tabular characters) instructions with ASCII codes for spaces.

When an Initialize Scratchpad message is sent by the host computer, GCP begins collecting characters from the alphanumeric keyboard, and stores them in the refresh file starting at the address specified in word 1. This address must be even. 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.

Characters are collected in the scratchpad until the total count specified in word 2 is reached. At that point, GCP sends a Scratchpad for Alphanumeric Keyboard Ready message to the host computer. RETURN, which may be typed at any time, terminates collection of characters and causes GCP to send a Scratchpad for Alphanumeric Keyboard 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 8. Note that typing RETURN only causes the Ready 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 Zero Scratchpad, a SU (selective update), or MU (memory update) message from the host computer; b) they are replaced when another Initialize Scratchpad 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.

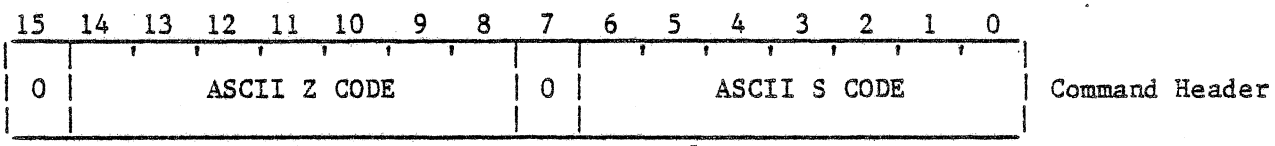
NOTE

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 an Initialize Scratchpad message to the GRAPHIC 8 in which words 1 and 2 (address and character count) are all zeros. GCP then sends a KY (alphanumeric keyboard no. 1) message to the host computer each time a character is typed.

(H→G8) ZERO SCRATCHPAD

|ZS| no. 1 Command header code (octal): 055123

|ZU| no. 2 055125



The Zero Scratchpad message is normally sent after an Initialize Scratchpad message has been processed. The Zero Scratchpad message causes GCP to replace all characters in the scratchpad with spaces. After the scratchpad is set to spaces, the scratchpad input pointer is positioned to the beginning of the scratchpad area.

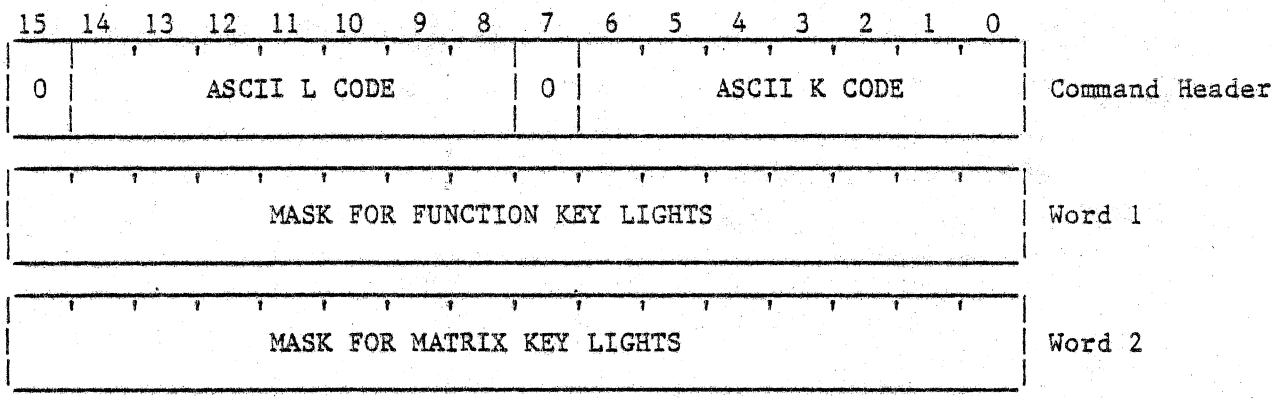
NOTE

The Initialize Scratchpad message is used first to define the starting address and size of the scratchpad in the refresh memory.

(H->G8) LIGHT KEYS ON FUNCTION KEYBOARD

|LK| no. 1 Command header code (octal): 046113

|LT| no. 2 046124



The three-word Light keys message is used to light function and/or matrix keys on a keyboard. Bit 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

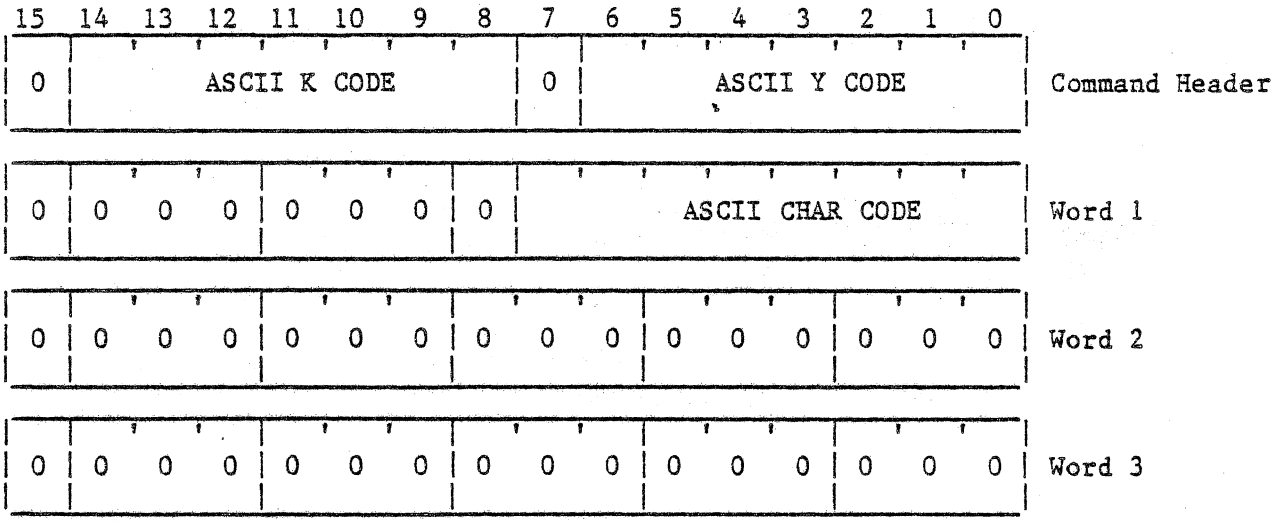
MATRIX KEYS

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

(G8→H) ALPHANUMERIC KEYBOARD

|KY| no. 1 , Command header code (octal): 045531

|KT| no. 2 045524



Alphanumeric Keyboard messages are associated with alphanumeric inputs from a keyboard. Each time an alphanumeric key is typed, GCP sends the message to the host computer if the following conditions are met:

a. The keyboard is enabled (refer to host-to-GRAPHIC 8 IK messages).

b. The keyboard is not being operated in the scratchpad mode (refer to host-to-GRAPHIC 8 Initialize Scratchpad 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 Alphanumeric Keyboard 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.

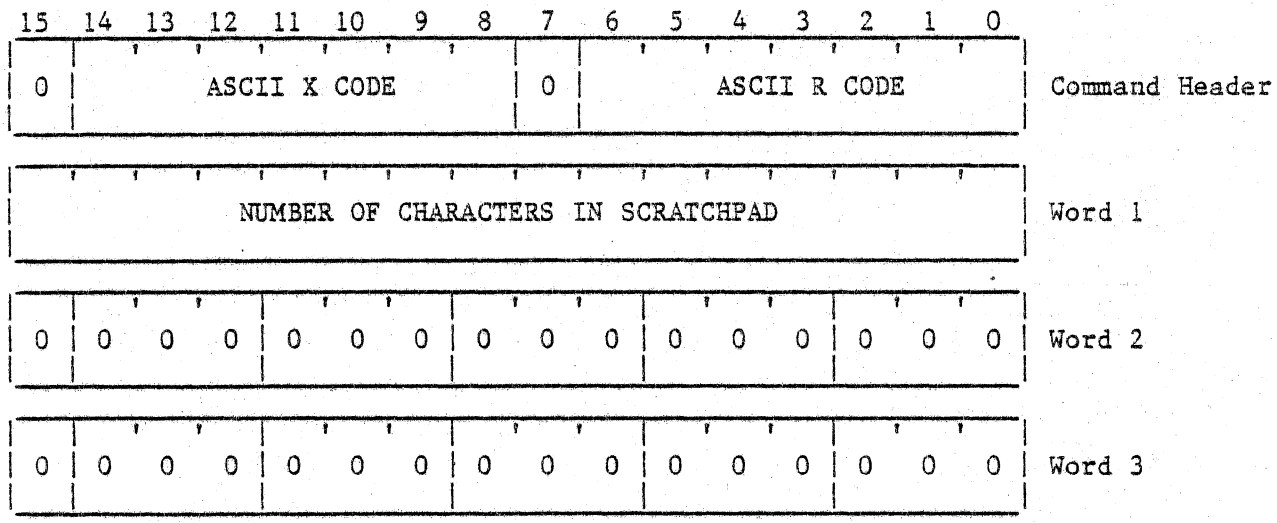
NOTE

Keyboards are automatically enabled by GCP when the GRAPHIC 8 is initialized in the system mode.

(G8→H) SCRATCHPAD READY FOR ALPHANUMERIC KEYBOARD

|XR| no. 1 Command header code (octal): 054122

|XT| no. 2 054124

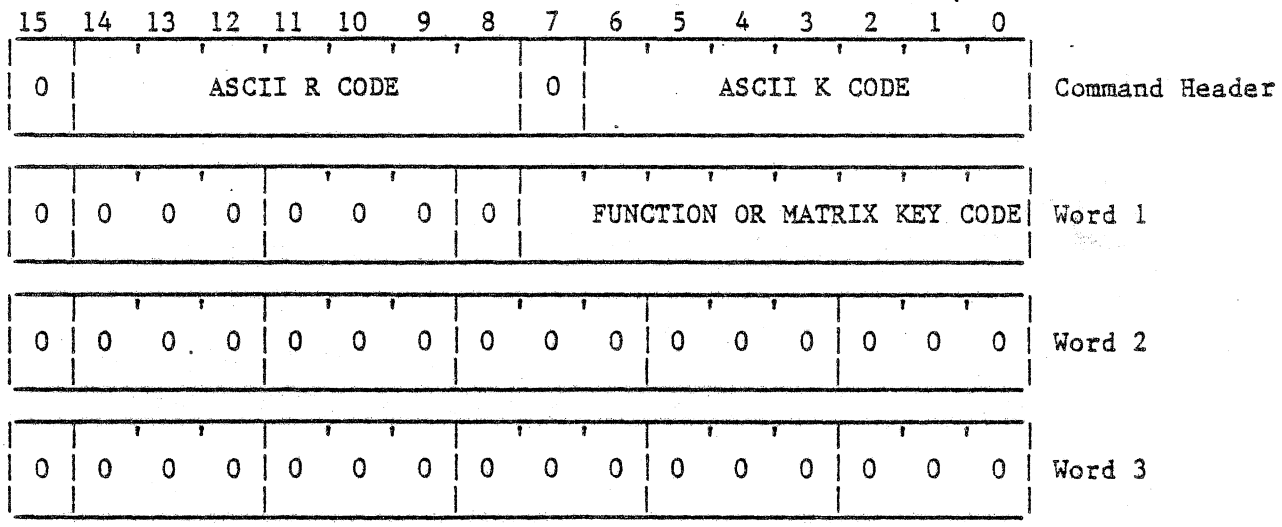


Scratchpad Ready messages are generated by GCP to inform the host computer that data in the scratchpad for the alphanumeric keyboard is ready to be transferred. GCP sends a Scratchpad Ready message to the host computer whenever an alphanumeric keyboard is operated in the scratchpad mode and the RETURN key is typed. GCP also sends this 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 obain the contents of the scratchpad.

(G8→H) FUNCTION KEYBOARD

|RK| no. 1 , Command header code (octal): 051113

|RL| no. 2 051114



Function Keyboard messages are associated with the function or matrix keys on the keyboard. If a function keyboard has been enabled (refer to host-to-GRAPHIC 8 IK message), GCP sends a Function Keyboard message to the host computer each time a key is typed. Each 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.

5.3.6 POSITIONAL ENTRY DEVICE RELATED MESSAGES*. The positional entry device related group consist of the following messages:

Host-to-GRAPHIC 8

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 8-to-Host

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

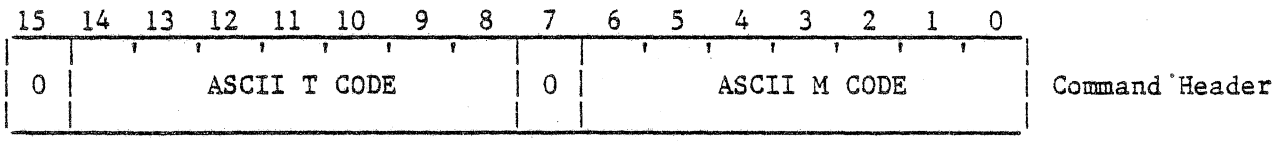
The following paragraphs discuss these messages and give details concerning the format and application of each.

*See Section 5.3.7 for messages for device numbers greater than 2.

(H→G8) ASSIGN DATA TABLET AS PED

|TM| no. 1 Command header code (octal): 052115

|TN| no. 2 052116



The Assign Data Tablet message is used to inform GCP that all messages received on ports 4 and 8 should be interpreted as data tablet type messages. By default GCP is initialized to interpret all messages received on ports 4 and 8 as trackball/forcestick type messages.

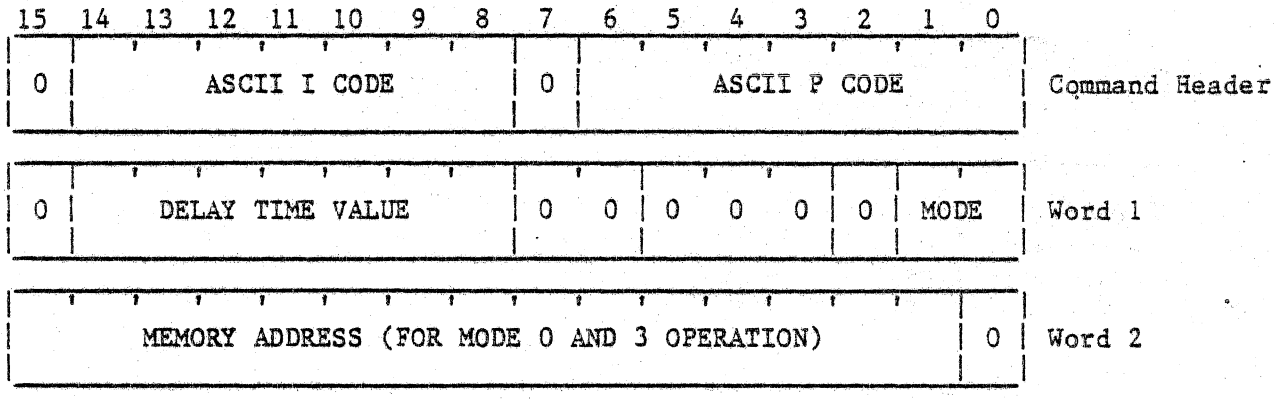
NOTE

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

(H→G8) INITIALIZE PED

|IP| no. 1 Command header code (octal): 044520

|IT| no. 2 044524



The Initialize PED message is a three-word message used to establish the operating mode for the PED (trackball/forcestick or data tablet). Bits 1 and 0 of word 1 specify the operating mode in binary form as follows: automatic tracking mode (mode 0) = 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 0 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 zeros. When mode 0 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 if an address of 0 is specified, the symbol is "detached" from PED action and further PED interrupts are disabled.

Mode 0, 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 0), 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 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 0 operation for a PED is specified, word 2 of the IP or IT 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.

The delay time values in word 1 associated with mode 0 are given below:

BIT	DELAY TIME VALUE (seconds)	
8	1/60	(or 1/50)*
9	2/60	(or 2/50)
10	4/60	(or 4/50)
11	8/60	(or 8/50)
12	16/60	(or 16/50)
13	32/60	(or 32/50)
14	64/60	(or 64/50)

The delay time applies to all PED's in mode 0 on the same GRAPHIC 8 controller.

When the bit is set to 1, the associated delay factor is activated. When the bit is set to 0, 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 approximately 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 a Return PED message to the host to reflect the latest position of the last data tablet entry. If the delay time is set for 0 seconds, then each time a data tablet message is received, a Return PED message is sent to the host computer. With a delay time of 0 seconds, the host computer could be overloaded with a series of identical Return PED messages. (E.g., if the data tablet pen switch remains pressed for 2 seconds, then 200 Return PED messages would have to be processed by the host computer.) The recommended delay time should be approximately 1/4 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 Return PED message is sent to the host computer until the new delay time expires.

When the automatic tracking mode is selected for the PED, the Give PED message must not be sent from the host computer to the GRAPHIC 8. If a message is sent to the GRAPHIC 8 when the PED 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 a Return PED message. This message is sent each time the display processor is interrupted by the PED. When the automatic mode is selected for the PED, the Give PED message must not be sent from the host computer to the GRAPHIC 8. If a GP message is sent to the GRAPHIC 8 when the PED is operating in the automatic mode, GCP responds by sending an XX message back to the host computer.

* For 50 cps power frequency

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. Movements of the PED is indicated 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 the PED is specified, word 2 of the IP or IT 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 automatically to the host computer in mode 1 operation while absolute position data is returned in mode 0, mode 2, and mode 3 operation, but only upon request of the host message (GP or OT).

When mode 3 operation for the PED is specified, word 2 of the initialize PED message (IP, IT, or IV) from the host computer contains the address of an LDXA or LDDI instruction. The LDXA is used to define (by user) a symbol or cursor in refresh as follows:

```
.... ;      specify x
LDXA x;     and y position of symbol
MVYA y;     graphic orders defining a symbol
......;     cursor (e.g., CHAR < * >)
```

A LDDI graphic order is used to define (by the user) a hardware crosshair cursor control routine as follows:

```
....
LDDI < XCRn, 0 >      ; 2 words, opcode + XCRn data
LDDI < YCRn, 0 >      ; 2 words, opcode + YCRn data
LDDI < STAn, 1040 >   ; 2 words to enable crosshair cursor
....
```

NOTE

The cross hair cursor can be disabled (removed from the screen) by using the graphic order:

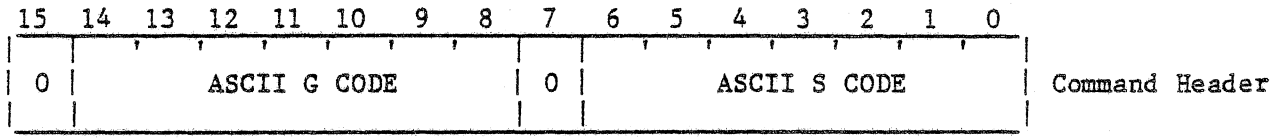
LDDI < STAn, 1000 >

The GCP input PED handler (in mode 3 or 0) tests for the presence of LDXA or LDDI (pointed to by word 2) and updates the X and Y position in refresh correctly for either case. If the GCP input PED handler does not recognize the LDXA or LDDI, the X and Y position in refresh file is not updated.

In summary:

Mode	Trackball Forcestick	Data Tablet
0	XXXXX	Update internal X, Y position and X, Y position in refresh file. Return X, Y when pen is released.
Automatic mode (1)	Send relative X, Y to host	XXXXX
Request mode (2)	Update internal X, Y position only	Update internal X, Y position
Tracking mode (3)	Update internal X, Y position and X, Y position in user refresh file.	Update internal X, Y position and X, Y position in user refresh file.

|GS| (H→G8) GET STATUS OF PEDS Command header code (octal): 043523

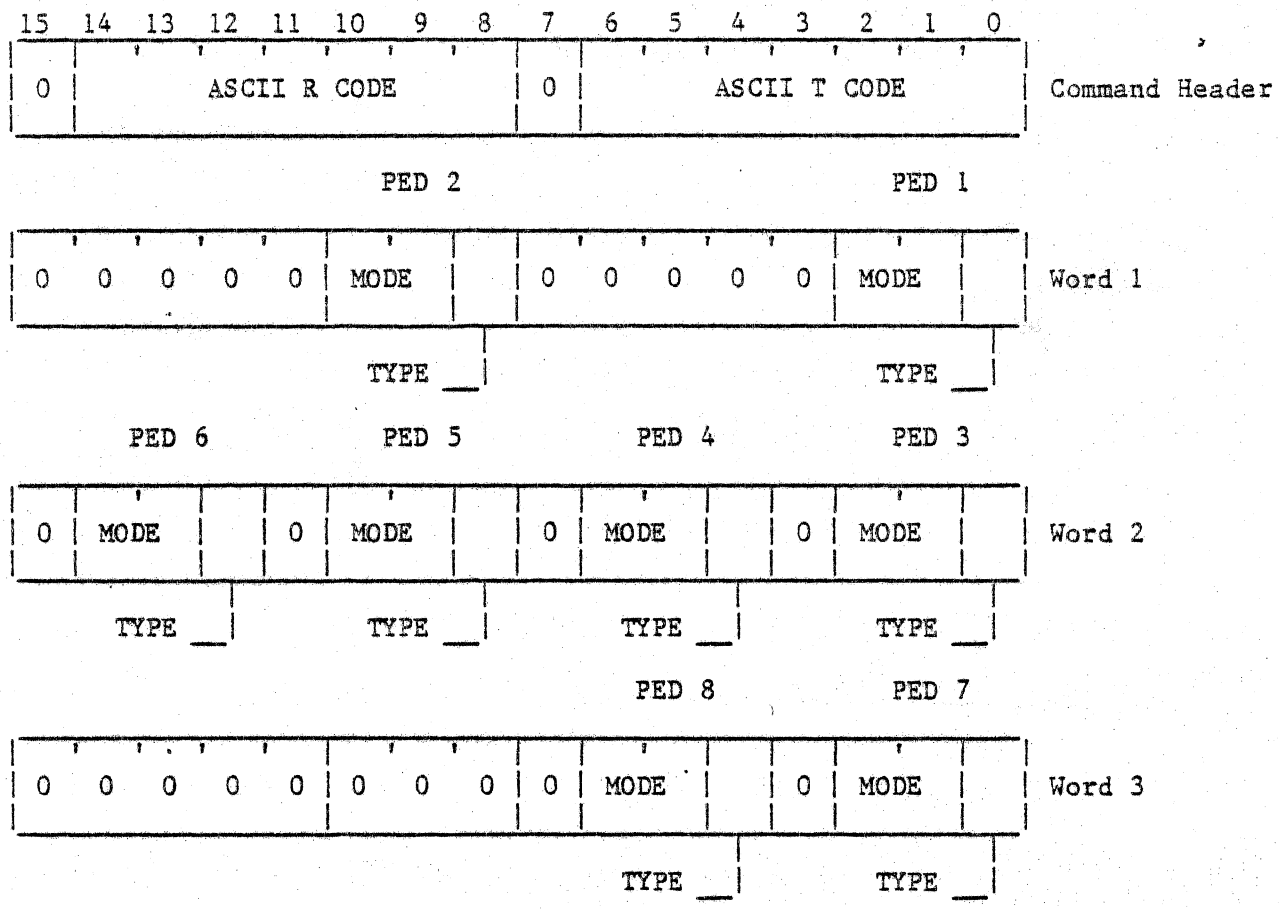


The GS message is used to request the current status of each PED. An RT message is sent by GCP to the host computer in response to the GS message.

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| (G8→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.

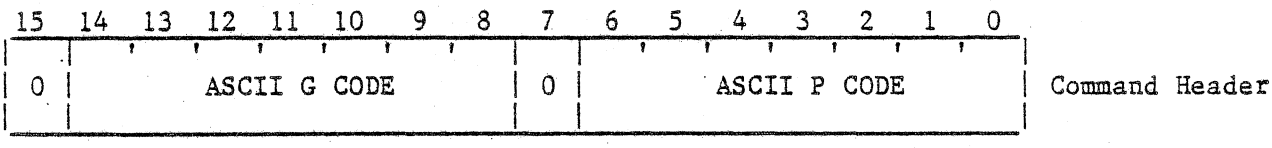
Bits 0, 1, and 2 are associated with PED 1 (PED connected to port 4 on serial multiport interface 1). Bits 8, 9 and 10 are associated with PED 2 (PED connected to port 8 on serial multiport interface 2). The meaning of the TYPE and MODE bits are given below:

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

(H→G8) GIVE PED

|GP| no. 1 Command header code (octal): 043520

|GT| no. 2 043524

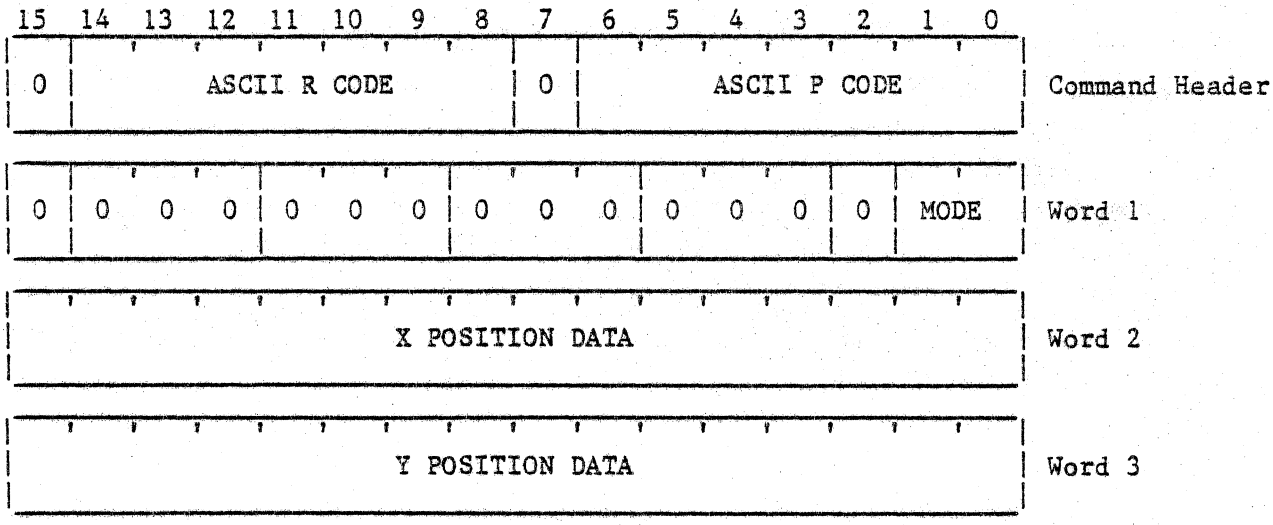


The Give PED message is a single-word message used to request the current absolute coordinate data for the PED (trackball/forcestick or data tablet). Requested data is returned by GCP to the host computer using a Return PED message. A Give PED message can be used only when the PED is operating in mode 2 (request mode) or mode 3 (tracking mode). If this message is sent when the PED is operating in mode 0 (automatic tracking mode) or in mode 1 (automatic mode), GCP responds with an XX (error status) message.

(G8→H) RETURN PED

|RP| no. 1 Command header code (octal): 051120

|RW| no. 2 051127



The Return PED messages are associated with the PED (trackball/forcestick or data tablet). When the PED is operating in the automatic tracking mode (mode 0) or in the automatic mode (mode 1), these messages are sent automatically by GCP. When the PED is operating in the request or tracking mode (modes 2 and 3, respectively), these messages are sent in response to Give PED messages from the host computer. The operating mode for PED is established by a corresponding Initialize PED message from the host computer. For all Return PED 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 the PED is operating in mode 0 (data tablet only) GCP sends a Return PED 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 the PED is operating in mode 1, GCP sends this message to the host computer every time the PED generates an interrupt to the display processor. PED interrupts are enabled or inhibited by host-to-GRAPHIC 8 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 the PED is operating in mode 2 or mode 3, GCP sends Return PED messages to the host computer in response to a Give PED message 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 8 is initialized in the system mode.

5.3.7 EXTENDED DEVICE CONTROL MESSAGE. The four Extended Device Control messages extend the functions described in Sections 5.3.5 and 5.3.6 to eight peripheral devices of the same device type.

HOST-to-GRAPHIC 8

IX - Enter Extended Device Control (EDC) mode

IV - Initialize a peripheral device

OU - Output or Request to Device

GRAPHIC 8-to-HOST

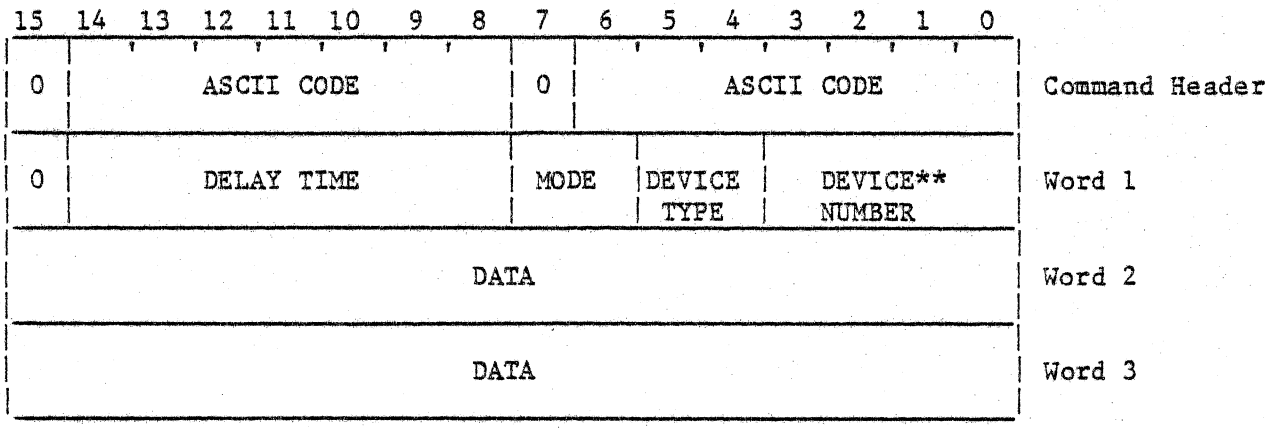
IN - Input from device

Refer to table 5-2 for the device types and the corresponding functions. Refer to the messages that the EDC messages functional replace for a complete description of the function.

Table 5-2. GCP Extended Device Control*
(Functional Replacement)

DEVICE TYPE (OCTAL) \ MESSAGE	INITIALIZE DEVICE IV	OUTPUT OR REQUEST DEVICE OT	INPUT DEVICE IN (4 WORDS)
0 Alphanumeric Keyboard	Initialize Scratchpad (ZR, ZT)	Zero Scratchpad (ZS, ZU)	Mode = 0, Scratchpad Ready (XR, XT)
			Mode = 1, Character data (KY, KT)
1 Function Keyboard		Light function keys (LK, LT)	Function or Matrix Key Code (RK, RL)
2 Trackball/ Forcestick	Initialize PED (IP, IT)	Request Position data (GP, GT)	X and Y position data (RP, RW)
3 Tablet	Initialize PED as Tablet (IP,IT)+(TM,TN)	Request Position data (GP, GT)¯	X and Y position data (RP, RW)

General Message Format:



*Enter Extended Device Control (EDC) mode via an IX message. GCP initialization in system mode is in non EDC mode.
**Keyboard or PED is device no. n-1.

The format of words 1, 2, and 3 for each device type and message are shown below.

|IV| — Initialize Device Command Header (octal 44126)

- Initialize Scratchpad

  Word 1 - device type = 0; device number = 0-7
  Word 2 - starting address of scratchpad in refresh file
  Word 3 - number of characters in line in scratchpad

- Initialize PED

  Word 1 - delay time; mode = 1, 2, or 3; type = 2; number = 0-7
  Word 2 - address of LDXA of PED symbol in refresh file if software symbol
  - address of LDDI if hardware cursor

- Initialize PED as data tablet

  Word 1 - delay time; mode = 0, 2, or 3; type = 3, number = 0-7
  Word 2 - address of LDXA of tablet symbol in refresh file if software symbol
  - address of LDDI if hardware curosr

|OT| — Output or Request Device Command Header (octal 47524)

- Zero Scratchpad

  Word 1 - type = 0; device number = 0-7

- Light Function Keys

  Word 1 - type = 1; device number = 0-7
  Word 2 - mask for function key lights
  Word 3 - mask for matrix key lights

- Request PED Position Data

  Word 1 - Mode 2 or 3, type = 2, number = 0-7

- Request Tablet Position Data

  Word 1 - Mode 2 or 3, type = 3, number = 0-7

|IN| — Input Device Command Header (octal 44516)

- Scratchpad Ready

  Word 1 - mode = 0; type = 0; number = 0-7
  Word 2 - number of characters in the scratchpad
  Word 3 - 0

- Character Data

  Word 1 - mode = 1, type = 0; device number = 0-7
  Word 2 - ASCII character code in bits 0-7
  Word 3 - 0

- Function or Matrix Key Data

  Word 1 - type = 1; device number 0-7
  Word 2 - Function key code in bits 0-7
  Word 3 - 0

- X and Y PED Position Data

  Word 1 - mode = 1, 2, or 3; type = 2; number = 0-7
  Word 2 - X position coordinate
  Word 3 - Y position coordinate

- X and Y Tablet Position Data

  Word 1 - mode = 0, 2, or 3; type = 3; number = 0-7
  Word 2 - X position coordinate
  Word 3 - Y position coordinate

5.3.8 FORTRAN SUPPORT (FSP) MESSAGES. The Fortran support (FSP) group consists of the following messages:

Host-to-GRAPHIC 8

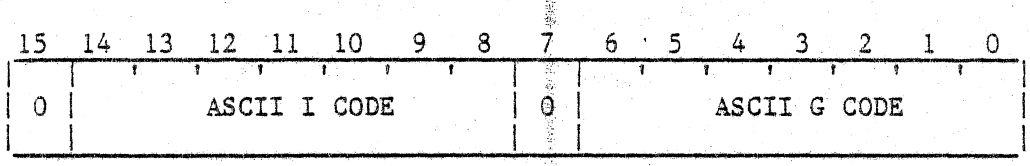
IG	Initialize GCP to support FSP
GU	Graphic update
MI	Move image
NP	Enable box display
ZP	Disable box display
NN	Enable error number
ZN	Disable error number
PV	Packed vector

GRAPHIC 8-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→G8) INITIALIZE FSP SUPPORT Command header code (octal): 044507



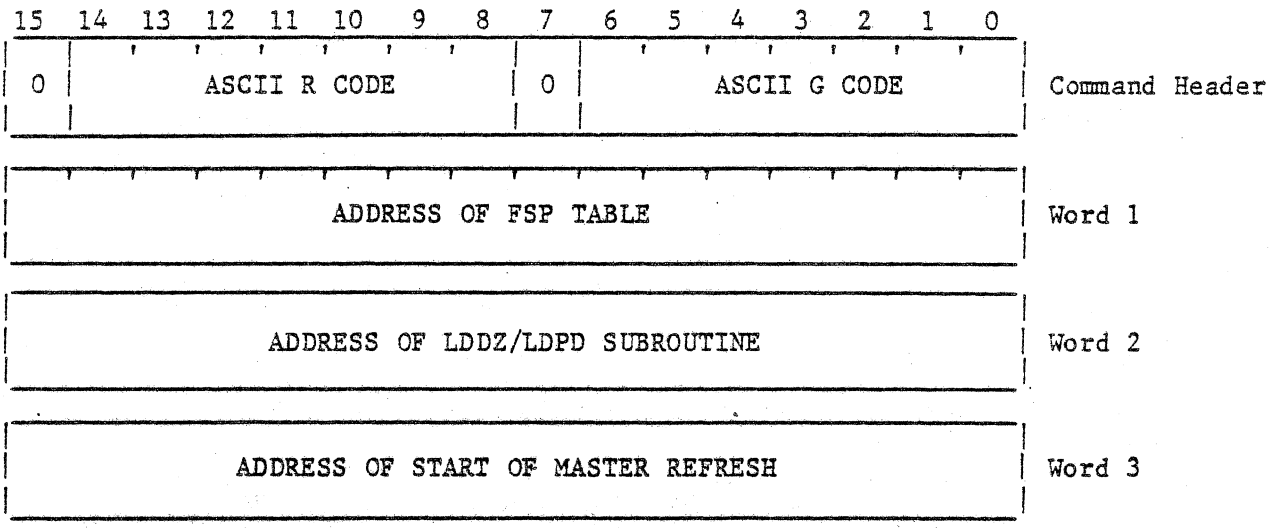
The IG message is used to initialize GCP to operate in the Fortran support program (FSP) environment. Associated with this environment is a Sanders-developed Fortran Graphic Support program. This program is host resident and consists of a collection of Fortran callable subroutines. This program simplifies the task of generating a 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.

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 GRPAHIC 8. FSP uses these addresses to manage the refresh program associated with the FSP environment.

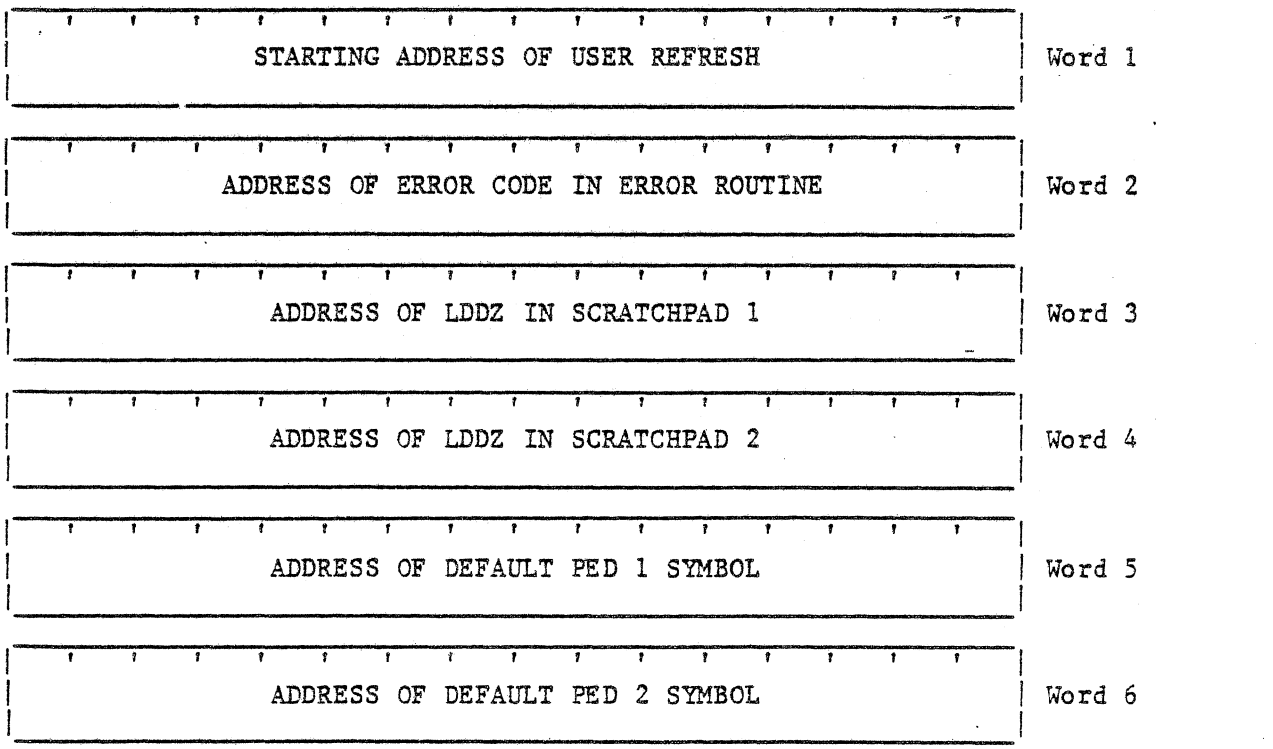
NOTE

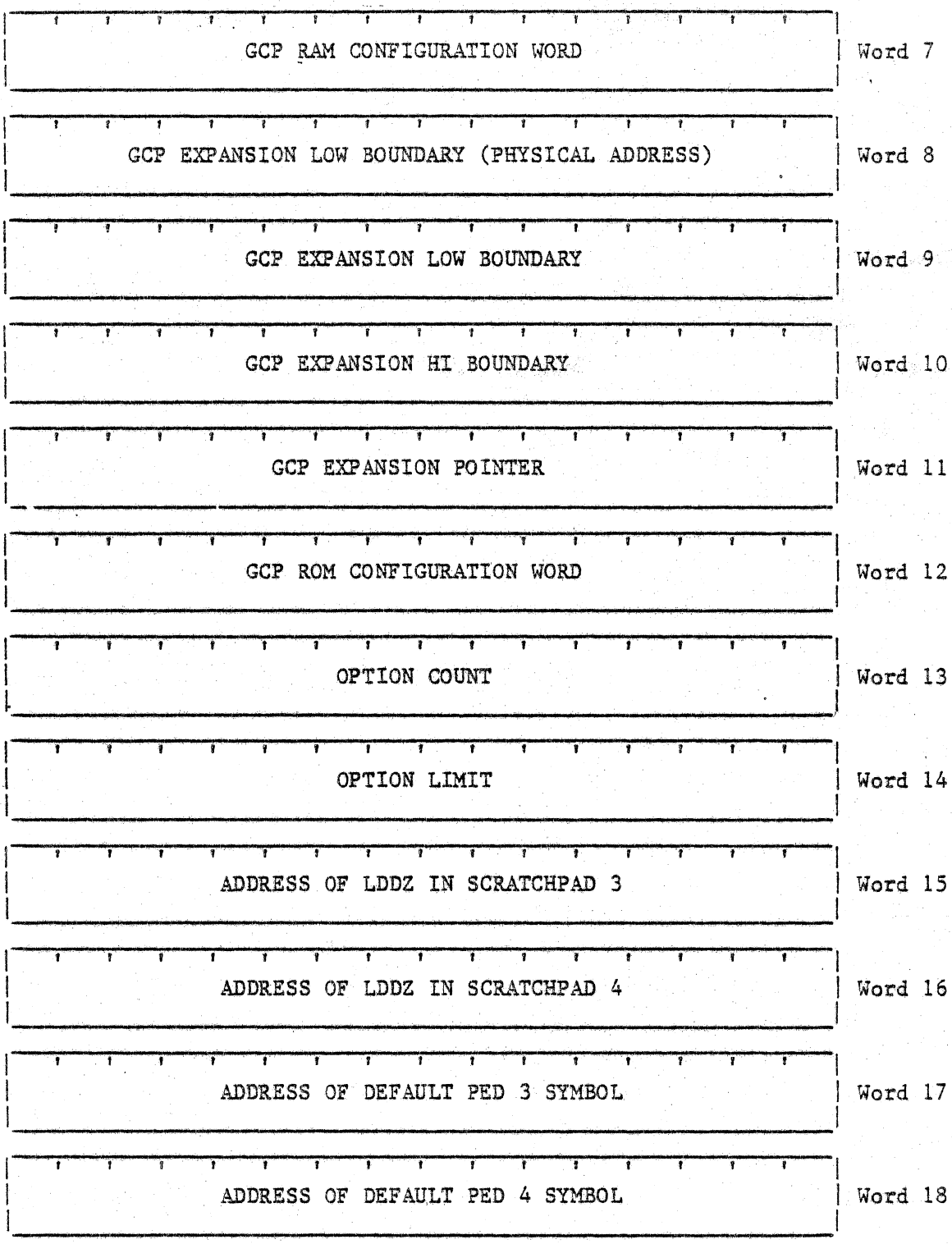
Although the IG message is primarily intended for use in the FSP environment, the user has the option of developing his own host package to communicate with GCP in the FSP environment.

|RG| (G8—>H) RETURN FSP TABLE ADDRESS Command header code (octal): 051107

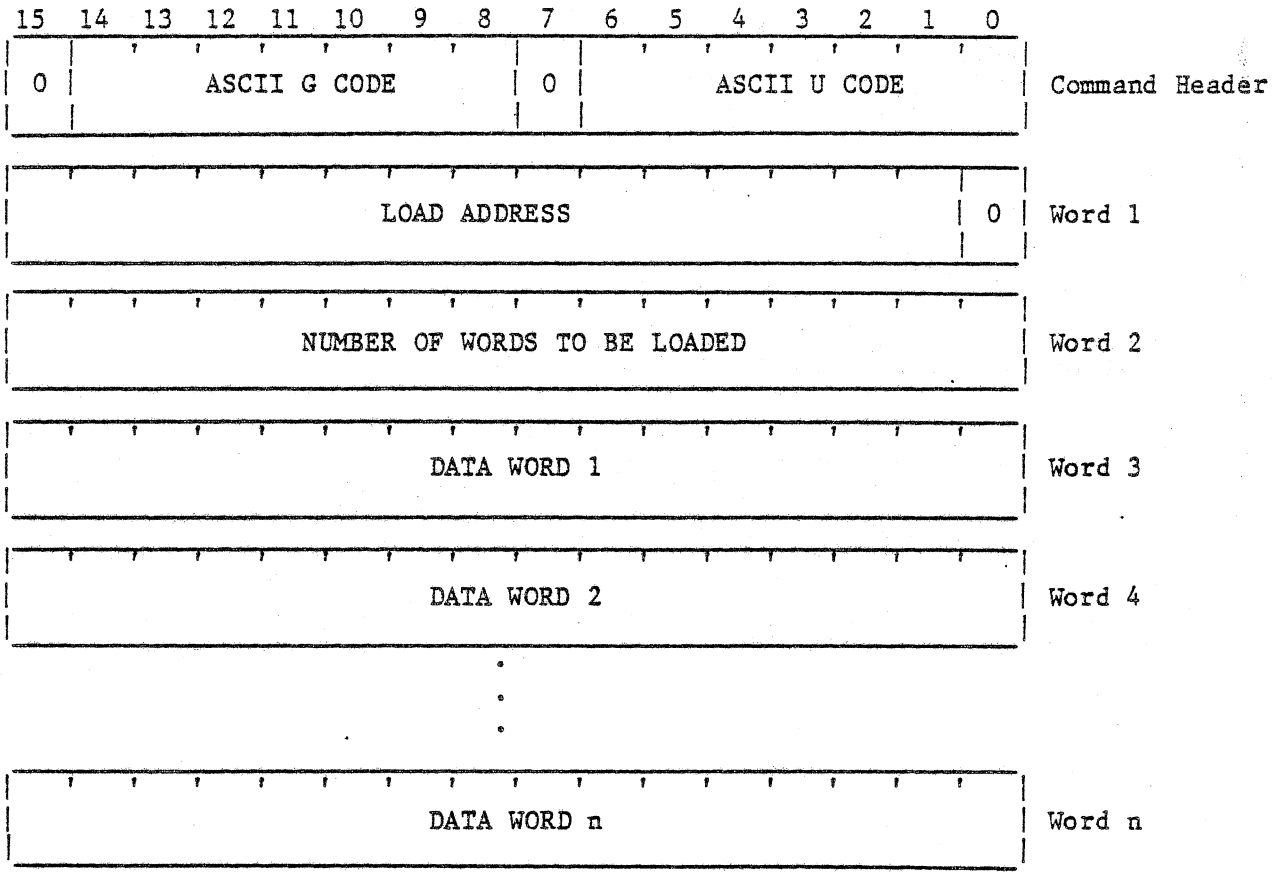


The RG message is returned to the host computer in response to the IG message. Word 1 contains the starting address of the FSP table in GRAPHIC 8 memory. Word 2 contains the address of the LDDZ instruction loaded by GCP to support GRAPHIC 7 FSP. GRAPHIC 8 FSP will replace this LDDZ with an LDPD. Word 3 contains the address of the start of Master Refresh which controls System and User Refresh. The FSP 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 1 (octal). The FSP table always resides in bank 0 and contains the following information:




GCP RAM CONFIGURATION WORD
Word 7
GCP EXPANSION LOW BOUNDARY (PHYSICAL ADDRESS)
Word 8
GCP EXPANSION LOW BOUNDARY
Word 9
GCP EXPANSION HI BOUNDARY
Word 10
GCP EXPANSION POINTER
Word 11
GCP ROM CONFIGURATION WORD
Word 12
OPTION COUNT
Word 13
OPTION LIMIT
Word 14
ADDRESS OF LDDZ IN SCRATCHPAD 3
Word 15
ADDRESS OF LDDZ IN SCRATCHPAD 4
Word 16
ADDRESS OF DEFAULT PED 3 SYMBOL
Word 17
ADDRESS OF DEFAULT PED 4 SYMBOL
Word 18

|GU| (H→G8) GRAPHIC UPDATE Command header code (octal): 043525

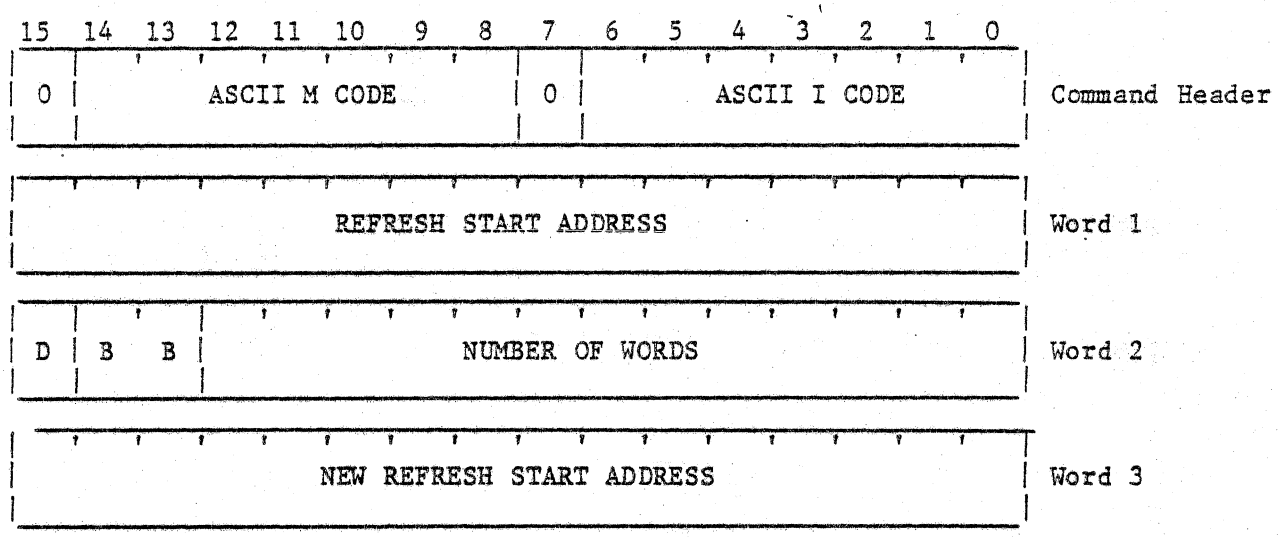


The GU message is a variable-length message used to load data into read/write memory of the GRAPHIC 8. This message is primarily intended for use in the FSP environment but it can also be used by the GCP application programmer. The GU message is a special form of the MU and SU messages.

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).

MI (H→G8) 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

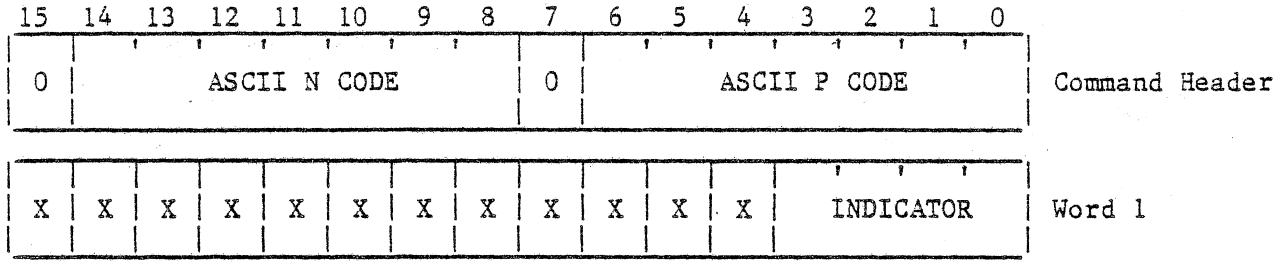
0	Transfer refresh data to current bank (ignore bits 13 and 14)
1	Transfer refresh data to bank specified in bits 13 and 14

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 (RTRN) instruction is appended to the refresh data moved.

|NP| (H→G8) ENABLE BOX DISPLAY Command header code (octal): 047120

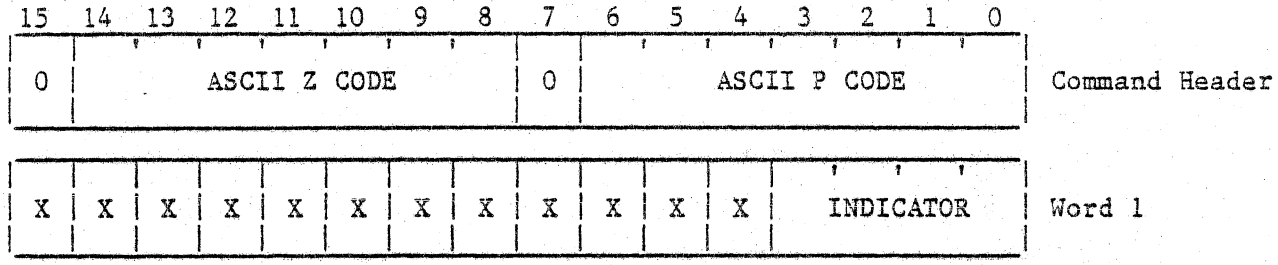


The NP message is used to enable the box display on selected indicators when operating in the FSP environment. Bits 4 through 15 in word 1 are ignored by GCP. Bits 0 through 3 specify which indicators the box should be displayed on. These bits are defined as follows:

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->G8) DISABLE BOX DISPLAY Command header code (octal): 055120



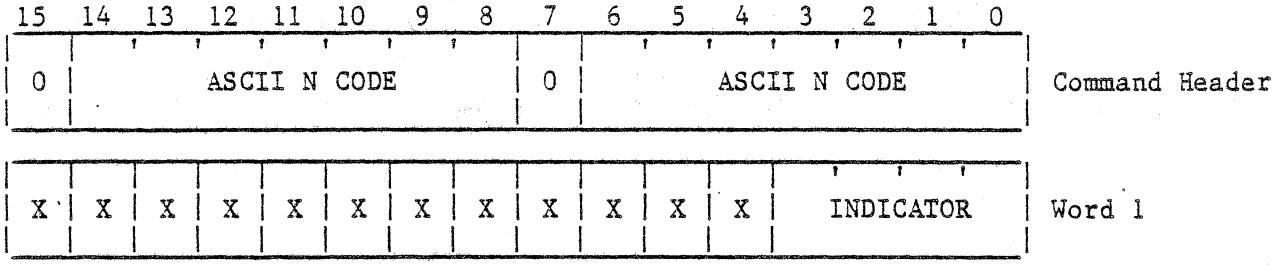
The ZP message is used to disable the box display on selected indicators when operating in the FSP environment. Bits 4 through 15 in word 1 are ignored by GCP. Bits 0 through 3 specify which indicators the box should be removed from. These bits are defined as follows.

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| (H→G8) ENABLE ERROR NUMBER Command header code (octal): 047116

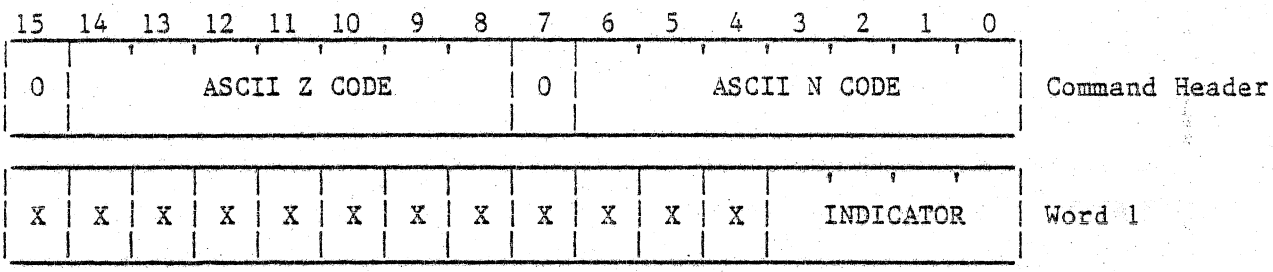


The NN message is used to enable the error number display on selected indicators when operating in the FSP environment. These error numbers are updated by FSP to give the user a visual indication that an error has occurred. Bits 4 through 15 are ignored by GCP. Bits 0 through 3 specify which indicators 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->G8) DISABLE ERROR NUMBER Command header code (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:

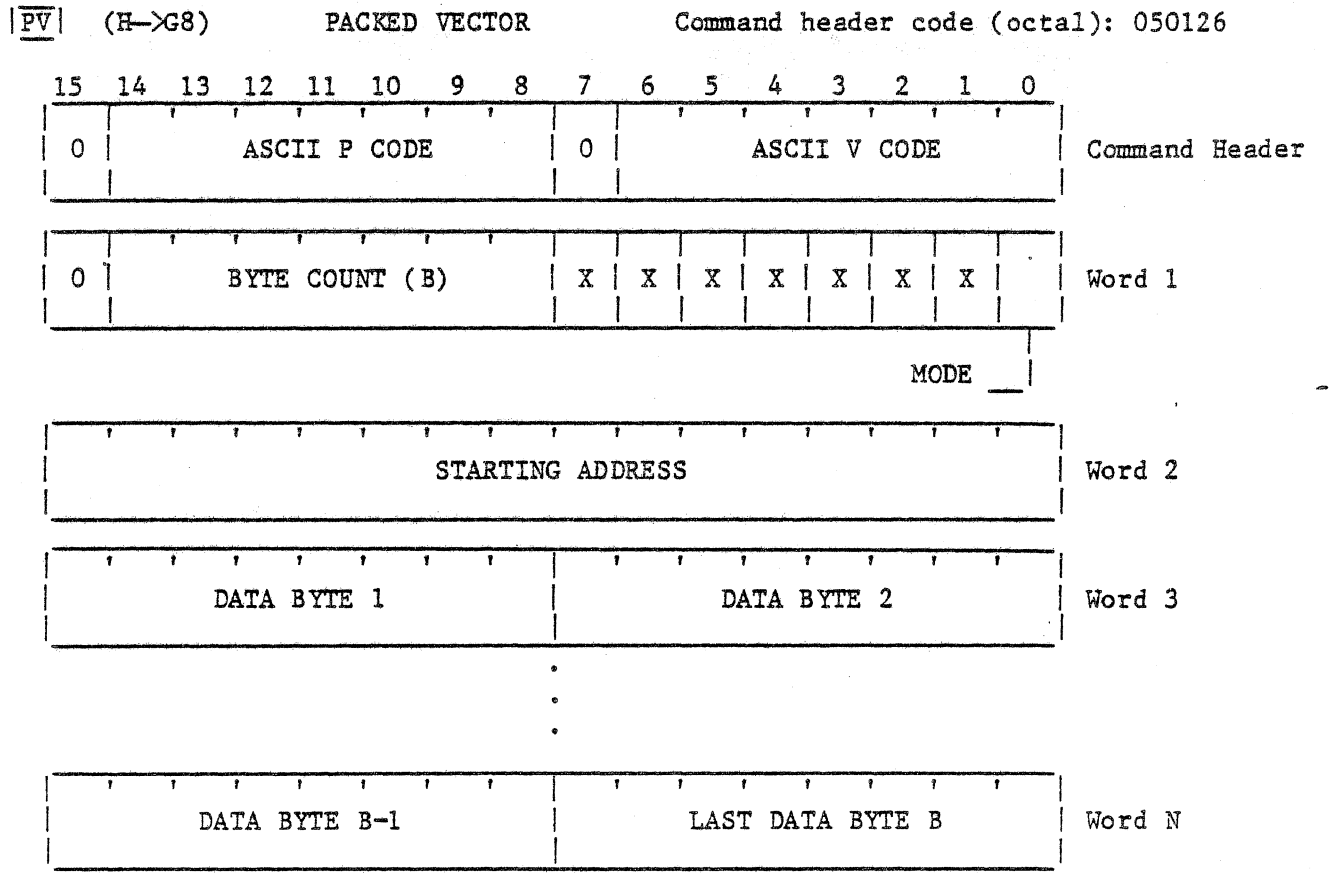
BITS 3 2 1 0	
1 0 0 0	Remove error number from indicator 1
0 1 0 0	Remove error number from indicator 2
0 0 1 0	Remove error number from indicator 3
0 0 0 1	Remove error number from indicator 4

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

5.3.8.1 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 8 to get these instructions stored in refresh.

When packed vector mode is used, a coded PV message is sent to the GRAPHIC 8. The PV message contains a series of ASCII characters that reflect the moves and draws that should be stored in refresh. The GRAPHIC 8 decodes the PV message and generates the equivalent LDXA, MVYA, and DRYA instructions and stores them in refresh.

The PV message is given below:



Bits 8 through 14 in word 1 contain the byte count indicating the number (B) of data bytes contained in the PV message. Bit 0 in word 1 selects the mode. When bit 0 is set to 0, add mode is selected. For add mode, an RTRN, return instruction is added to the end of the refresh code created from the data bytes contained in the PV message.

When bit 0 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 8, 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 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 X or Y 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 5-3 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 5-3. 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	
HY 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 5-3 for further clarification of meaning of bytes sent. A 0 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 8, 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 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.

5.3.9 OPTION SUPPORT. Software options allow the GRAPHIC 8 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. 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 8 end).

2. Customer-developed software to meet a unique requirement (e.g., additional GCP messages to permit local editing of text at the GRAPHIC 8 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.

5.3.9.1 Option Messages. The option group consist of the following messages:

Host-to-GRAPHIC 8

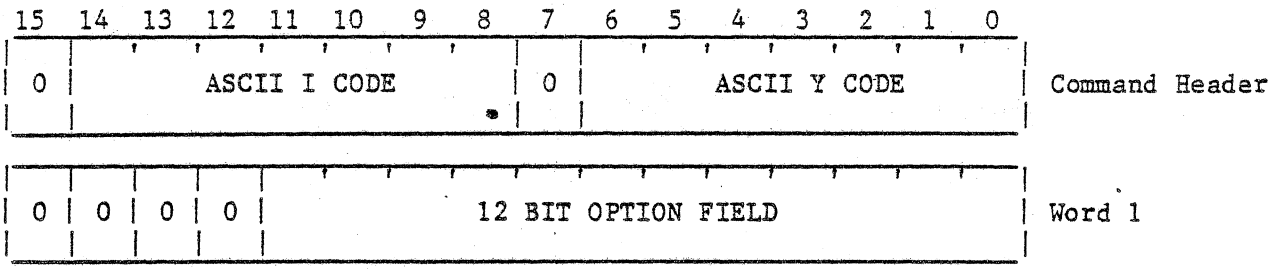
IY	Initialize 2
GO	Give option status
OU	Option update (host downloading - described in option manual, H-79-0357)

GRAPHIC 8-to-Host

RO	Return option

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

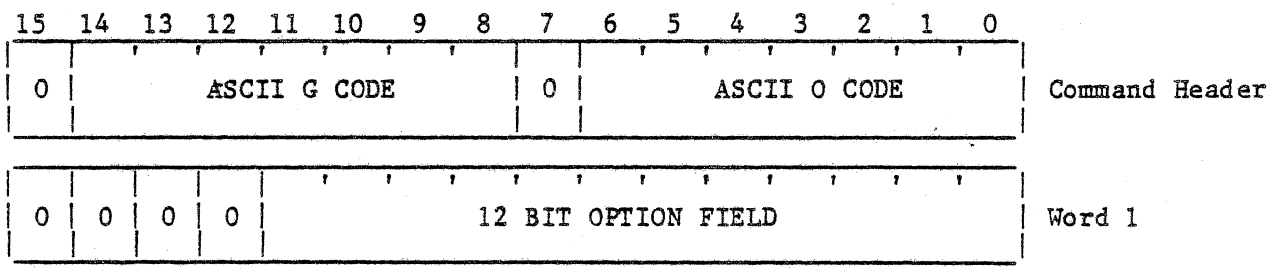
|IY| (H→G8) INITIALIZE 2 Command header code (octal): 044531



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

(OCTAL) OPTION FIELD	ACTION
0	Load all system automatic load options
4000	Unload all options
1 to 3777 4001 to 7777	Load specified option (if unloaded), initialize option, and update option status

|GO| (H→G8) GIVE OPTION STATUS Command header code (octal): 043517



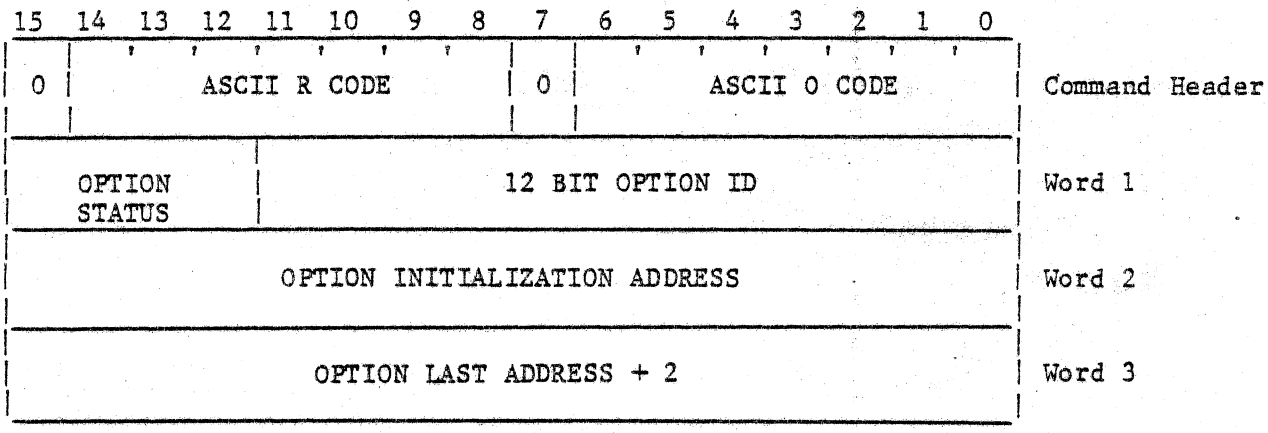
The give option status message is a two word message which allows the host to verify an option's status. One or two messages will be returned to the host as specified below.

OPTION FIELD	ACTION
0	Return status of all options via R0, VL
Non-zero	Return status of specified option via R0

Two styles of R0 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.

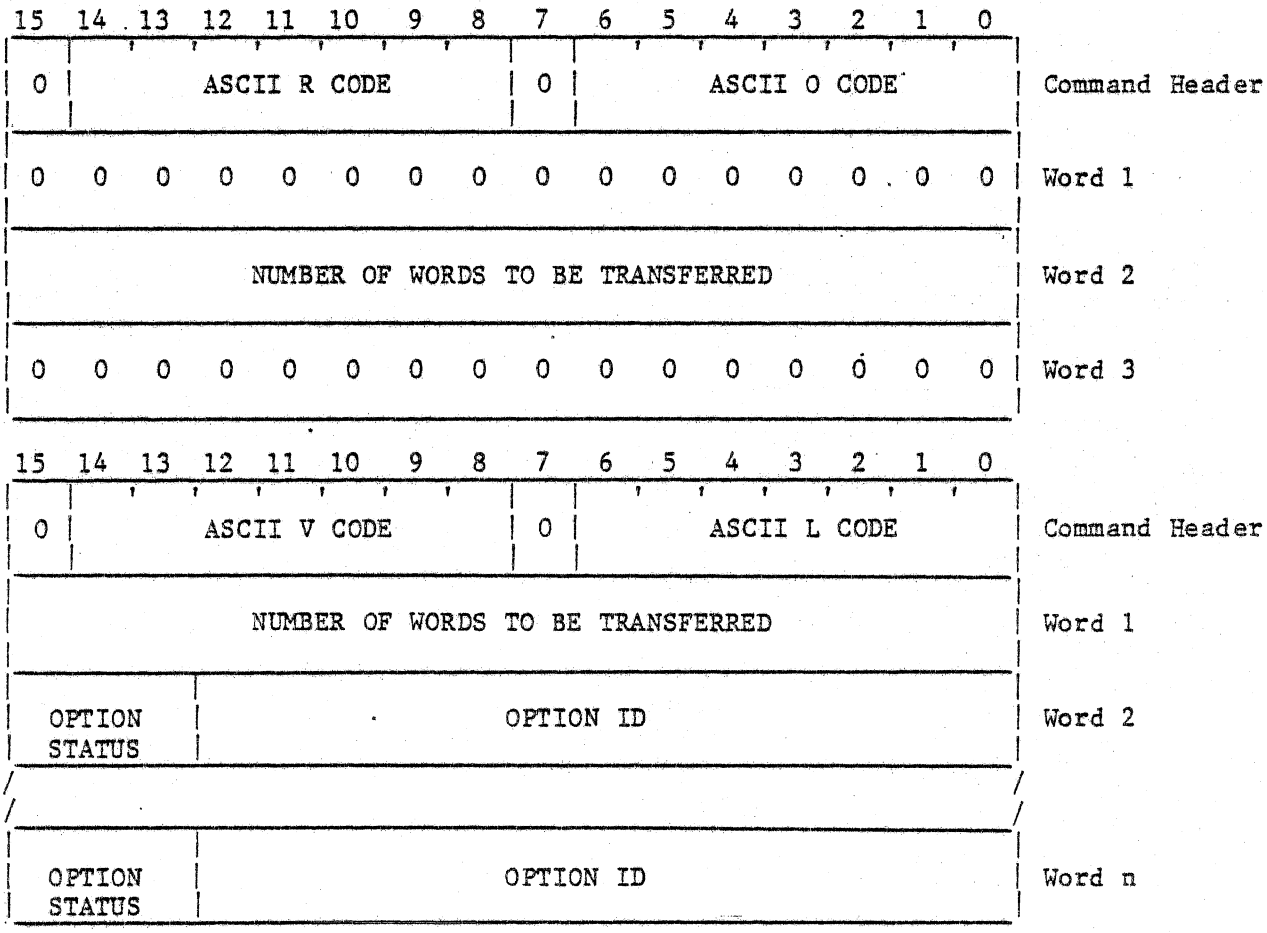
|RO| (G8→H) RETURN OPTION Command header code (octal): 051117

Single option status return.



|RO| (G8→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 0 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 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
- Selectively establish the desired interrupt control

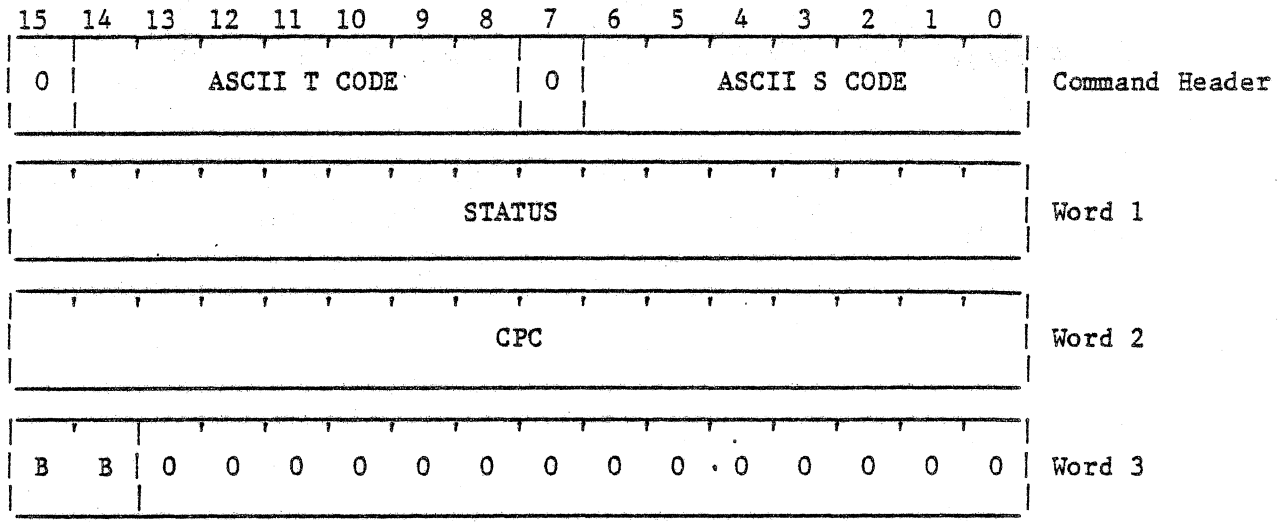
When 3-D interrupts are generated, an appropriate TS message is returned to the host computer.

NOTE

Please refer to Sanders Publication H-79-0305 for more information on the 3-D coordinate converter.

|TS| (G8—>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 8 equipment.

6.2 STARTUP PROCEDURES

Startup procedures consist of initializing the GRAPHIC 8 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 8 TURNED ON AFTER HOST COMPUTER. When power is applied to the GRAPHIC 8 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 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 8 TURNED ON BEFORE HOST COMPUTER. If the GRAPHIC 8 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 8. This is done by sending an IZ (initialize) message to the GRAPHIC 8 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 8 and the host computer and that power is first restored to the GRAPHIC 8. This condition is similar to that described in paragraph 6.2.2 in that the XX message automatically generated by the GRAPHIC 8 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 8 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 8 IN TELETYPEWRITER EMULATION MODE. A special startup procedure is required when the teletypewriter emulation capability of the GRAPHIC 8 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 8. This character causes the GRAPHIC 8 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 8 is in the teletypewriter emulation mode). The resulting initializaiton 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.

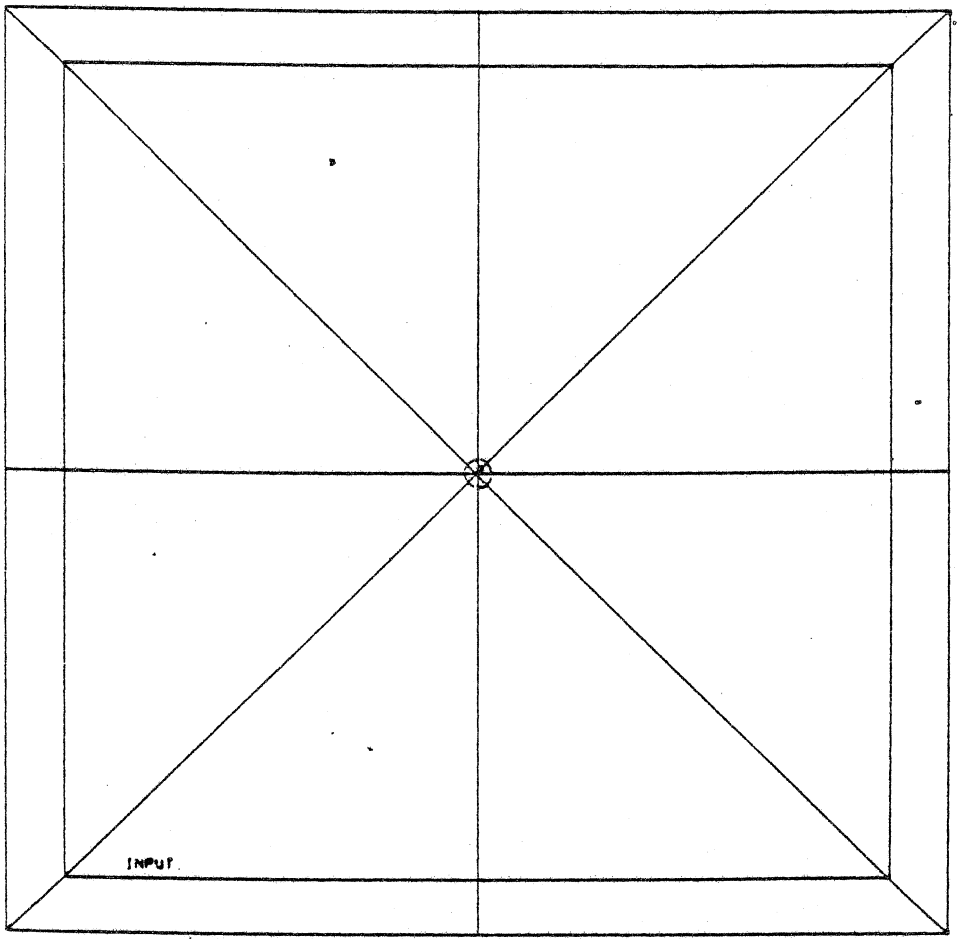
6.3 REFRESH FILES

The following paragraphs describe the generation, transmission, and alteration of refresh files to be processed by the digital 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 8 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 8. The first part of the file (before the label RLOOP) initializes parameters as required to ensure the proper interpretation of the instructions that follow.

Following the WATE instruction is the sequence of instructions used to display 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.



GA-77-419-07

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

Table 6-1. Sample Refresh File No. 1

```
00100                    ;
00200                    ; SET DISPLAY PARAMETERS
00300                    ;
00400  003000  013707            LDDZ <ALL, BLOFF, LINE, BR7>
00500  003002  014010            LDDP <NOROTATE, CSO>
00600  003004  140114            LDTI 14
00700  003006           RLOOP:
00800  003006  005000            NOOP
00900  003010  020000            LDXA 0
01000  003012  060000            MVYA 0                  ;CENTER AT 0,0
01100  003014  007000            WATE
01200                    ;
01300                    ; DRAW DIAGONALS
01400                    ;
01500  003016  020777            LDXA 777                ;MOVE TO
01600  003020  060777            MVYA 777                ; UPPER RIGHT
01700  003022  023000            LDXA -1000              ;DRAW DIAGONAL
01800  003024  043000            DRYA -1000              ; TO LOWER LEFT
01900  003026  050777            MVXA 777                ;MOVE TO LOWER RIGHT
02000  003030  023000            LDXA -1000              ;DRAW DIAGONAL
02100  003032  040777            DRYA 777                ; TO UPPER LEFT
02200  003034  050000            MVXA 0                  ;MOVE TO TOP CENTER
02300  003036  043000            DRYA -1000              ;DRAW STRAIGHT DOWN
02400  003040  020777            LDXA 777                ;MOVE TO FAR
02500  003042  060000            MVYA 0                  ; RIGHT CENTER
02600  003044  033000            DRXA -1000              ;DRAW HORIZONTAL TO LEFT
02700                    ;
02800                    ; DRAW INSIDE SQUARE
02900                    ;
03000  003046  063000            MVYA -1000              ;MOVE TO BOTTOM LEFT
03100  003050  030777            DRXA 777                ;DRAW STRAIGHT UP
03200  003052  040777            DRYA 777                ;DRAW TOP EDGE TO RIGHT
03300  003054  033000            DRXA -1000              ;DRAW RIGHT EDGE DOWN
03400  003056  043000            DRYA -1000              ;DRAW BOTTOM EDGE TO LEFT
03500                    ;
03600                    ; DRAW INNER SQUARE
03700                    ;
03800  003060  023076            LDXA -702               ;SET INSIDE POINT
03900  003062  063076            MVYA -702               ; AT LOWER LEFT
04000  003064  045604            DRYR 1604               ;DRAW STRAIGHT UP
04100  003066  035604            DRXR 1604               ;DRAW TOP EDGE TO RIGHT
04200  003070  046174            DRYR -1604              ;DRAW STRAIGHT DOWN
04300  003072  036174            DRXR -1604              ;DRAW BOTTOM EDGE TO LEFT
04400                    ;
04500                    ; SCRATCHPAD PROMPTER          04600                    ;
04700  003074  023200            LDXA -600               ;POSITION INSIDE INNER
04800  003076  063110            MVYA -670               ; SQUARE AT LOWER LEFT
04900  003100  147311            TXT I,N                 ;INSERT 'INPUT:'
05000  003102  152720            TXT P,U
05100  003104  135324            TXT T,:
```

Table 6-1. Sample Refresh File No. 1 (Cont)

```
05200                    ;
05300                    ; SCRATCHPAD
05400                    ;
05500  003106            SCRPD:
05600  003106  120240        *   TXT < >,< >              ;TEN
05700  003110  120240            TXT < >,< >              ; SPACES
05800  003112  120240            TXT < >,< >              ;  FOR
05900  003114  120240            TXT < >,< >              ;   SCRATCHPAD
06000  003116  120240            TXT < >,< >              ;    AREA
06100                    ;
06200                    ; PED CONTROLLED MOVING CIRCLE
06300                    ;
06400  003120            MCRCLE:
06500  003120  020000            LDXA 0                   ;PED CONTROLLED
06600  003122  060000            MVYA 0                   ; POSITION OF
06700  003124  073020            LDKX 3,20                ;  SMALL
06800  003126  077020            DRKY 3,20                ;   CIRCLE
06900  003130  001000            JUMP RLOOP               ;REPEAT
07000  003132  002006
```

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, the refresh file is terminated with a JUMP instruction. The JUMP instruction causes the digital graphic controller to loop back to the point 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 digital graphic controller. The listing specifies that 46 words are required and that they are to be loaded into read/write memory beginning at octal location 3000 and ending at octal location 3132. Figure 6-1 illustrates the display that is created when this refresh file is processed by the digital 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 8 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 8):

a. Host-to-GRAPHIC 8 MU (memory update) message:

046525 - MU command header

003000 - load address for first data word

000056 - number of data words to be loaded

013007 - first data word to be loaded

014010 - second data word to be loaded

.
.
.

002006 - last data word to be loaded

b. Host-to-GRAPHIC 8 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 8 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 8 IS (enabled selected interrupts) message:

044523 - IS command header

000002 - enable halt interrupt

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

b. Host-to-GRAPHIC 8 SU (selective update) message:

051525 - SU command header

002100 - 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. After displaying "READY", the digital graphic controller halts and interrupts the display processor. This causes GCP to send a HI message to the host computer.

c. GRAPHIC 8-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.

d. Host-to-GRAPHIC 8 KP (continue picture) message:

045520 - KP command header

Each time an HI message is received from the GRAPHIC 8, the host computer must respond with a KP message to restart the graphic controller.

e. Host-to-GRAPHIC 8 MU (memory update) messages:

046525 - MU command header

002100 - 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 8 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 8.

6.4 OPTIONAL EQUIPMENT USAGE

Optional GRAPHIC 8 equipment includes keyboards, PEDs, and a hard copy unit. At the time the GRAPHIC 8 is initialized in the system mode, keyboards and PEDs are automatically enabled. Following initialization, these devices are enabled and disabled as required by IK (interrupt control) messages from the host computer to the GRAPHIC 8. The hard copy unit is controlled only by pressing the start button on the hardcopy unit. The following paragraphs provide examples of how to control each type of optional equipment (unless otherwise stated, the examples are assumed to refer to alphanumeric keyboard no. 1, and PED no. 1, in non-EDC mode).

6.4.1 KEYBOARDS. After a keyboard has been enabed, 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. For 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 8 ZR (initialize scratchpad for alphanumeric keyboard no. 1) message:

055122 - ZR command header

002100 - starting address (first address of scratchpad)

000012 - number of characters in line (ten)

Once this message has been received by the GRAPHIC 8, 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 independent 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 8-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 8 message:

043511 - GI command header

002100 - 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 8-to-host RI (return image) message:

051111 - RI command header

002100 - starting address (first address of scratchpad)

000005 - number of words to be transferred

000000 - filler

b. GRAPHIC 8-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 8 to clear the scratchpad area:

Host-to-GRAPHIC 8 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 0), 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 8 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 0 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 8 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 8. 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 8 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 8 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 8, GCP automatically enables the interrupt associated with the PED.

6.5 MULTISTATION USAGE

In the discussions in paragraphs 6.3 and 6.4, it was assumed that identical images were desired on any display monitor or hard copy unit that might be enabled. It is possible, however, to send different images to each display indicator if required. The images may have elements in common or may be entirely different.

Table 6-2 is a listing for a refresh file that causes an image similar to that shown in figure 6-1 to appear on each of two display monitor (no. 1 and 2 are assumed for purposes of illustration). This refresh file causes the squares and the intersecting vectors to be drawn simultaneously on both displays. Then, only display monitor no. 1 is enabled while "INPUT:" is drawn, the scratchpad for keyboard no. 1 is initialized, and the circle for PED no. 1 is drawn. Finally, only display monitor no. 2 is enabled while the corresponding elements are drawn for its display.

At this point, the images on both display monitors should appear to be identical. The scratchpad, and circle of each display, however, can be controlled separately by the associated peripheral devices. It is only necessary for the host computer application program to recognize the discrete interrupts caused by each peripheral device. Messages similar to those discussed in paragraph 6.4 would be used by GCP and the host computer but data for each display indicator would be transmitted in separate messages. Thus it would be possible for the circle to appear in different positions on each display and for different text to appear in each scratchpad area.

Table 6-2. Sample Refresh File No. 2

```
00100                       ;
00200                       ; SET DISPLAY PARAMETERS
00300                       ;
00400  003000  014010              LDDP <NOROTATE, CSO>
00500  003002  140114              LDTI 14
00600  003004               RLOOP:
00700  003004  005000              NOOP
00800  003006  013407              LDDZ <CRT1, CRT2, BLOFF, LINE, BR7>
00900  003010  020000              LDXA 0                  ;CENTER AT 0,0
01000  003012  060000              MVYA 0
01100  003014  007000              WATE
01200                       ;
01300                       ; DRAW DIAGONALS
01400                       ;
01500  003016  020777              LDXA 777                ;MOVE TO
01600  003020  060777              MVYA 777                ; UPPER RIGHT
01700  003022  023000              LDXA -1000              ;DRAW DIAGONAL
01800  003024  043000              DRYA -1000              ; TO LOWER LEFT
01900  003026  050777              MVXA 777                ;MOVE TO LOWER RIGHT
02000  003030  023000              LDXA -1000              ;DRAW DIAGONAL
02100  003032  040777              DRYA 777                ; TO UPPER LEFT
02200  003034  050000              MVXA 0                  ;MOVE TO TOP CENTER
02300  003036  043000              DRYA -1000              ;DRAW STRAIGHT DOWN
02400  003040  020777              LDXA 777                ;MOVE TO FAR
02500  003042  060000              MVYA 0                  ; RIGHT CENTER
02600  003044  033000              DRXA -1000              ;DRAW HORIZONTAL TO LEFT
02700                       ;
02800                       ; DRAW INSIDE SQUARE
02900                       ;
03000  003046  063000              MVYA -1000              ;MOVE TO BOTTOM LEFT
03100  003050  030777              DRXA 777                ;DRAW STRAIGHT UP
03200  003052  040777              DRYA 777                ;DRAW TOP EDGE TO RIGHT
03300  003054  033000              DRXA -1000              ;DRAW RIGHT EDGE DOWN
03400  003056  043000              DRYA -1000              ;DRAW BOTTOM EDGE TO LEFT
03500                       ;
03600                       ; DRAW INNER SQUARE
03700                       ;
03800  003060  023076              LDXA -702               ;SET INSIDE POINT
03900  003062  063076              MVYA -702               ; AT LOWER LEFT
04000  003064  045604              DRYR 1604               ;DRAW STRAIGHT UP
04100  003066  035604              DRXR 1604               ;DRAW TOP EDGE TO RIGHT
04200  003070  046174              DRYR -1604              ;DRAW STRAIGHT DOWN
04300  003072  036174              DRXR -1604              ;DRAW BOTTOM EDGE TO LEFT
04400                       ;
04500                       ; SCRATCHPAD PROMPTER #1
04600                       ;
04700  003074  013007              LDDZ <CRT1>             ;SELECT DISPLAY #1
04800  003076  023200              LDXA -600               ;POSITION INSIDE INNER
04900  003100  063110              MVYA -670               ; SQUARE AT LOWER LEFT
05000  003102  147311              TXT I,N                 ;INSERT 'INPUT:'
05100  003104  152720              TXT P,U
05200  003106  135324              TXT T.:
```

Table 6-2. Sample Refresh File No. 2 (Cont)

```
05300                   ;
05400                   ; SCRATCHPAD FOR DISPLAY #1
05500                   ;
05600  003110           SCRPD1:
05700  003110  120240          TXT < >,< >          ;TEN
05800  003112  120240          TXT < >,< >          ; SPACES
05900  003114  120240          TXT < >,< >          ;  FOR
06000  003116  120240          TXT < >,< >          ;   SCRATCHPAD
06100  003120  120240          TXT < >,< >          ;    AREA
06200                   ;
06300                   ; MOVING CIRCLE #1
06400                   ;
06500  003122           MCRCL1:
06600  003122  020000          LDXA 0               ;PED CONTROLLED
06700  003124  060000          MVYA 0               ; POSITION OF
06800  003126  073020          LDKX 3,20            ;  SMALL
06900  003130  077020          DRKY 3,20            ;   CIRCLE
07000                   ;
07100                   ; SCRATCHPAD PROMPTER #2
07200                   ;
07300  003132  012407          LDDZ <CRT2>          ;SELECT DISPLAY #2
07400  003134  023200          LDXA -600            ;POSITION INSIDE INNER
07500  003136  063110          MVYA -670            ; SQUARE AT LOWER LEFT
07600  003140  147311          TXT I,N              ;INSERT 'INPUT:'
07700  003142  152720          TXT P,U
07800  003144  135324          TXT T,:
07900                   ;
08000                   ; SCRATCHPAD FOR DISPLAY #2
08100                   ;
08200  003146           SCRPD2:
08300  003146  120240          TXT < >,< >          ;TEN
08400  003150  120240          TXT < >,< >          ; SPACES
08500  003152  120240          TXT < >,< >          ;  FOR
08600  003154  120240          TXT < >,< >          ;   SCRATCHPAD
08700  003156  120240          TXT < >,< >          ;    AREA
08800                   ;
08900                   ; PED CONTROLLED MOVING CIRCLE #2
09000                   ;
09100  003160           MCRCL2:
09200  003160  020000          LDXA 0               ;PED CONTROLLED
09300  003162  060000          MVYA 0               ; POSITION OF
09400  003164  073020          LDKX 3,20            ;  SMALL
09500  003166  077020          DRKY 3,20            ;   CIRCLE
09600  003170  001000          JUMP RLOOP           ;REPEAT
09700  003172  002004
```

It is also possible for a refresh file to contain entirely different images for each display monitor. Normally, such a file would contain a main program loop that calls two subroutines, each subroutine being the instructions associated with a particular display monitor. The following is an example of how such a refresh file might be structured:

```
MAIN:

     WATE                    ;SWAP PIXEL MEMORIES

     CALL RFRS1              ;DRAW IMAGE ON DISPLAY MONITOR NO. 1

     CALL RFRS2              ;DRAW IMAGE ON DISPLAY MONITOR NO. 2

     JUMP MAIN               ;LOOP BACK TO BEGINNING

RFRS1:

     LDDZ <CRT1,.....ARGn>   ;SELECT DISPLAY MONITOR NO. 1

     LDDP <ARG1,.....ARGn>   ;  AND ESTABLISH

     LDTI nn                 ;   DESIRED PARAMETERS

       .                     ;INSTRUCTIONS

       .                     ;  FOR

       .                     ;    DRAWING

       .                     ;      IMAGE ON

       .                     ;        DISPLAY

       .                     ;          MONITOR NO. 1

     RTRN                    ;RETURN TO MAIN LOOP
```

```
RFRS2:

     LDDZ <CRT2,.....ARGn>     ;SELECT DISPLAY MONITOR NO. 2
     LDDP <ARG1,.....ARGn>     ;  AND ESTABLISH
     LDTI nn                   ;   DESIRED PARAMETERS
       .                       ;INSTRUCTIONS
       .                       ;  FOR
       .                       ;    DRAWING
       .                       ;      IMAGE ON
       .                       ;        DISPLAY
       .                       ;          MONITOR 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 each display monitor. (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 monitor 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 0. 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 monitor).

In the example, a hardcopy of display monitor no. 1 could be generated as follows:

1. Send an SU message to modify the first LDDZ in RFRS2 to select display monitor no. 2.

2. Send an SU message to modify the first LDDZ in RFRS1 to select display monitor no. 1 and no. 4.

3. Press the hardcopy button on the hardcopy unit.

A hardcopy of display monitor no. 2 could be generated as follows:

1. Send an SU message to modify the first LDDZ in RFRS1 to select display monitor no. 1.

2. Send an SU message to modify the first LLDZ in RFRS2 to select display monitor no. 2 and no. 4.

3. Press the hardcopy button on the hardcopy unit.

SECTION 7

ADVANCED GRAPHIC CONTROL PROGRAM USAGE

7.1 INTRODUCTION

For certain applications, the GRAPHIC 8 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 digital graphic controller can operate independently, each executing a separate program located in the GRAPHIC 8 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 8 in the same manner that refresh files to be processed by the digital graphic controller are loaded (refer to the descriptions of the host-to-GRAPHIC 8 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 8. 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 (Sanders publication H-81-0022) 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 8 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-interruptible, 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 alpha-numeric (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 interruptible 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 8 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

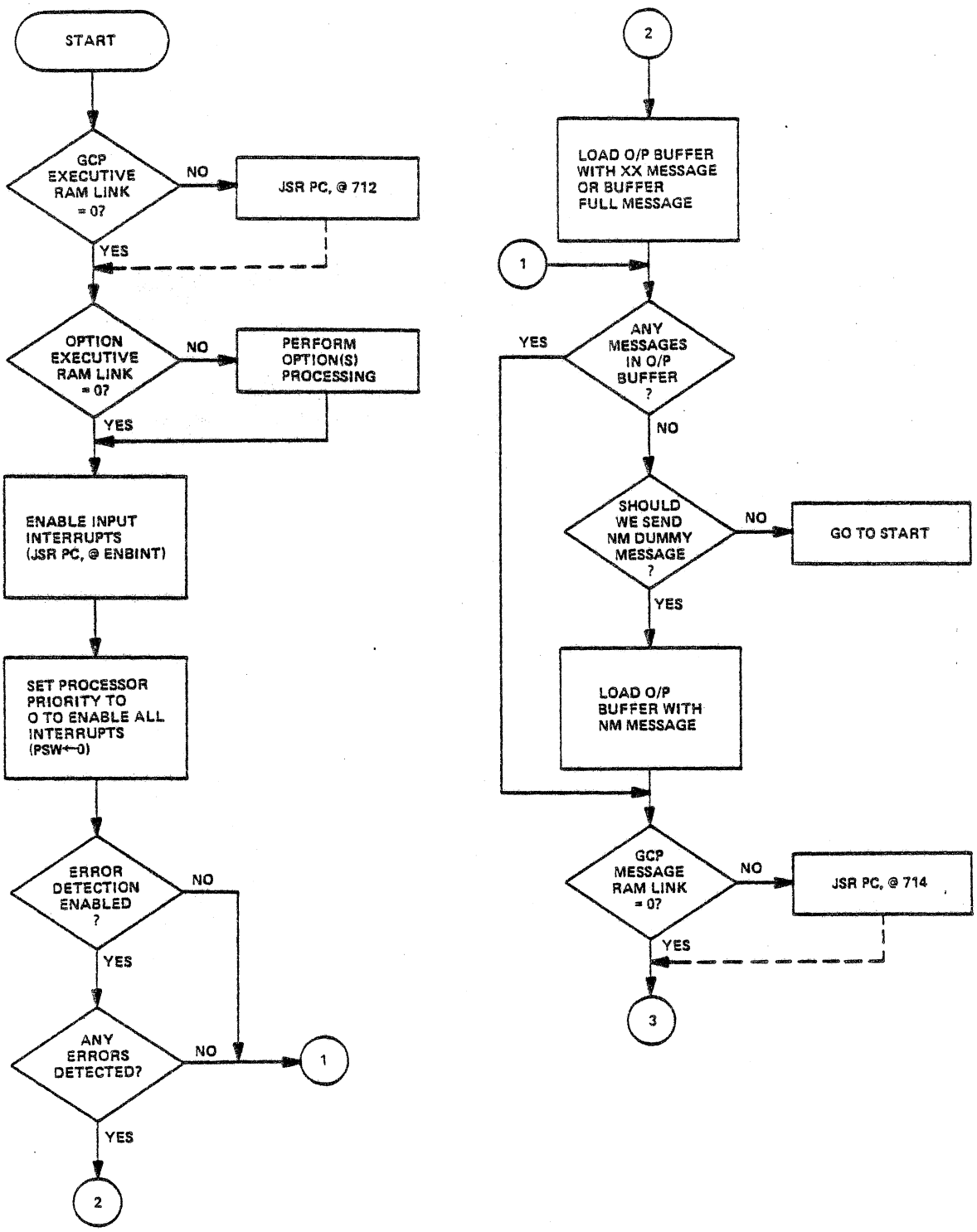


Figure 7-1. GCP Executive Loop Flowchart (Sheet 1 of 2)

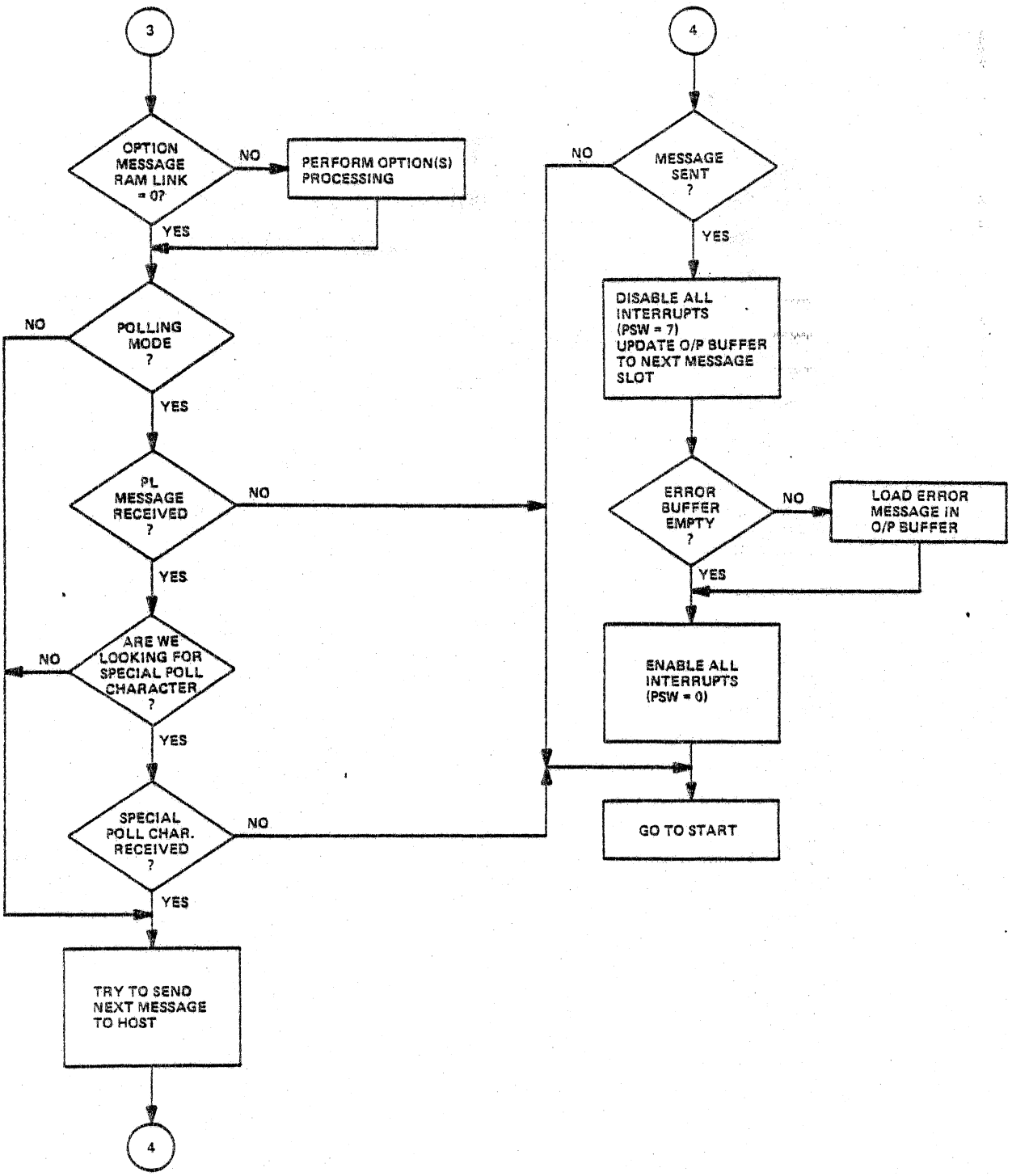


Figure 7-1. GCP Executive Loop Flowchart (Sheet 2 of 2)

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 8 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 digital 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.
- Parallel processing to be accomplished by the digital 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 digital graphic controller encounters a LINK instruction in the refresh file, the following operations occur:

a. The digital graphic controller fetches the address portion of the LINK instruction and then performs a jump and mark to that address.

b. The digital graphic controller stops fetching words from the refresh file, halts, and interrupts the display processor.

c. The contents of a few 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 digital 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 digital 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 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.

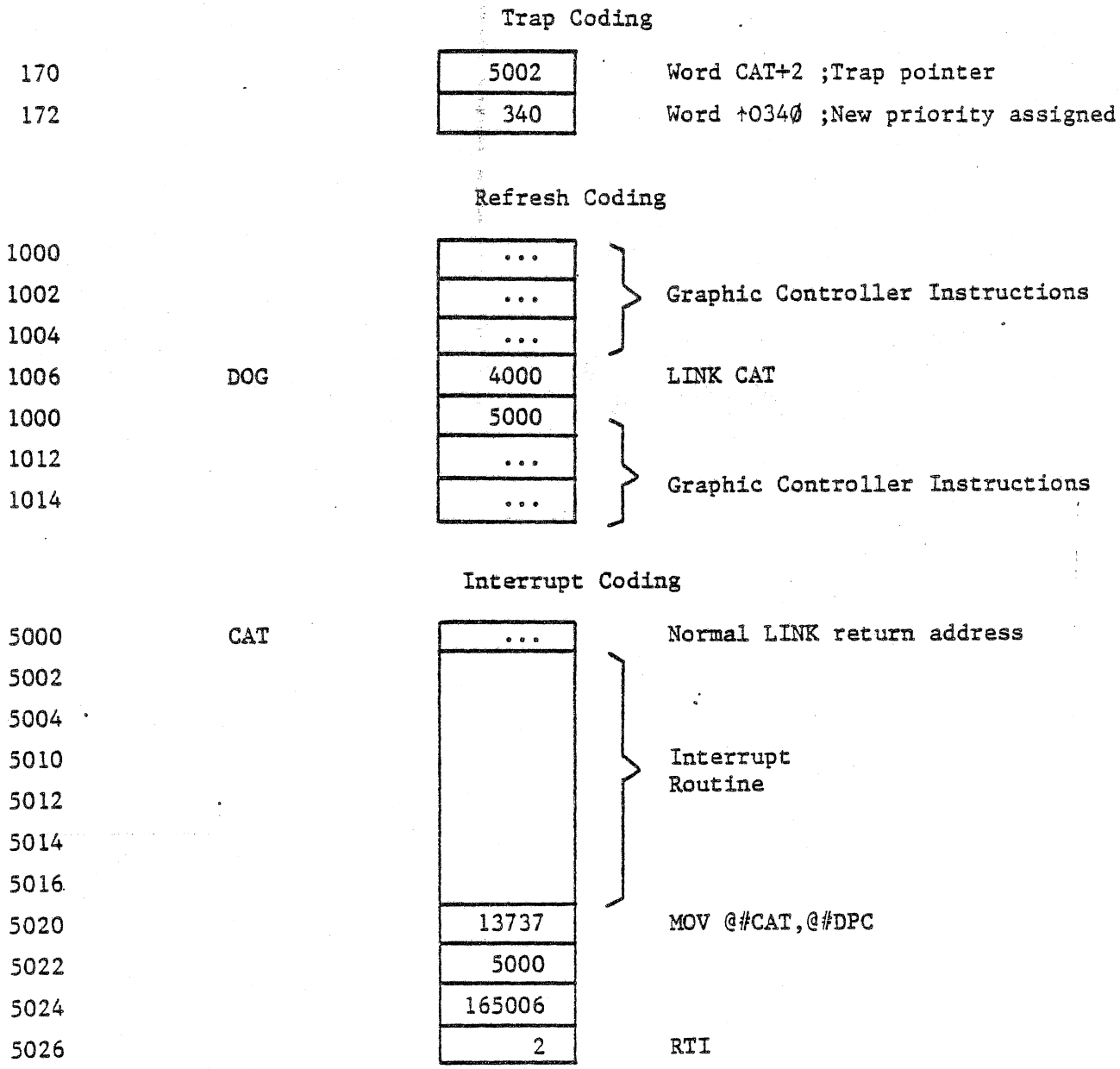


Figure 7-2. Synchronized Linkage Program Coding Example

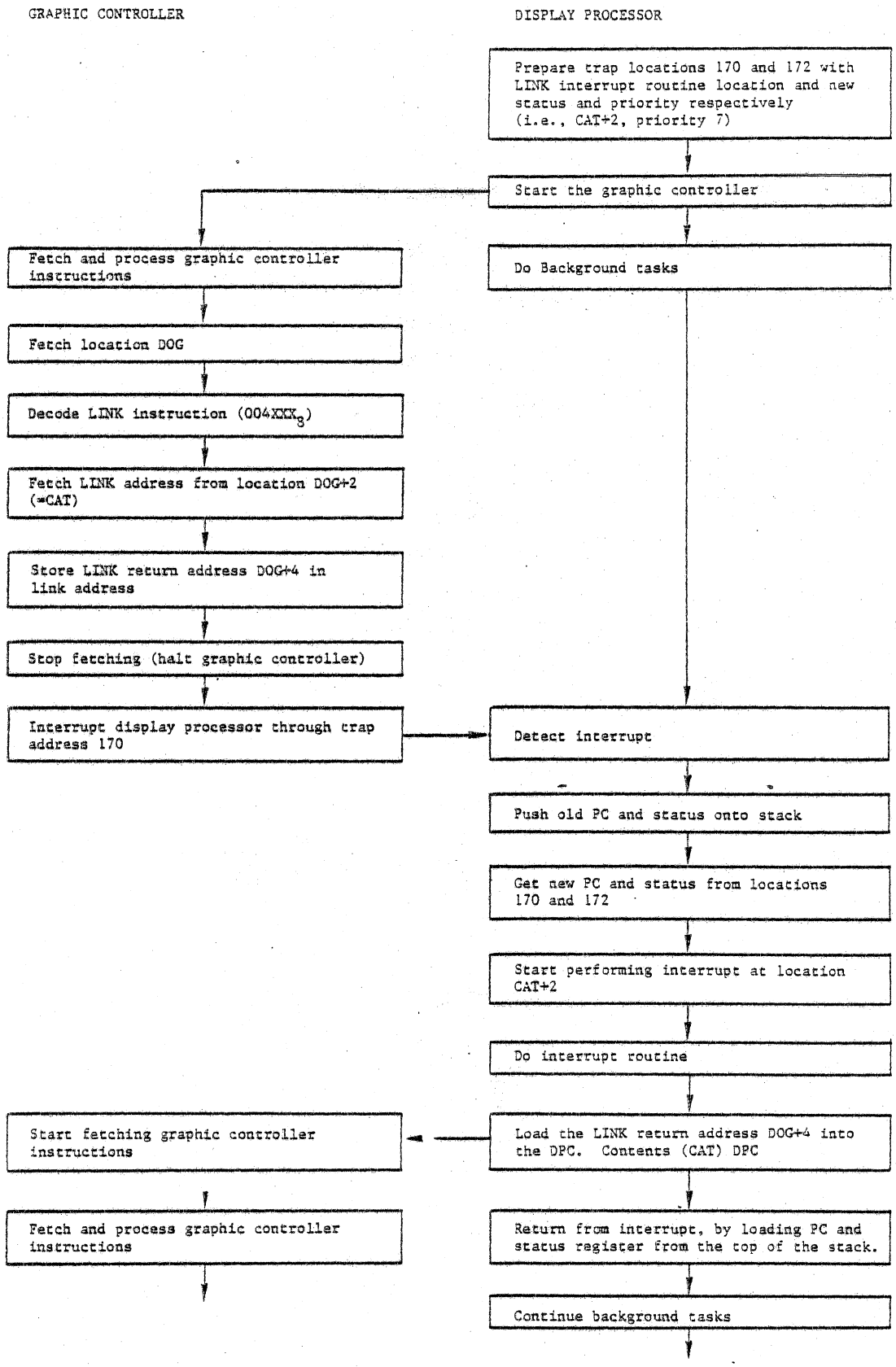


Figure 7-3. Synchronized Linkage Flow Chart Example

The result of using the sync link method is that display processor instructions can be placed directly in a refresh file. When the digital graphic controller encounters the LINK instruction, 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 digital 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 digital 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 location in the refresh file. This permits parallel processing by the display processor and the digital 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 digital graphic controller is required to be halted for a minimum amount of time. Therefore, the maximum data load that can be handled by the digital graphic controller is not reduced significantly. If data internal to the digital graphic controller is required by the display processor, it can be read before the digital graphic controller is restarted and then processed while the digital graphic controller is running.

The super sync method can be used to perform both simple and intricate functions.

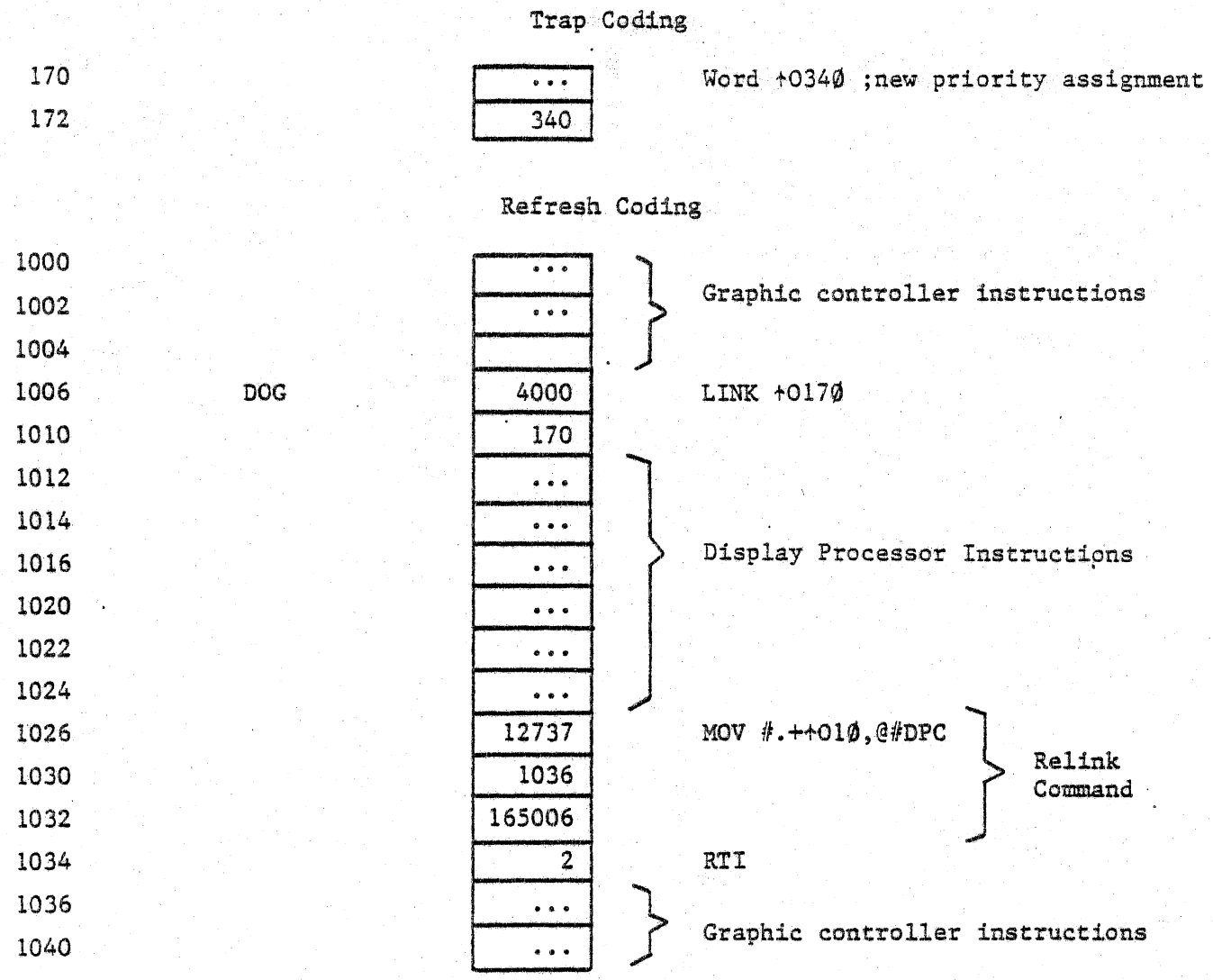


Figure 7-4. Sync Link Program Coding Example

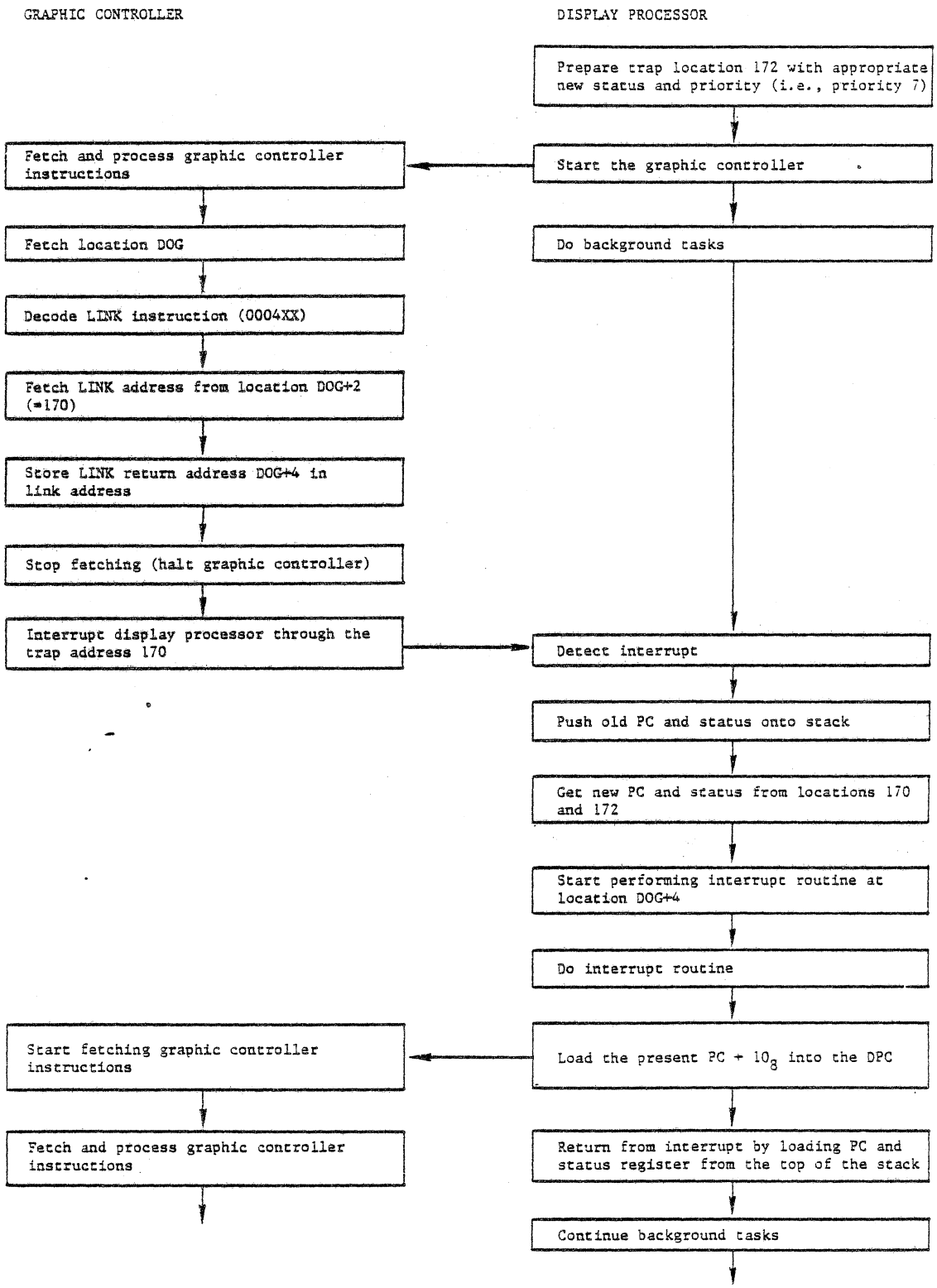


Figure 7-5. Sync Link Flow Chart Example

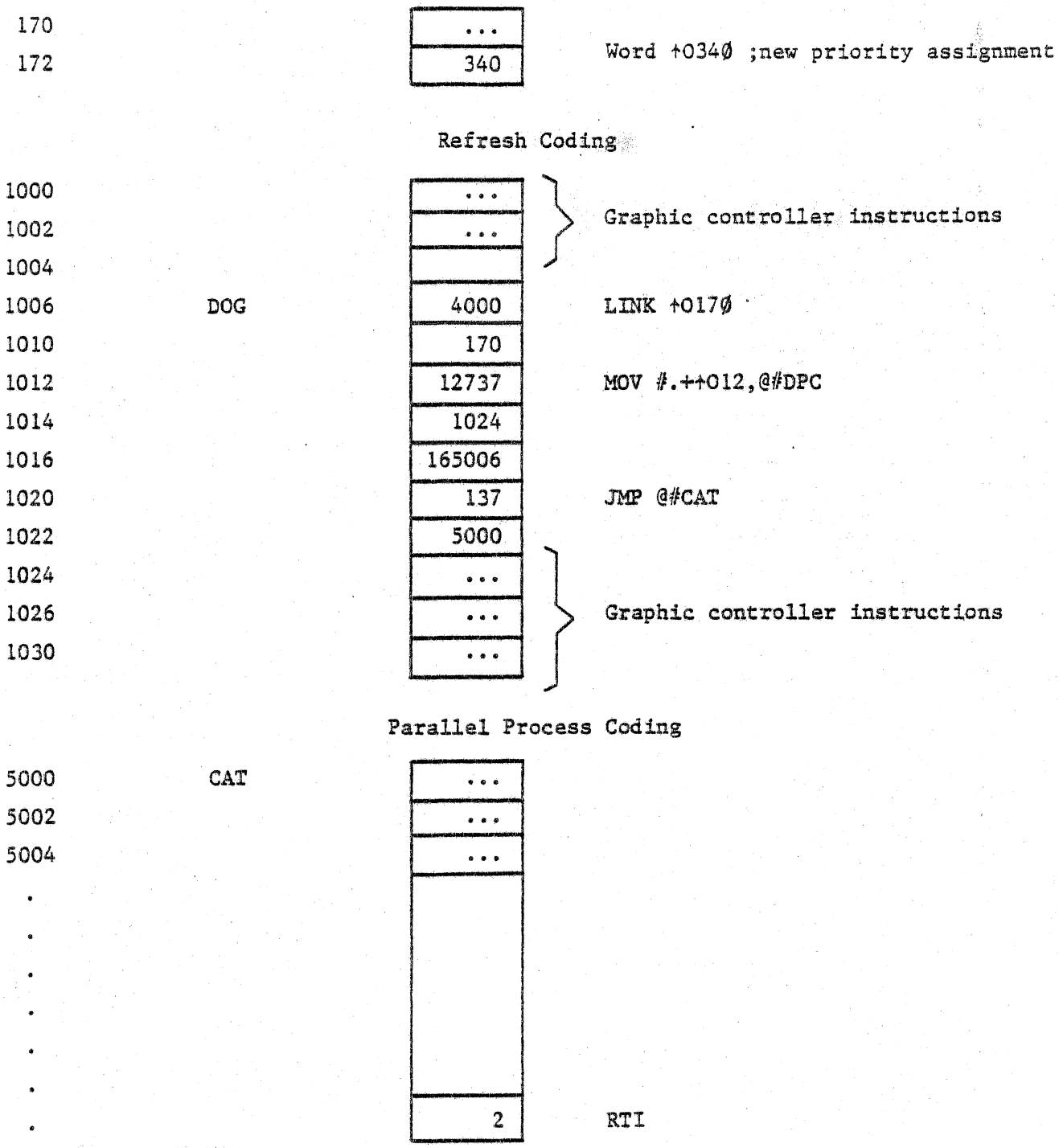


Figure 7-6. Super Sync Program Coding Example

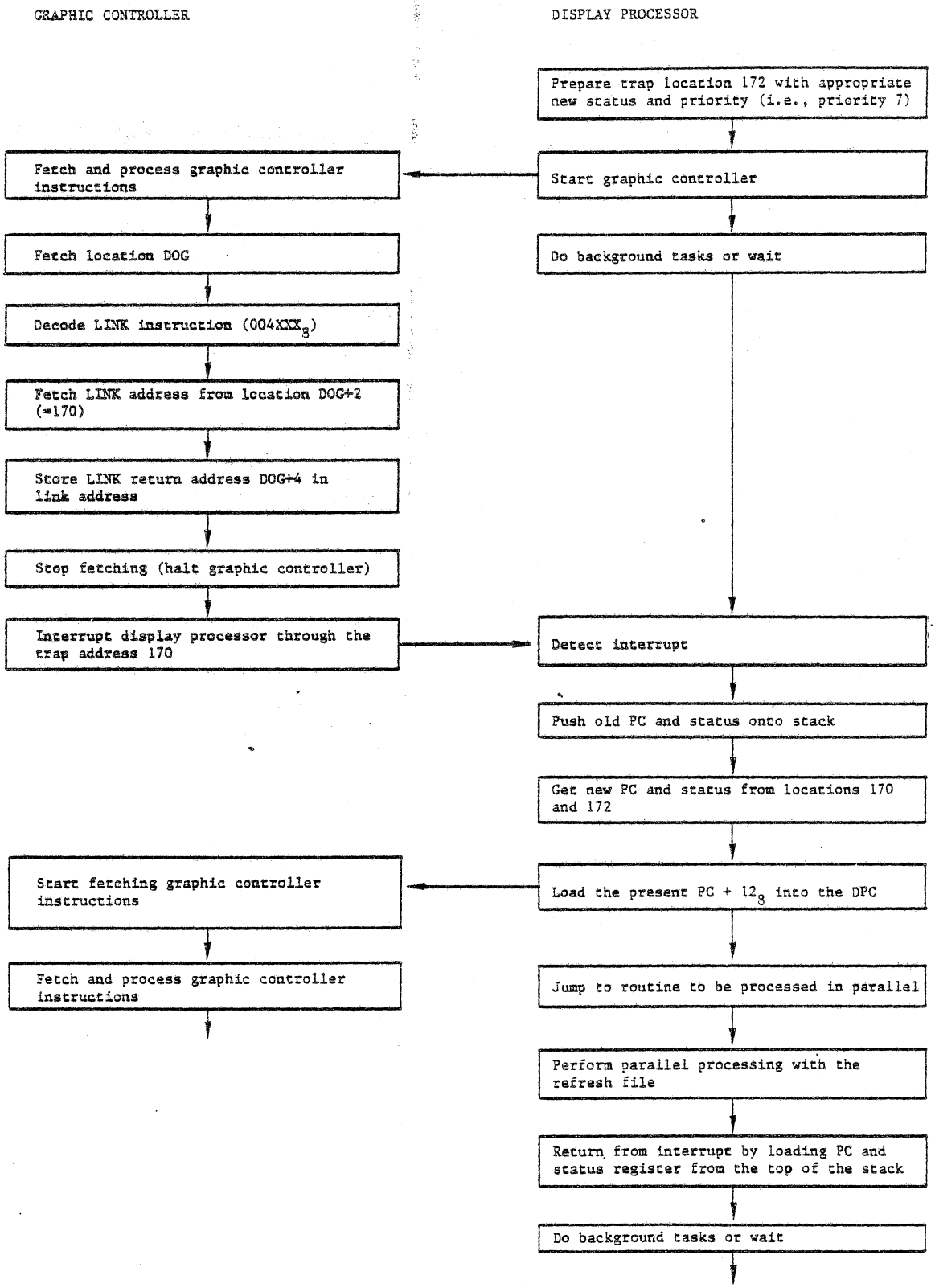


Figure 7-7. Super Sync Flow Chart Example

7.4 THE DIGITAL GRAPHIC CONTROLLER AS A DEVICE

Although the digital graphic controller operates independently of the display processor, it is under the control of the display processor at all times. The digital 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 digital 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 digital graphic controller is halted. With the digital 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).

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 8 is read from the output data register (ODR1) while data to be sent from the GRAPHIC 8 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,R0       ;MOVE DATA WORD INTO R0
        BIS   #40,@#STR1      ;SET 'OUTPUT WORD RECEIVED' BIT IN STR1
ACKNLG: BIT   #40,@#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   R0,@#IDR1       ;MOVE DATA WORD INTO IDR1
        BIS   #20000,@#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   #10,@#STR1      ;ENABLE DMA OUTPUT MODE
        MOV   #DATABF,@#MAR1  ;SET MAR1 TO INTERNAL BUFFER ADDR
        MOV   #-100.,@#WCR1   ;START DMA TRANSFER OF 100 WORDS
COMPL:  BIT   #20,@#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   #10,@#STR1      ;ENABLE DMA INPUT MODE
        MOV   #DATABF,@#MAR1  ;SET MAR1 TO INTERNAL BUFFER ADDR
        MOV   #-100.,@#WCR1   ;START DMA TRANSFER OF 100 WORDS
WAIT:   BIT   #20,@#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 8 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 three 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 8 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 and PED's) 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 'RECEIVE DONE' BIT OF RSR1
       BPL   WAIT            ;  (DONE INDICATES CHARACTER RECEIVED
                             ;   FROM HOST)
       MOV   @RDB1, R0       ;PUT CHAR IN LOW ORDER BYTE OF R0
```

b. To send data to the host computer:

```
       MOV   R0,@#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. 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,R0       ;PUT CHAR IN LOW ORDER BYTE OF R0
       BIC   #177600,R0      ;CLEAR R0 EXCEPT FOR 7-BIT ASCII CHAR CODE

       BIT   #100,R0         ;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,R0       ;SET UP KEYBOARD
        JSR   PC, OUT       ;  TO ACCEPT LAMP DATA
        MOV   #12,R0        ;LIGHT FUNCTION
        JSR   PC,OUT        ;  KEYS 1 AND 3
        MOV   #4,R0         ;LIGHT FUNCTION
        JSR   PC,OUT        ;  KEY 10
        MOV   #40,R0        ;LIGHT MATRIX
        JSR   PC,OUT        ;  KEY 5
        MOV   #60,R0        ;LIGHT MATRIX
        JSR   PC,OUT        ;  KEYS 12 AND 13
         .
         .
         .
OUT:    MOV   R0,@#TDB3     ;MOVE BYTE FROM R0 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):

7	6	5 4 3 2 1 0	
0	±	X COORD DATA	X coordinate displacement byte
1	±	Y COORD DATA	Y coordinate displacement byte

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.

7.7 PROGRAMMING EXAMPLES

7.7.1 PROGRAMMING THE COLOR DISPLAY MONITOR. Three primary colors are available for the color display monitor. They are Red (R), Green (G), and Blue (B). The displays are selected by the LDDZ instruction. The color is specified by the use of the Load Pixel Data Register (LDPD) instruction or the LDDZ instructions (for configurations with 3 bits per pixel plus blink).

The value in the Pixel Data Register (without the MSB if blink on) is the index into the lookup table (LUT). Each 8-bit entry in the lookup table has a value that represents the actual color seen on the screen (written to pixel memory). The format of the entry is:

7	6	5	4	3	2	1	0
B1	B0	G2	G1	G0	R2	R1	R0

where:

Rn is the nth intensity bit of Red,

Gn is the nth intensity bit of Green,

Bn is the nth intensity bit of Blue.

For a 3 plus blink (*) configurations any 8 colors can be selected from a choice of 256 colors by the use of the Lookup Table and the Modify Lookup Table (MDLU) instruction. For example, the following LUT values will give a reasonable set of colors.

LUT BYTE INDEX	LUT VALUE (OCTAL)	COLOR
0	0	BLACK
1	7	RED
2	70	GREEN
3	77	YELLOW
4	300	BLUE
5	307	MEGENTA
6	370	CYAN
7	377	WHITE

7.7.2 USE OF BLINK AND LUT. The size (8 bit bytes) of the mapped portion of the Lookup Table (LUT) is the number of simultaneous intensities or colors available for a given configuration. (See Table 7-1.) The blink bit acts on the data independent of the contents of the LUT.

Table 7-1. Mapped LUT Size

BITS/ PIXEL	SIMULTANEOUS INTENSITIES OR COLORS	LUT BYTE SIZE
2	2 plus blink	2
2	4	4
4	8 plus blink	8
4	16	16
8	128 plus blink	128
8	256	256

For example, a blinking blue vector on the screen with a configuration of 4 bits/pixel with blink as the MSB could be produced by the following refresh.

```
REFRESH:  INIT
          LDDP . .
          LDDZ . .
          MDLU   ADR, 10,1,400  ; where ADR: BYTE 0,0,0,0,300,0,0,0

LOOP:     WATE
           :
          LDPD    14        ; blink + intensity = 14
          LDXA     0        ; center of screen
          MVYA     0
          DRYA   200        ; draw vector
          JUMP   LOOP
```

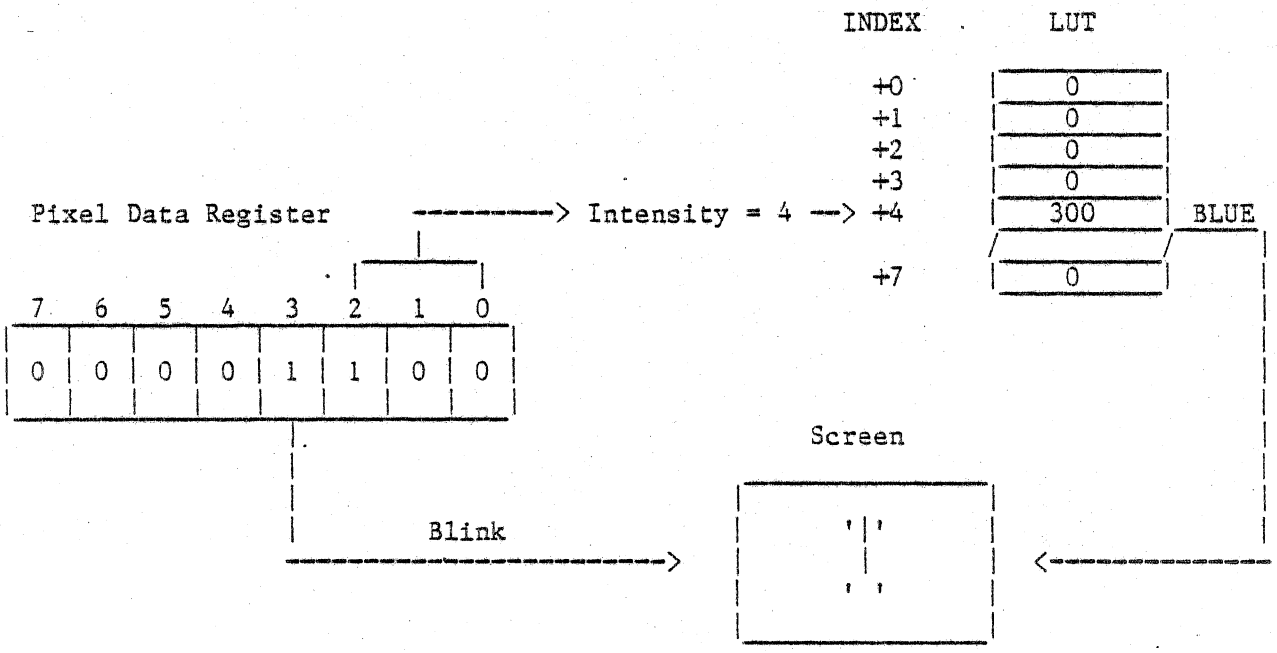
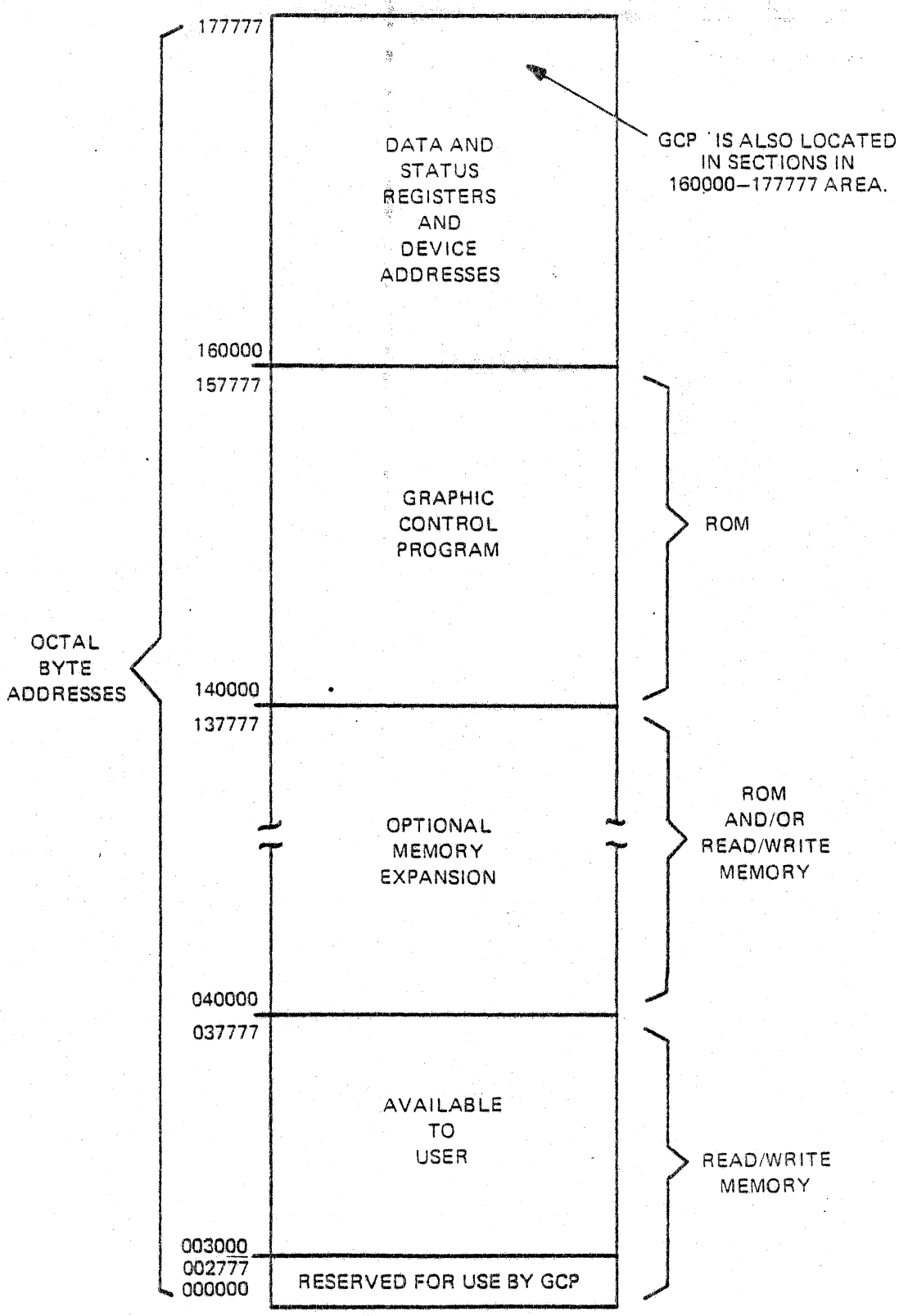


Figure 7-8. Relationship between PDR and LUT

To stop the vector from blinking, remove the blink bit from the LDPD instruction and execute the refresh file.

APPENDIX A

SUMMARY INFORMATION



H-79-0348-1

Figure A-1. GRAPHIC 8 System Memory Map

Table A-1. GRAPHIC 8 Local Mode Command Summary

KEYBOARD ENTRY	OPERATION
RETURN	Execute local mode command or returns system to local monitor level.
nnnnnn/	Display contents of memory address nnnnnn (octal).
/	Increment memory address counter by two and displays address contents.
or	Decrement memory address counter by two and displays address contents.
Bn	Select memory bank. (B0 0-32K; B1 32-64K; B2 64-96K; B3 96-128K; and B4 16-32K RAM).
S	Transfer GRAPHIC 8 to system mode operation.
T RETURN	Transfer to the verification test pattern.
L RETURN	Load memory from paper tape reader.
nnnnL RETURN	Load selected option from expansion module.
U RETURN	Unload all options.
O RETURN	Display status of all options loaded.
Q	Decrement contents of display processor Q register by two and displays result. Used with diagnostics to indicate address at which display processor halted.
nnnnnnD RETURN	Direct graphic controller to display refresh file beginning at address nnnnnn (octal).
nnnnnnG RETURN	Transfer control of display processor to program beginning at memory address nnnnnn (octal).
Y RETURN or P RETURN	Call teletypewriter emulation program. After entering emulation program, function key F0 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 8 to system operating mode. (Y = serial, P = parallel)
RUB OUT	Delete last octal entry from keyboard.

Table A-2. GRAPHIC 8 Controller Instruction Summary

NUMERICAL LIST

Legend:

Opcode word

The opcode word contains bits in one of three categories:

1. Numeric represents actual opcode bits. Note that opcode bits are not necessarily adjacent.

2. A thru G represent variable bits 0 (either binary zero or one) that effect the function of the instruction.

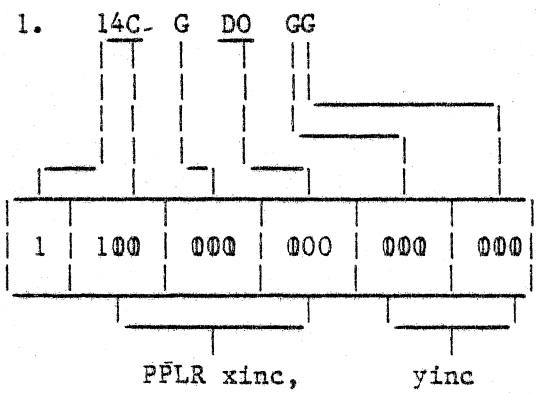
A = 000	E = 000
B = 000	F = 000
C = 000	G = 000
D = 000	

3. X means any bits left (up to 3) are ignored by the instruction. Underline (_) means a combination of variable and left over bits that fit into three bits as:

D2A = 010 binary
XC = X00 binary

For example:

1.



2.

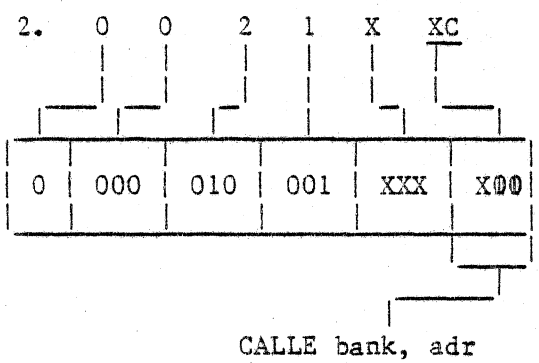


Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

MNEMONICS

adr = 16 bit address

bank = 2 bit bank 0-3 (if absent, bank 0 is assumed)

vcnum = video controller number 1-4

ix = initial x value

iy = initial y value

fx = final x value

fy = final y value

D = adr is displacement (if absent, adr is absolute address)

DR # = display register number

I = indirect address bit

F = final bit 15 of data word

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

NUMERICAL LIST

OPCODE	MNEMONICS	PAGE NUMBER
0000XX	HREF	3-20
0010XX	JUMP adr,I (I for mode 0 only)	3-12
*0010XXC	JUMPE bank, adr	3-12
0011XX	JRMP inc	3-13
*0011XX	JRMPE inc	3-13
0012XX	JMPZ adr,I (I for mode 0 only)	3-14
*0012XXC	JMPZE bank, adr	3-14
0013XX	JPRZ inc	3-15
*0013XX	JPRZE inc	3-15
0020XX	JMPM adr,I (I for mode 0 only)	3-18
*0020XX	JMPME adr	3-18
0021XX	CALL adr	3-15
*0021XXC	CALLE bank,adr	3-15
0022XX	CALR inc	3-16
*0022XX	CALRE inc	3-16
0023XX	RTRN	3-17
*0023XX	RTRNE	3-17
0030XX	IZPR (G7 only)	3-27
0040XX	LINK adr,I (I for mode 0 only)	3-19
*0040XX	LINKE adr	3-19
0041GG	SAVD DR#	3-24
*0041GG	SAVDE DR#	3-24

*Extended instruction only.

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

NUMERICAL LIST

OPCODE	MNEMONICS	PAGE NUMBER
0042GG	RESD DR#	3-24
*0042GG	RESDE DR#	3-24
0043GG	ADDI DR#,data	3-23
005000	NOOP	3-13
005GGG	JMPR inc	3-13
0060GG	LDRI dev,reg,data (G7 only)	3-23
0061GG	LDDI DR#,data	3-22
0062XX	LDSP adr	3-22
*0062XXC	LDSPE bank,adr	3-22
*0063XXC	MVPD |ABS| , |VERT| , |TOPM| , adr,D,bytes,ix,iy,fx,fy |REL| |HOR| |FROMPM|	3-29
0064GG *0065XCG	⟶ (See 3D instruction set in manual H-79-0350)	
0066XXC	MDLU bank,adr1,D,bytes,vcnum,adr2	3-31
*0067XX	CLRM	3-28
0070XX	WATE	3-21
*0071XX	INIT	3-28
0072GG	MODE mode	3-28
*0073XX	UPDT	3-21
*0075XXC	FLPG |ABS| , bank,adr,D,n |REL|	3-32

*Extended instruction only.

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

NUMERICAL LIST

OPCODE	MNEMONICS	PAGE NUMBER
*0076GG	PPTA ROT,TABINC	3-7
	COOR1 .... COORn,F	
*0077GG	PPTR ROT,TABINC	3-8
	INC1 .... INCn,F	
010CGGG	LDDZ zparm	3-26
014CGGG	LDDP dparm	3-25
020CGGG	LDXA xcoor	3-3
024CGGG	LDXR xinc	3-3
030CGGG	DRXA xcoor (mode 0 only)	3-5
034CGGG	DRXR xinc (mode 0 only)	3-5
*034XXCGG	LDPD data	3-28
040CGGG	DRYA ycoor	3-6
044CGGG	DRYR yinc	3-6
050CGGG	MVXA xcoor (mode 0 only)	3-4
*050CGGG	PPYA ycoor	3-7
054CGGG	MVXR xinc (mode 0 only)	3-4
*054CGGG	PPYR yinc	3-7
060CGGG	MVYA ycoor	3-4
064CGGG	MVYR yinc	3-4

*Extended instruction only.

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

NUMERICAL LIST

OPCODE	MNEMONICS	PAGE NUMBER
070CGGG	LDKX q , xlen	3-11
074CGGG	DRKY q , ylen	3-11
10CGDOGG	DRSR xinc, yinc	3-6
10CGD1GG	MVSR xinc, yinc	3-5
1GGD2AGG	TXT chr1,chr2	3-10
116AD2AGG	CHAR B,S,chr	3-9
14CGDOGG	PPLR xinc,yinc	3-6
14CGD1GG	LDTI linc,tinc	3-27

*Extended instruction only.

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
ADDI	0 0 0 0 1 0 0 0 1 1 DR# ± DATA	ADD TO DISPLAY REGISTER IMMEDIATE	3-23
CALL	0 0 0 0 0 1 0 0 0 1 X X X X X X SUBROUTINE ADDRESS	CALL SUBROUTINE	3-15
*CALLE	0 0 0 0 0 1 0 0 0 1 X X X X A17 A16 A15 . . SUBROUTINE ADDRESS . . A0	CALL EXTENDED SUBROUTINE	3-15
CALR(E)	0 0 0 0 0 1 0 0 1 0 X X X X X X SUBROUTINE INCREMENT (IN EVEN BYTES)	CALL RELATIVE	3-16
CHAR	1 0 0 1 1 1 B 1 1 CHARACTER ASCII CODE	DRAW SINGLE CHARACTER	3-9
*CLRM	0 0 0 0 1 1 0 1 1 1 X X X X X X	CLEAR MAPPING MEMORY	3-28
DRKY	0 1 1 1 1 QIV QII Y SEMI-AXIS LENGTH (OR CIRCLE RADIUS)	DRAW CONIC Y	3-11

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
DRSR	1 0 ± Y-INCREMENT 0 0 ± X-INCREMENT	DRAW SHORT RELATIVE	3-6
DRXA	0 0 1 1 0 ± X-COORDINATE	DRAW X ABSOLUTE	3-5
DRXR	0 0 1 1 1 ± X-INCREMENT	DRAW X RELATIVE	3-5
DRYA	0 1 0 0 0 ± Y-COORDINATE	DRAW Y ABSOLUTE	3-6
DRYR	0 1 0 0 1 ± Y-INCREMENT	DRAW Y RELATIVE	3-6
*FLPG	0 0 0 0 1 1 1 1 0 1 X X X X A17 A16 A15 . . ADDRESS OF LIST OF VERTICES . . A0 AM REL NUMBER OF VERTICES	FILL A CONVEX POLYGON	3-32
HREF	0 0 0 0 0 0 0 0 0 0 X X X X X X	HALT REFRESH	3-20
*INIT	0 0 0 0 1 1 1 0 0 1 X X X X X X	INITIALIZE	3-28

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
IZPR	0 0 0 0 0 1 1 0 0 0 X X X X X X	INITIALIZE	3-27
JMPM	0 0 0 0 0 1 0 0 0 0 X X X X X X I JUMP ADDRESS (IN EVEN BYTES)	JUMP AND MARK	3-18
JMPR	0 0 0 0 1 0 1 ± JUMP INCREMENT (IN EVEN BYTES)	JUMP SHORT RELATIVE	3-13
JMPZ	0 0 0 0 0 0 1 0 1 0 X X X X X X I JUMP ADDRESS	JUMP IF DISPLAY REGISTER 0 CONTENTS ≠ 0	3-14
*JMPZE	0 0 0 0 0 0 1 0 1 0 X X X X A17 A16 A15 A14 . . JUMP ADDRESS . . A0	JUMP EXTENDED IF DISPLAY REGISTER 0 CONTENTS ≠ 0	3-14
JPRZ	0 0 0 0 0 0 1 0 1 1 X X X X X X JUMP INCREMENT (IF EVEN BYTES)	JUMP RELATIVE IF DISPLAY REGISTER 0 CONTENTS ≠ 0	3-15

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
JRMP	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 1 X X X X X X JUMP INCREMENT (IN EVEN BYTES)	JUMP RELATIVE	3-13
JUMP	0 0 0 0 0 0 1 0 0 0 X X X X X X I JUMP ADDRESS	JUMP	3-12
*JUMPE	0 0 0 0 0 0 1 0 0 0 X X X X A17 A16 A15 A14 . . JUMP ADDRESS . . A0	JUMP EXTENDED ADDRESS	3-12
LDDI	0 0 0 0 1 1 0 0 0 1 DR# DATA	LOAD DISPLAY REGISTER IMMEDIATE *	3-22
LDDP	0 0 0 1 1 DISPLAY PARAMETERS	LOAD DISPLAY PARAMETER REGISTER	3-25
LDDZ	0 0 0 1 0 DISPLAY Z PARAMETERS	LOAD DISPLAY Z REGISTER	3-26

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
LDKX	0 1 1 1 0 QIII QI X SEMI-AXIS LENGTH	LOAD CONIC X REGISTER	3-11
*LDPD	0 0 1 1 1 X X X GRAY LEVEL	LOAD PIXEL DATA REGISTER	3-28
LDRI	0 0 0 0 1 1 0 0 0 0 DEV# REG# X X X X DATA	LOAD DEVICE REGISTER IMMEDIATE	3-23
LDSP	0 0 0 0 1 1 0 0 1 0 X X X X X X ADDRESS -	LOAD STACK POINTER	3-22

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
*LDSPE	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 0 0 1 1 0 0 1 0 X X X X A17 A16 A15 . . ADDRESS . . A0	LOAD EXTENDED STACK POINTER	3-22
LDTI	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 1 1 0 0 0 0 0 0 0 1 TEXT INCREMENT	LOAD TEXT INCREMENT REGISTER	3-27
LDXA	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 1 0 0 ± X-COORDINATE	LOAD X ABSOLUTE	3-3
LDXR	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 1 0 1 ± X-INCREMENT	LOAD X RELATIVE	3-3
LINK	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 A14 . LINK ADDRESS . . A0	SYNCHRONIZED LINKAGE	3-19
*LINKE	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 A15 . . LINK ADDRESS . . A0	SYNCHRONIZED LINKAGE EXTENDED	3-19

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
*MDLU	0 0 0 0 1 1 0 1 0 1 X X X X A17 A16	MODIFY LOOKUP TABLE	3-31
	A15 . . GRAPHIC 8 MEMORY ADDRESS . . A0		
	AM NUMBER OF GRAPHIC 8 8-BIT BYTES		
	0 0 0 0 0 0 0 0 VIDEO CONTROLLER(S)		
	0 0 0 0 0 0 0 1 LUT ADDRESS		
MODE	0 0 0 0 1 1 1 0 1 0 MODE	LOAD INSTRUCTION MODE REGISTER	3-28
*MVPD	0 0 0 0 1 1 0 0 1 1 X X X X A17 A16	MOVE PIXEL DATA	3-29
	A15 . . GRAPHIC 8 MEMORY ADDRESS . . A0		
	AM REL X Y INC DIR		
	INITIAL X VALUE		
	INITIAL Y VALUE		
	FINAL X VALUE		
	FINAL Y VALUE		
MVSR	1 0 ± Y-INCREMENT 0 1 ± X-INCREMENT	MOVE SHORT RELATIVE	3-5

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
MVXA	0 1 0 1 0 ± X-COORDINATE	MOVE X ABSOLUTE	3-4
MVXR	0 1 0 1 1 ± X-INCREMENT	MOVE X RELATIVE	3-4
MVYA	0 1 1 0 0 ± Y-COORDINATE	MOVE Y ABSOLUTE	3-4
MVYR	0 1 1 0 1 ± Y-INCREMENT	MOVE Y RELATIVE	3-4
NOOP	0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0	NO OPERATION	3-13
PPLR	1 1 ± Y-INCREMENT 0 0 ± X-INCREMENT	POINT PLOT RELATIVE	3-6
*PPTA	0 0 0 0 1 1 1 1 1 0 ROT TAB. INCR.	POINT PLOT TABULAR ABSOLUTE	3-7
	F X X X X ± COORDINATE		

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT	DESCRIPTION	PAGE NUMBER
*PPTR	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 0 0 0 0 1 1 1 1 1 1 ROT TAB. INCR. F X X X X ± INCREMENT	POINT PLOT TABULAR RELATIVE	3-8
*PPYA	0 1 0 1 0 ± Y-COORDINATE	POINT PLOT Y ABSOLUTE	3-7
*PPYR	0 1 0 1 1 ± Y-INCREMENT	POINT PLOT Y RELATIVE	3-7
RESD(E)	0 0 0 0 1 0 0 0 1 0 DR#	RESTORE DISPLAY REGISTER	3-24
RTRN	0 0 0 0 0 1 0 0 1 1 X X X X X X	RETURN	3-17
SAVD(E)	0 0 0 0 1 0 0 0 0 1 DR#	SAVE DISPLAY REGISTER	3-24
TXT	1 CHARACTER 2 ASCII CODE 1 CHARACTER 1 ASCII CODE	DRAW TWO TABULAR CHARACTERS	3-10

Table A-2. GRAPHIC 8 Controller Instruction Summary (Cont)

ALPHABETICAL LIST

MNEMONICS	FORMAT															DESCRIPTION	PAGE NUMBER
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
*UPDT	0	0	0	0	1	1	1	0	1	1	X	X	X	X	X	X	UPDATE VIDEO CONTROLLER REGISTERS	3-21
WATE	0	0	0	0	1	1	1	0	0	0	X	X	X	X	X	X	WAIT	3-21

Table A-3. Graphic Controller Register Format Summary

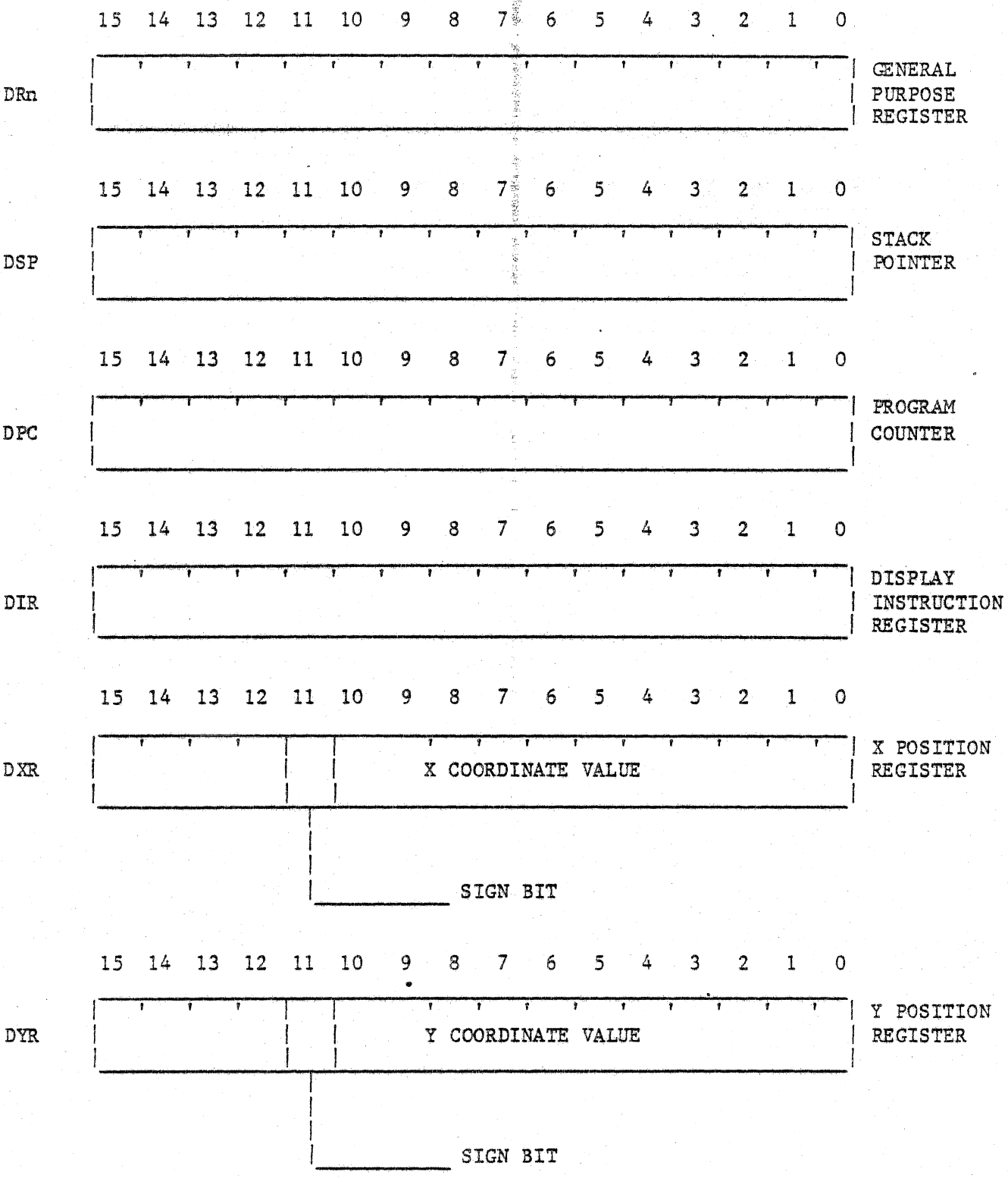


Table A-3. Graphic Controller Register Format Summary (Cont)

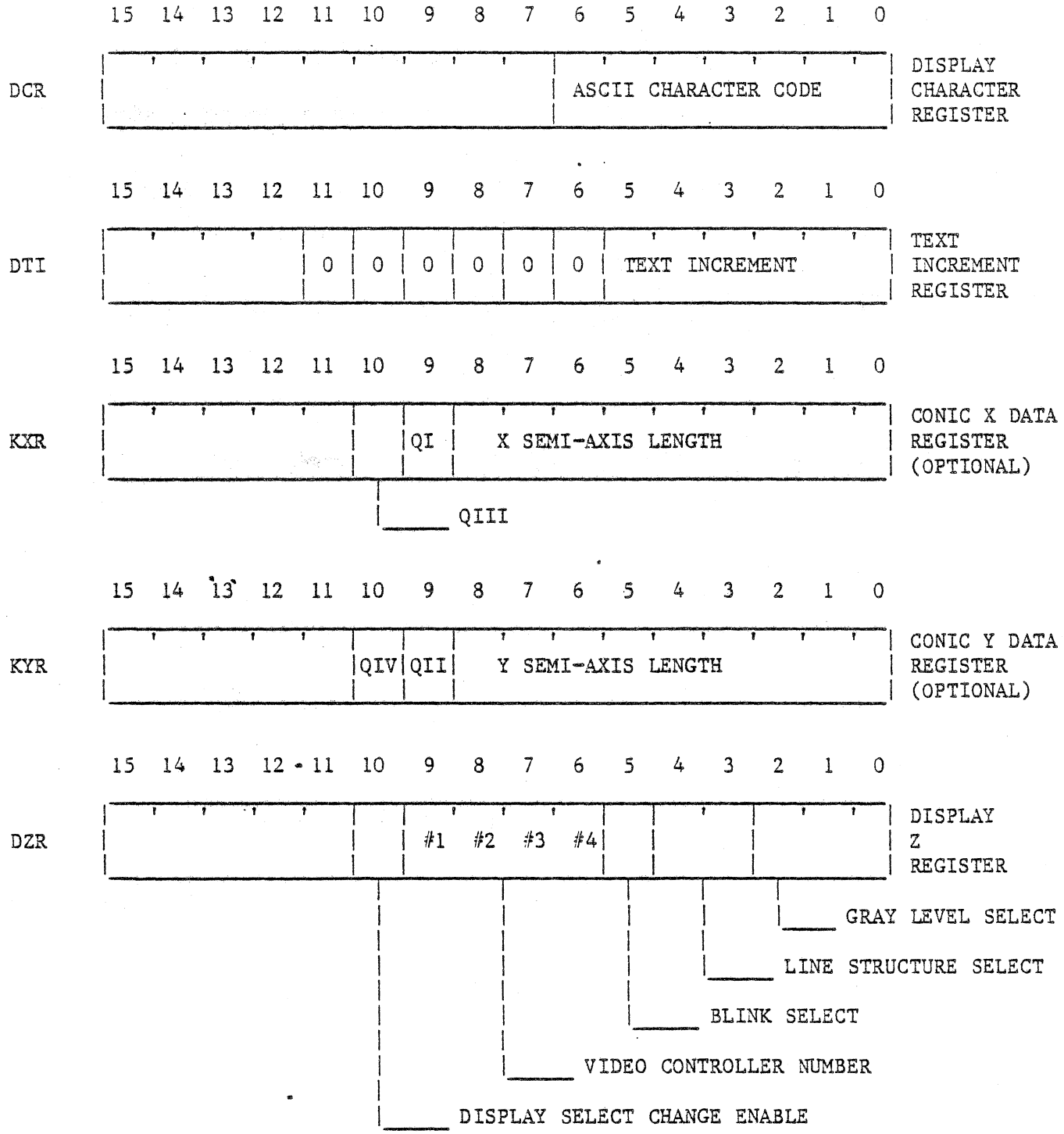


Table A-3. Graphic Controller Register Format Summary (Cont)

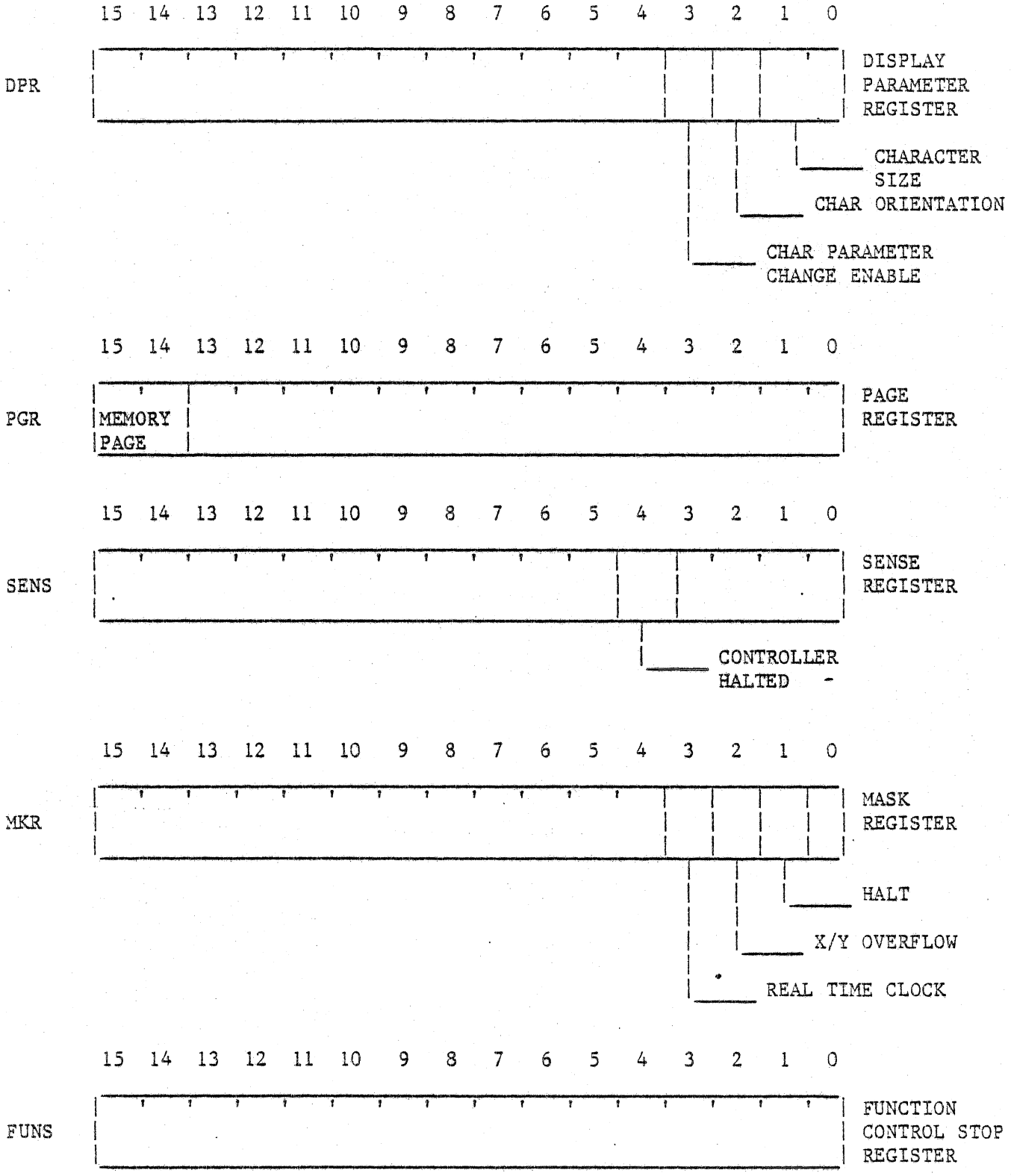


Table A-3. Graphic Controller Register Format Summary (Cont)

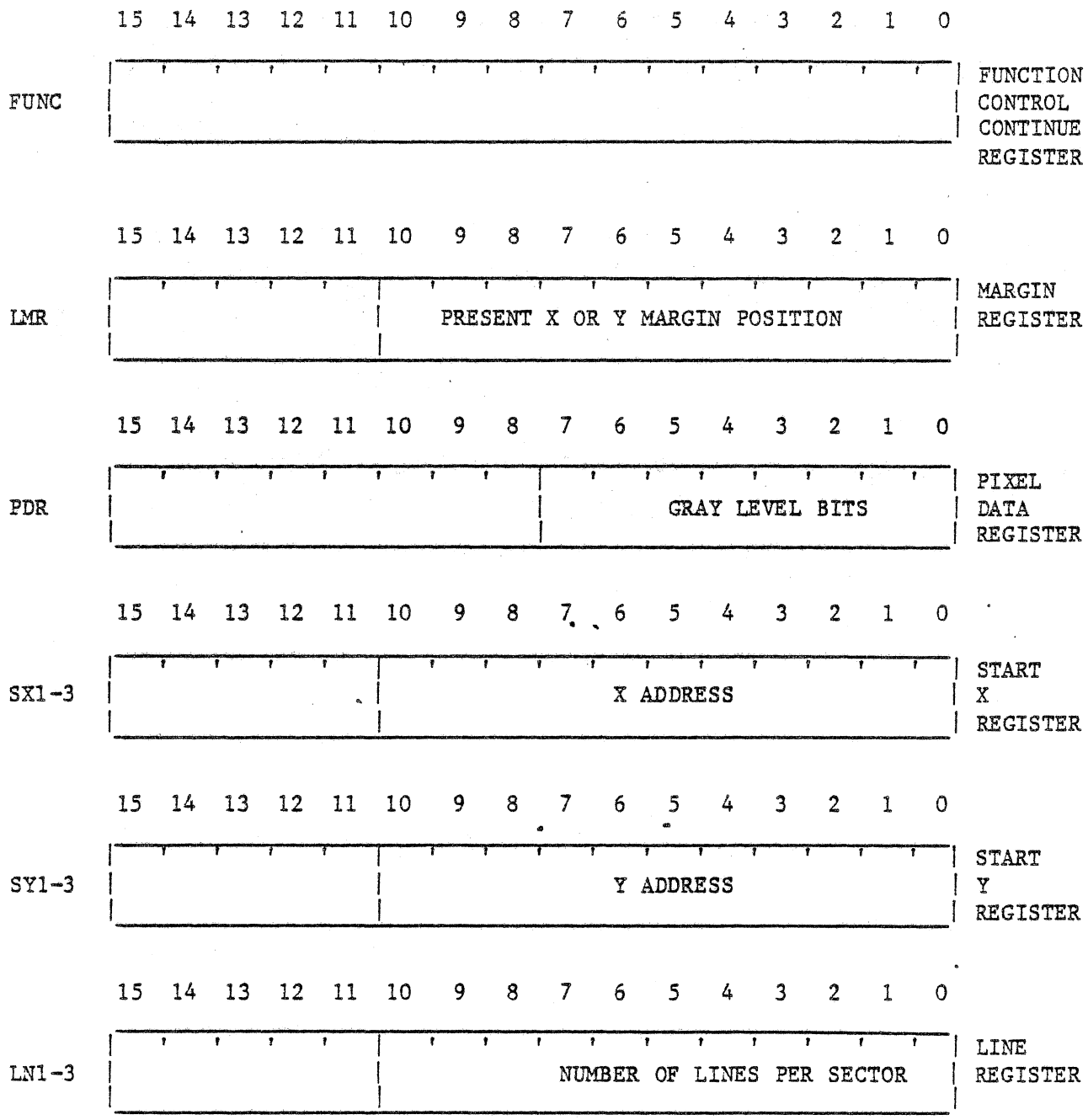


Table A-3. Graphic Controller Register Format Summary (Cont)

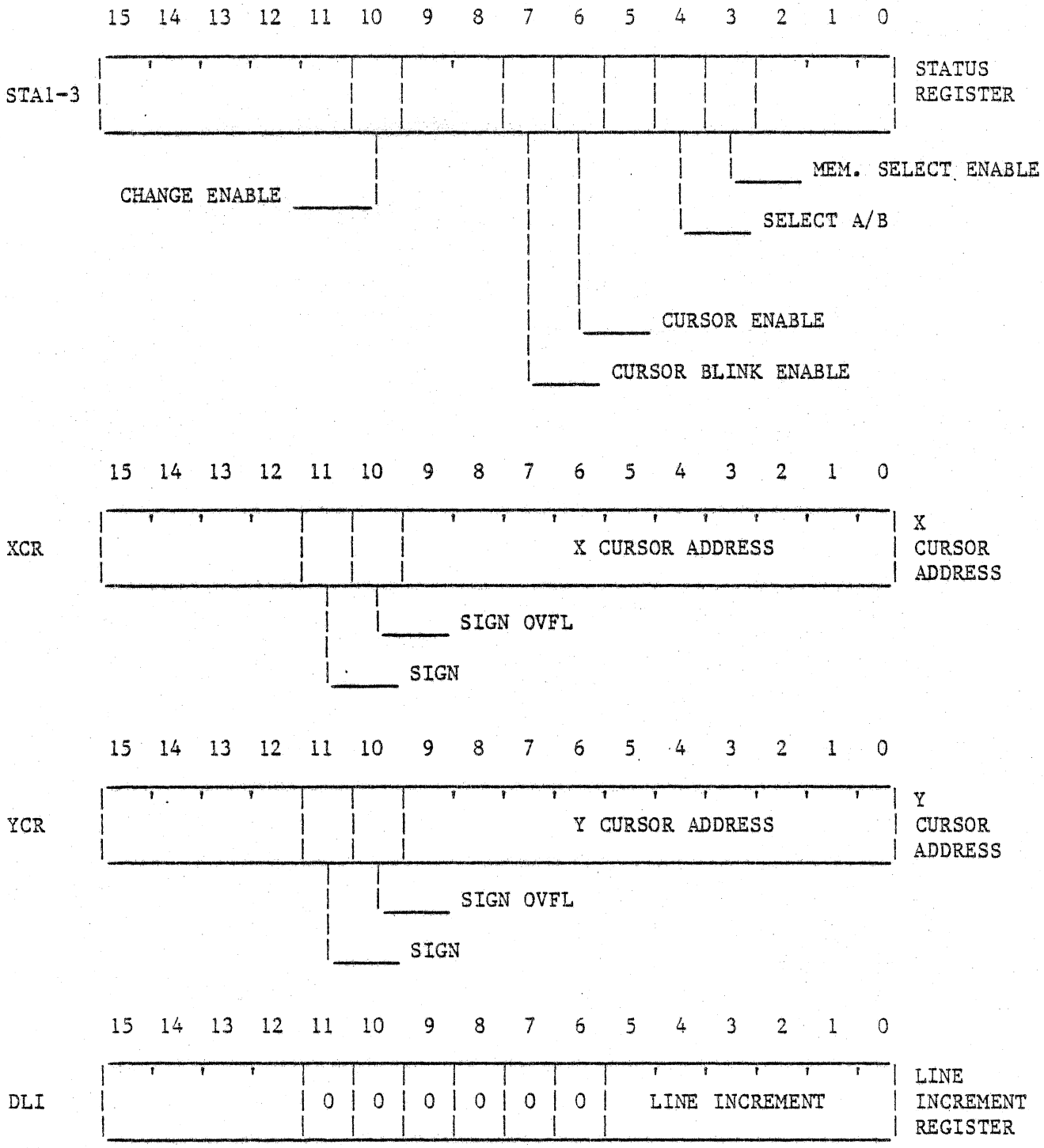


Table A-3. Graphic Controller Register Format Summary (Cont)

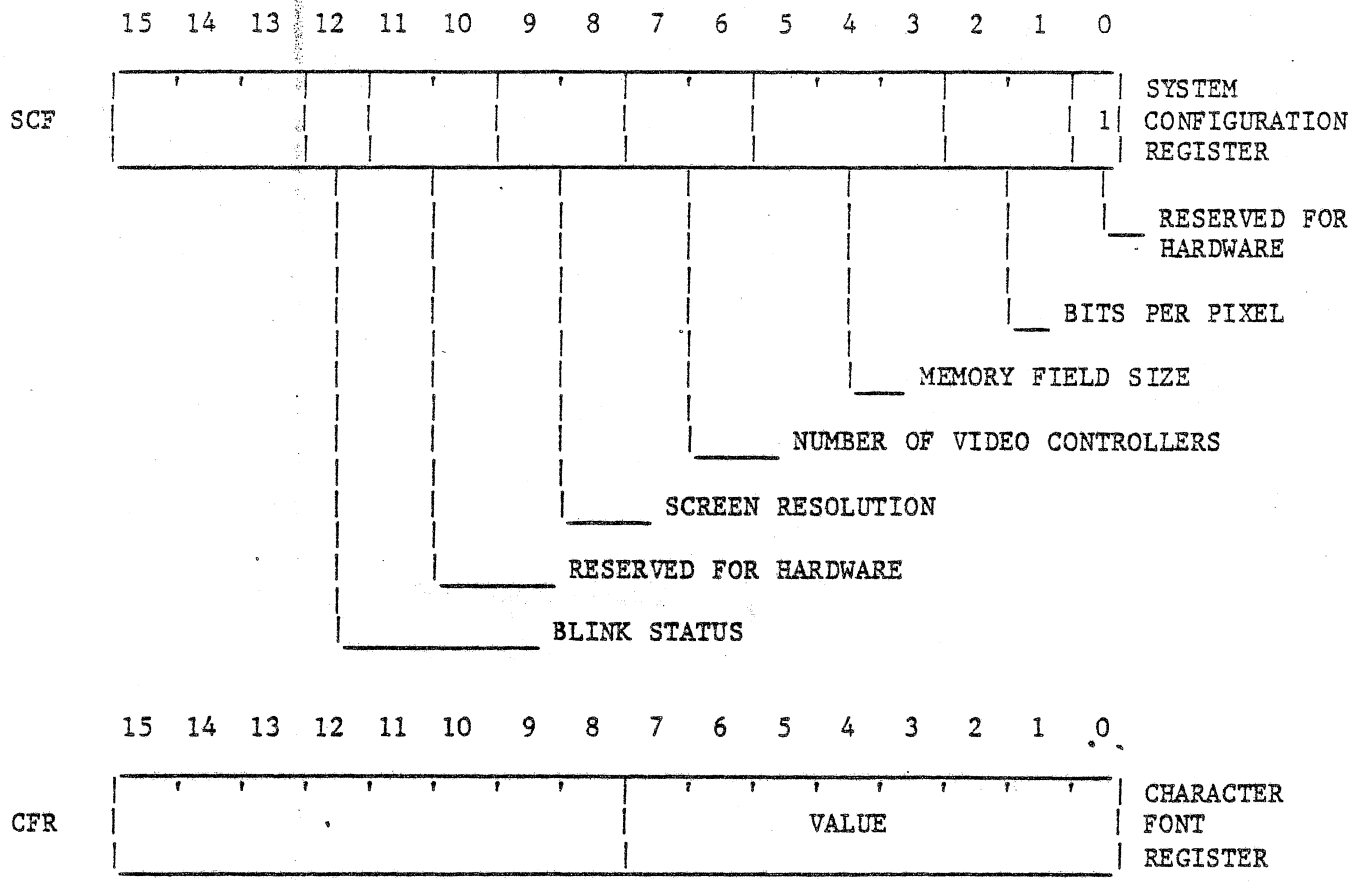
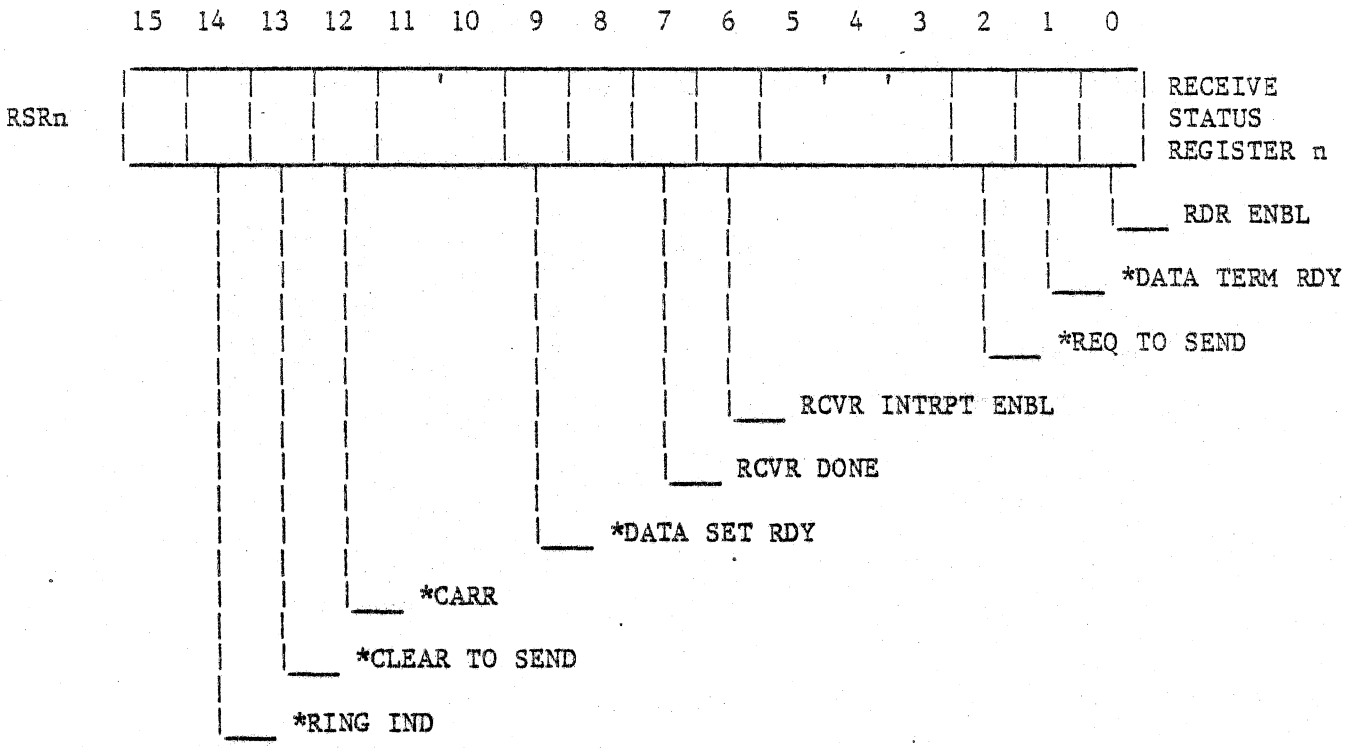
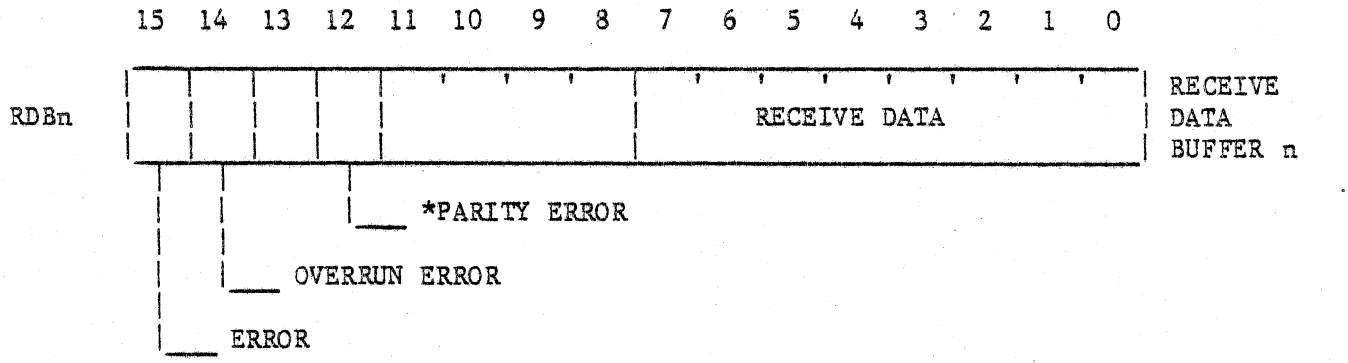


Table A-4. Serial Interface Register Format Summary



*Used on full RS-232C interface ports 1, 5 and 9



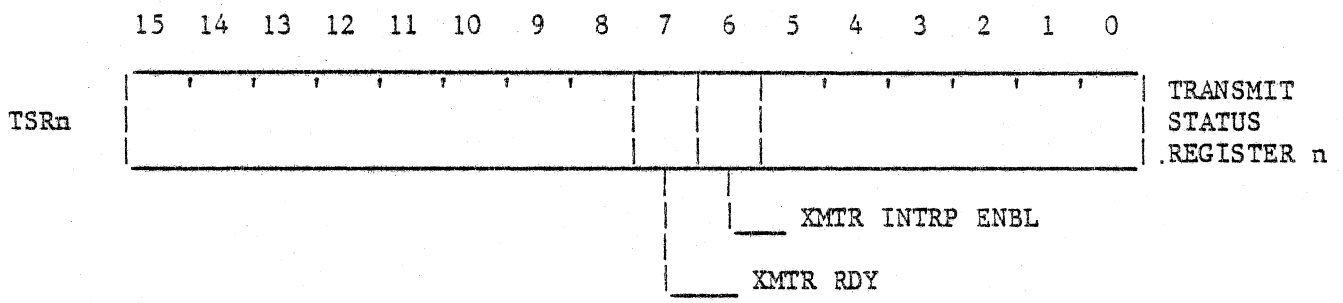


Table A-4. Serial Interface Register Format Summary (Cont)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TDBn									TRANSMIT DATA								TRANSMIT DATA BUFFER n

NOTE: Unidentified bits not used.

Table A-5. Parallel Interface Register Format Summary

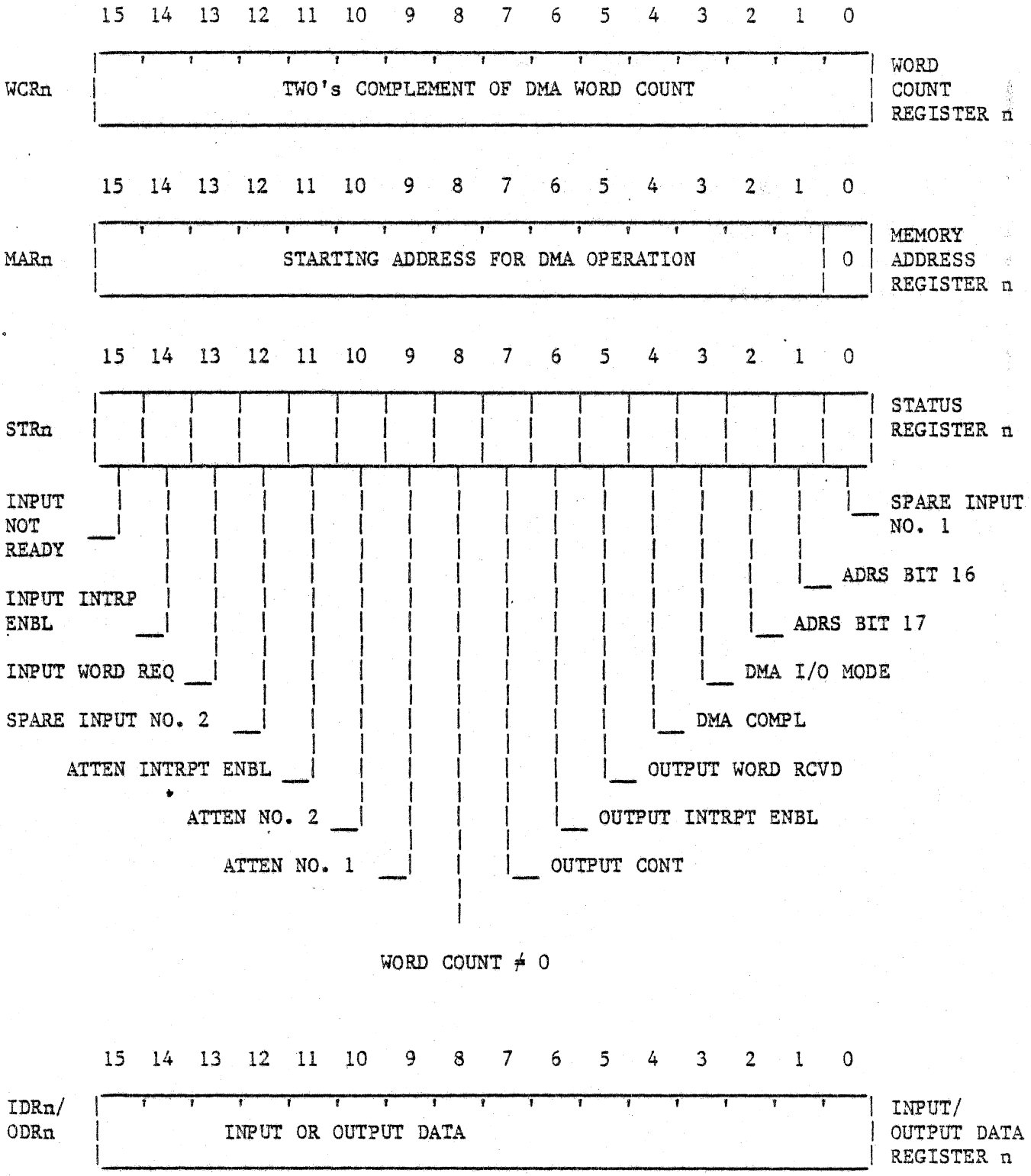


Table A-6. Register Designations and Address Assignments

DEVICE	REGISTER	MNEMONIC	ADDRESS
DISPLAY PROCESSOR	Processor Status Word	PSW	177776
	Reserved		177760 thru 177772
	8-bit switch register	SWT	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	WRC1	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
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 (Keyboard No. 3 or PED No. 6)		
	Receive Status Register 2	RSR2	176510
	Receive Data Buffer 2	RDB2	176512
	Transmit Status Register 2	TSR2	176514
	Transmit Data Buffer 2	TDB2	176516
	Port No. 3 (Keyboard No. 1 or PED No. 8)		
	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 (Keyboard No. 8 or PED No. 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 (unused)		
	Receive Status Register 5	RSR5	176540
	Receive Data Buffer 5	RDB5	176542
	Transmit Status Register 5	TSR5	176544
	Transmit Data Buffer 5	TDB5	176546

Table A-6. Register Designations and Address Assignments (Cont)

DEVICE	REGISTER	MNEMONIC	ADDRESS
MULTIPORT SERIAL INTERFACE CARD NO. 2 (OPTIONAL) (Cont)	Port No. 6 (Keyboard No. 6 or PED No. 3)		
	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 (Keyboard No. 2 or PED No. 7)		
	Receive Status Register 7	RSR7	176560
	Receive Data Buffer 7	RDB7	176562
	Transmit Status Register 7	TSR7	176564
	Transmit Data Buffer 7	TDB7	176566
	Port No. 8 (Keyboard No. 7 or PED No. 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
MULTIPORT SERIAL INTERFACE CARD NO. 3	Port No. 9 (Keyboard No. 4 or PED No. 5)		
	Receive Status Register 9	RSR9	176600
	Receive Data Buffer 9	RDB9	176602
	Transmit Status Register 9	TSR9	176604
	Transmit Data Buffer 9	TDB9	176606
	Port No. 10 (Keyboard No. 5 or PED No. 4)		
	Receive Status Register 10	RSR10	176610
	Receive Data Buffer 10	RDB10	176612
	Transmit Status Register 10	TSR10	176614
	Transmit Data Buffer 10	TDB10	176616
	Port No. 11 (Spare)		
	Receive Status Register 11	RSR11	176620
	Receive Data Buffer 11	RDB11	176622
	Transmit Status Register 11	TSR11	176624
	Transmit Data Buffer 11	TDB11	176626
	Port No. 12 (Spare)		
	Receive Status Register 12	RSR12	176630
	Receive Data Buffer 12	RDB12	176632
	Transmit Status Register 12	TSR12	176634
	Transmit Data Buffer 12	TDB12	176636
MULTIPORT SERIAL INTERFACE CARD NO. 4 (OPTIONAL)	Port No. 13 (Spare)		
	Receive Status Register 13	RSR13	176640
	Receive Data Buffer 13	RDB13	176642
	Transmit Status Register 13	TSR13	176644
	Transmit Data Buffer 13	TDB13	176646
	Port No. 14 (Spare)		
	Receive Status Register 14	RSR14	176650
	Receive Data Buffer 14	RDB14	176652
	Transmit Status Register 14	TSR14	176654
	Transmit Data Buffer 14	TDB14	176656

Table A-6. Register Designations and Address Assignments

DEVICE	REGISTER	MNEMONIC	ADDRESS
MULTIPORT SERIAL INTERFACE CARD NO. 4 (OPTIONAL) (Cont)	Port No. 15 (Spare)		
	Receive Status Register 15	RSR15	176660
	Receive Data Buffer 15	RDB15	176662
	Transmit Status Register 15	TSR15	176664
	Transmit Data Buffer 15	TDB15	176666
	Port No. 16 (Spare)		
	Receive Status Register 16	RSR16	176670
	Receive Data Buffer 16	RDB16	176672
	Transmit Status Register 16	TSR16	176674
	Transmit Data Buffer 16	TDB16	176676
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
	Bank Register	PGR	165014
	Display Z Register	DZR	165016
	X Position Register	DXR	165020
	Y Position Register	DYR	165022
	Display Character Register	DCR	165024
	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 thru 172356

Table A-6. Register Designations and Address Assignments (Cont)

DEVICE	REGISTER	MNEMONIC	DR # (OCTAL) ADDRESS
Graphic Controller VC 1	Memory status	STA1	4
	Start Y section 1	SY11	5
	Start X section 1	SX11	6
	Lines section 1	LN11	7
	Start Y section 2	SY12	11
	Start X section 2	SX12	12
	Lines section 2	LN12	13
	Start Y section 3	SY13	15
	Start X section 3	SX13	16
	Lines section 3	LN13	17
	X cursor address	XCR1	21
	Y cursor address	YCR1	22
Graphic Controller VC 2	Memory status	STA2	23
	Start Y section 1	SY21	24
	Start X section 1	SX21	25
	Lines section 1	LN21	26
	Start Y section 2	SY22	30
	Start X section 2	SX22	31
	Lines section 2	LN22	32
	Start Y section 3	SY23	34
	Start X section 3	SX23	35
	Lines section 3	LN23	36
	X cursor address	XCR2	40
	Y cursor address	YCR2	41
Graphic Controller VC 3	Memory status	STA3	42
	Start Y section 1	SY31	43
	Start X section 1	SX31	44
	Lines section 1	LN31	45
	Start Y section 2	SY32	47
	Start X section 2	SX32	50
	Lines section 2	LN32	51
	Start Y section 3	SY33	53
	Start X section 3	SX33	54
	Lines section 3	LN33	55
	X cursor address	XCR3	57
	Y cursor address	YCR3	60
Graphic Controller VC 4	Memory status	STA4	61
	Start Y section 1	SY41	62
	Start X section 1	SX41	63
	Lines section 1	LN41	64
	Start Y section 2	SY42	66
	Start X section 2	SY42	67
	Lines section 2	LN42	70
	Start Y section 3	SY43	72
	Start X section 3	SX43	73
	Lines section 3	LN43	74
	X cursor address	XCR4	76

Table A-6. Register Designations and Address Assignments (Cont)

DEVICE	REGISTER	MNEMONIC	DR # (OCTAL)	ADDRESS
Graphic Controller VC4 (Cont)	Y cursor address	YCR4	77	
	Left Margin	LMR		
	Pixel Data	PDR		
	Line Increment	DLI		
	Video Controller Directory	VCF		165046
	System Configuration	SCF		165050
	Display Configuration	DCF		165052
	Character Font	CFR		165070

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
	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	Unassigned
	Output Data	Unassigned
	Attention (Optional)	Unassigned
MULTIPORT SERIAL INTERFACE CARD NO. 1	Port 1 - Host Computer	
	Input	300
	Output	304
	Port 2 - Keyboard No. 3 or PED No. 6	
	Input	310
	Output	314
	Port 3 - Keyboard No. 1 or PED No. 8	
	Input	320
	Output	324
	Port 4 - Keyboard No. 8 or PED No. 1	
	Input	330
	Output	334
MULTIPORT SERIAL INTERFACE CARD NO. 2	Port 5 - (unused)	
	Input	340
	Output	344
	Port 6 - Keyboard No. 6 or PED No. 3	
	Input	350
	Output	354
	Port 7 - Keyboard No. 2 or PED No. 7	
	Input	360
	Output	364
	Port 8 - Keyboard No. 7 or PED No. 2	
	Input	370
	Output	374

Table A-7. Display Processor Trap Addresses (Cont)

INTERRUPTION DEVICE	INTERRUPT	TRAP ADDRESS
MULTIPORT SERIAL INTERFACE CARD NO. 3	Port 9 - Keyboard No. 4 or PED No. 5	
	Input	400
	Output	404
	Port 10 - Keyboard No. 5 or PED No. 4	
	Input	410
	Output	414
	Port 11 - Spare	
	Input	420
	Output	424
	Port 12 - Spare	
	Input	430
	Output	434
MULTIPORT SERIAL INTERFACE CARD NO. 4	Port 13 - Spare	
	Input	440
	Output	444
	Port 14 - Spare	
	Input	450
	Output	454
	Port 15 - Spare	
	Input	460
	Output	464
	Port 16 - Spare	
	Input	470
	Output	474

Material formerly on pages A-36 through A-66 has been deleted.

Table A-9. Character Generator Code Assignments

b_6 →						0	0	0	0	1	1	1	1
b_5 →						0	0	1	1	0	0	1	1
b_4 →						0	1	0	1	0	1	0	1
BITS	b 3 ↓	b 2 ↓	b 1 ↓	b 0 ↓	row ↓ col --->	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	STX		"	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	LF		*	:	J	Z	j	z
	1	0	1	1	11			+	;	K	[	k	{
	1	1	0	0	12			,	<	L	\	l	|
	1	1	0	1	13	CR		-	=	M	]	m	}
	1	1	1	0	14	SHIFT OUT		.	>	N	^	n	~
	1	1	1	1	15	SHIFT IN		/	?	O	-	o	

STX - Set margin
LF - Line Feed
CR - Carriage Return

Table A-10. Multiport Serial Interface Port Assigmments

For 4 Keyboards and 4 PEDS

CARD	CONNECTOR	PORT DESIGNATION	ASSIGNMENT
Multiport Serial Interface Card No. 1	J2 (RS-232C) or J3	1	Host Computer
	J4	2	Keyboard #3/PED #6
	J5	3	Keyboard #1/PED #8
	J6	4	PED #1/KBD #8
Multiport Serial Interface Card No. 2	J2 (RS-232C) or J3	5	Unused
	J4	6	PED #3/KBD #6
	J5	7	Keyboard #2/PED #7
	J6	8	PED #2/KBD #7
Multiport Serial Interface Card No. 3	J3	9	Keyboard #4/PED #5
	J4	10	PED #4/KBD #5
	J5	11	Spare
	J6	12	Spare

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	Monitor (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 8 TK messages can be used to transfer control.

Table A-12. Character Font Summary for GRAPHIC 7

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

For GRAPHIC 8

Character Font: 7 x 9 pixels for 1024 x 1024 screen or 1024 x 768*
Increment Units = Screen Coordinates

SIZE	NOMINAL HEIGHT (INCHES)	RECOMMENDED TEXT INCREMENT DECIMAL . (OCTAL)	RECOMMENDED LINE FEED INCREMENT DECIMAL . (OCTAL)	RECOMMENDED NUMBER OF CHARACTERS PER LINE	RECOMMENDED NUMBER OF LINES PER SCREEN
0	0.105	9. (11)	13. (15)	113.	78.
1	0.105	9. (11)	13. (15)	113.	78.
2	0.210	17. (21)	23. (27)	60.	44.
3	0.315	25. (31)	34. (42)	40.	30.
Character Font: 5 x 7 pixels for 512 x 512 screen or 640 x 480* (1 pixel = 2 screen coordinates)					
0	0.164	12. (16)	21. (25)	85.	48.
1	0.164	12. (16)	21. (25)	85.	48.
2	0.328	24. (30)	42. (52)	42.	24.
3	0.492	36. (44)	63. (77)	28.	16.

*Screen sizes 640 x 480 and 1024 x 768 have the same character font as 512 x 512 and 1024 x 1024, respectively, except there is a different number of characters per line.

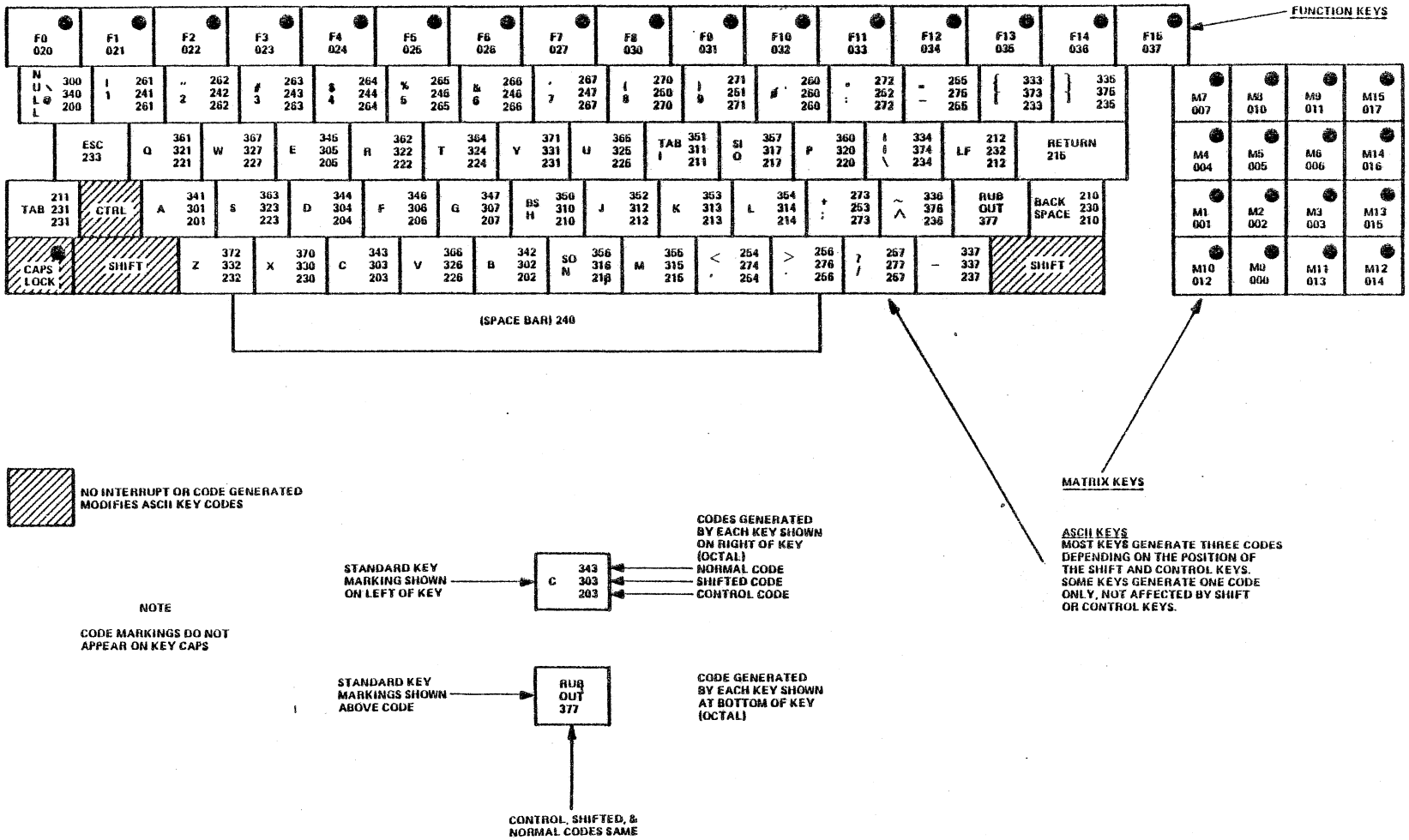


Figure A-2. Model 5784 Keyboard Layout and Code Assignments

Table A-13. 7-Bit ASCII Code

CHAR.	OCTAL CODE	DEC.	OCTAL SHF 10=	DEC.	CHAR.	OCTAL CODE	DEC.	OCTAL 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.	OCTAL SHF 10=	DEC.	CHAR.	OCTAL CODE	DEC.	OCTAL 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.	OCTAL SHF 10=	DEC.	CHAR.	OCTAL CODE	DEC.	OCTAL SHF 10=	DEC.
|	174	124	76000	31744	~	176	126	77000	32256
|	175	125	76400	32000	DEL	177	127	77400	32512

Table A-14. GRAPHIC 8 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 (DR0)	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	*
BANK 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	*

*These registers are written by refresh commands and read by programmed data transfers.

**The 2 bit Bank Register is read as bits 14 and 15 of the DPR.

***These registers are written by the I/O and refresh commands.

#A write to FUNC register while the digital graphic controller is running will cause an error trap through address 4 (error trap).

Table A-14. GRAPHIC 8 Registers (Cont)

REGISTER	MEMORY ADDRESS	I/O READ	I/O WRITE
#FUNCTION CONTROL CONTINUE (FUNC)	165036	No	Yes
FUNCTION CONTROL STOP (FUNS)	165040	No	Yes
ERROR REGISTER (ERR)	165312	Yes	Yes

***These registers are written by the I/O and refresh commands.

#A write to FUNC register while the digital graphic controller is running will cause an error trap through address 4 (error trap).

Table A-14. GRAPHIC 8 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

Table A-15. GRAPHIC 8 Instruction Timing

SANDERS' GRAPHIC 8

Vector Write Times

A. Vertical Solid Line

2.7 usec + 900 nsec/pixel

B. Horizontal Solid Line

4.5 usec + 600 nsec/pixel

C. 45° Line

6.6 usec + 1.2 usec/pixel

D. All Others

15.9 usec + 1.5 usec/pixel

Character Write Times (Small Size)

A. Overhead

1. CHAR instruction	4.5 usec	
2. TEXT instruction	8.4 usec	
B. "L" Character (15 pixels)	37.2 usec (5 x 7)	46.2 usec (7 x 9)
"E" Character (24 pixels)	38.7 usec (5 x 7)	52.2 usec (7 x 9)

Data Move

Configuration:

1. 8 bits/pixel	3.9 usec/pixel
2. 4 bits/pixel	5.1 usec/pixel

Point Plots	512^2	1024^2
PPTA	2.4 usec/pixel	2.1 usec/pixel
PPTR	2.7 usec/pixel	2.7 usec/pixel
PPYA/PPYR	2.1 usec/pixel	1.8 usec/pixel
PPLR	3.0 usec/pixel	2.7 usec/pixel

Polygon Fill

FLPG — 600 nsec/pixel average. Range is 100 ns/pixel to 1.0 us/pixel depending on the length and position of the horizontal pixel lines composing the fill.

Conics

LDKX, DRKY — 2.5 usec/pixel

APPENDIX B

GRAPHIC 8 MACRO DESCRIPTIONS

B-1. GENERAL

This appendix describes the GRAPHIC 8 display macros that have been developed by Sanders for Software Engineers 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 unless otherwise specified.

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 represents a single ASCII character.

   b. "arg" represents a specific argument identified in the macro description.

   c. "label" represents a label assigned by the aplications 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 are described in Sanders publication 79-0350.

2. All register mnemonics listed in table A-6 are defined in the GRAPHIC 8 macros and may be used as arguments for MACRO-11 instructions.

3. The following labels are used within the GRAPHIC 8 macros and must not be duplicated in any user written program that employs these macros:

ARG.OK	TI.
CH.	TMP$
DISLOP	TXT.
DPR.	YINC.
DZR.	

Table B-1. GRAPHIC 8 Display Macros

MACRO	FUNCTION
ADDI	Add to display register immediate
ADR	Absolute draw
AMV	Absolute move
CALL (E)	Call (Extended) subroutine
CALR (E)	Call relative
CHAR	Draw single character
CIRCLE	Draw circle
CLRM	Clear pixel memory
COLOR	Select color
CR	Carriage return control character
CRLF	Carriage return line feed control characters
DISEND	Display end
DISINT	Display initialize
DRKY	Draw conic Y
DRSR	Draw short relative
DRXA	Draw X absolute
DRXR	Draw X relative
DRYA	Draw Y absolute
DRYR	Draw Y relative
ELLIPSE	Draw ELLIPSE
ENTR	Provide subroutine entry point
FILL	Fill a convex polygon
HREF	Halt a refresh
INIT	Initialize
IZPR	Initialize the ramp generator (Graphic 7 only)
JMPM (E)	Jump and mark
JMPR	Jump short relative
JMPZ (E)	Jump (Extended) if display register 0 contents $\neq 0$
JPRZ (E)	Jump relative if display register 0 contents $\neq 0$
JRMP (E)	Jump relative
JUMP (E)	Jump (Extended)
LDDI	Load display register immediate
LDDP	Load display parameter register
LDDZ	Load display Z register
LDKX	Load conic X register
LDPD	Load pixel data register
LDRI	Load device register immediate (Graphic 7 only)
LDSP (E)	Load (Extended) stack pointer
LDTI	Load text and line increment registers
LDXA	Load X absolute
LDXR	Load X relative
LF	Line feed control character
LINK (E)	Synchronized linkage
MDLU	Modify look-up table
MODE	Load mode register

Table B-1. GRAPHIC 8 Display Macros (Cont)

MACRO	FUNCTION
MVPD	Move pixel data
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
PPTA	Point plot tabular absolute
PPTR	Point plot tabular relative
PPYA	Point plot Y absolute
PPYR	Point plot Y relative
RDR	Relative draw
RESD (E)	Restore display register
RLINK	Relink
RMV	Relative move
RTRN (E)	Return (Extended)
SAVD (E)	Save (Extended) display register
SETLF	Set line feed
SETMRG	Set left margin
SETTI	Set text increment
STX	Set left margin control character
TEXT	Draw tabular character string
TXT	Draw two tabular characters
UPDT	Update video controller register(s)
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" through "63", specifies one of the display registers (DR0 through DR63) of the digital 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 DYRA (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 instruction 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.
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 bank 0.
CALLE label, bank	Inserts a CALLE (call subroutine extended) instruction into the refresh file with the subroutine address specified by the combination of "bank" and "label". The subroutine address may be any even address in memory.
CALR (E) label	Inserts a CALR (E), call relative, instruction into the refresh file with the subroutine address specified by argument "label". Argument "label" may define any even location in memory bank 0.
CHAR a, <B,S,0>	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. If "0" argument is present, the argument "a" is interpreted as an octal equivalent number. If "S" argument is present, a shift out is applied to argument "a".

Table B-2. Detailed Macro Descriptions (Cont)

MACRO CALL	DESCRIPTION

LDDZ (cont)

Argument	Description	Octal Value
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.

MODE mode

Inserts a MODE (load mode register) instruction into the refresh file. "mode" = 0 indicates normal mode. "mode" = 1 indicates extended instruction mode. If "mode" is greater than 1, no change in mode will be made.

LDPD level

Inserts an LDPD (load pixel data register) instruction into the refresh file. Argument "level" which varies from 0 to 377 represents the gray level or color that is to be stored in the PDR (pixel data register) and used as the intensity of the pixels written by subsequent refresh instructions.

LDRI

Inserts an LDRI instruction (no operation for GRAPHIC 8).

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.

Table B-2. Detailed Macro Descriptions (Cont)

MACRO CALL	DESCRIPTION
LDSPE nnnnnn,bank	Inserts an LDSPE (load stack pointer extended) into the refresh file. The combination of "bank" and "nnnnnn" specifies the address to be loaded onto the stack in two successive words.
LDTI nn,11	Inserts an LDTI (load text and line 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. Argument "11" which may vary from 0 to 77, specifies the line increment to be used for the CR control character. If both arguments "nn" and "11" are not present and increments have previously been established, the established increments will be used. If argument "11" is not present and an increment has not previously been established, a default increment of "11" = 22. will be used.
LDXA nnnn	Inserts an LDXA (load X absolute) instruction into the refresh file. The argument "nnnn" which may vary from -2000 to 1777 defines the value or the x coordinate to be loaded in the current X Position Register.
LDXR nnnn	Inserts an LDXR (load X relative) instruction into the refresh file. The relative distance, in terms of coordinates, that is to be added to the Current X Position Register is specified by argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.
LF	Inserts CHAR 012,0 (line feed) into the refresh file.
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 address higher than 77776. If argument "label" is absent, the LINK instruction inserted into the refresh file will specify a direct link to address 170.
LINK label	If EIM = 1, then LINKE is used. Note "I" is not allowed.

Table B-2. Detailed Macro Descriptions (Cont)

MACRO CALL	DESCRIPTION
MDLU adr1,banks,bytes, vcnum,adr2,D	The data stored in sequential bytes starting at the GRAPHIC 8 memory address defined in the argument "adr1" is used to replace data in sequential bytes in the video controller look-up table starting at the address defined in "adr2". The combination of bank and adr1 is an absolute address if argument "D" is absent. adr1 is a relative address from the program counter if "D" is present. The video controller selected is defined by the value 1-4 in the argument "vcnum" and the number of bytes to be replaced is defined in the argument "bytes".
MVPD adr,bank,ix,iy, fx,fy,D,a,b,c	Inserts an MVPD (move pixel data) instruction into the refresh file. If argument "a" = "ABS" the vertices are expressed as absolute coordinates if "a" = "REL" the vertices are defined by the deltas from the current X and Y position. If argument "b" = "FROMPM", the transfer will be from mapping memory to GRAPHIC 8 memory. If "b" = "TOPM", the transfer will be from GRAPHIC 8 memory to mapping memory. If argument "c" = "VERT" the scan will be bottom to top - left to right if "c" = "HOR" the scan will be left to right - bottom to top. If argument "D" is present the displacement (in even bytes) to "ADR" will be calculated and stored in the address word. If "D" is absent, the combination of "bank" and "ADR" will be stored. "ix" and "iy" indicate the initial values for X and Y, respectively. "fx" and "fy" indicate the final X and Y, respectively.
MVSR xx,yy	Inserts an MVSR (move short relative) instruction into the refresh file. Arguments "xx" and "yy" specify the values to be added to the Current X and Y Position Registers, 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 Current X Position Register is to be changed 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 Current X Position Register is to be changed is specified by Argument "nnnn". Argument "nnnn" may vary from -2000 to 1777.

*MODE 0 only

Table B-2. Detailed Macro Descriptions (Cont)

MACRO CALL	DESCRIPTION
MVYA nnnn	Inserts MVYA (move Y absolute) instruction into the refresh file. The Y axis coordinate to which the Current Y Position Register is to be changed 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 Current Y Position Register is to be changed 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 insructions 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) 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. The Macro is in no way related to the STX control character.
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 Current X Position Register is to be moved. If argument "R" is absent, the Current Y Position will be moved to a new line for normal characters; if "R" is present, the Current X Position will be moved to a new line for rotated characters. Argument "ssss" specifies the number of spaces. Positive arguments move the Current X Position Registers left; negative arguments move the Current X Position right. If a text increment has not been established by a previous macro, a value of 14 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 Current Y Position is to be moved. Argument "1111" specifies the number of lines. Positive arguments move the Current Y Position down; negative arguments move the Current Y Position up. If a line feed increment has not been established by a SETLF (set line feed) macro, a value of 21 is assumed. Argument "1111" multiplied by the line feed increment may vary from -2000 to 1777. The Macro is in no way related to the LF control character.
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 Current X and Y Position are to be moved, respectively. A point is then displayed at the new position. Both "xx" and "yy" may vary from -40 to 37.
PPTA a	Inserts a PPTA (point plot tabular absolute) instruction with a = ± X (or Y) coordinate (range -2000 to 1777) into the refresh file.
PPTR a	Inserts a PPTR (point plot tabular relative) instruction with a = ± X (or Y) increment (range from -2000 to 1777) into the refresh file.
PPYA a	Inserts a PPYA (point plot Y absolute) with a = ± X (or Y) coordinate (range from -2000 to 1777) into the refresh file.
PPYR a	Inserts a PPYR (point plot Y relative) with a = ± X (or Y) increment (range from -2000 to 1777) into the refresh file.
RDR dx,dy	Causes a relative draw by inserting two instructions an LDXR (load X relative) and a DRYR (draw Y relative) into the refresh file. Arguments "dx" and "dy" specify the X and Y deltas, respectively.
RESD n	Inserts an RESD (restore display register) instruction into the refresh file. Argument "n" specifies the display registers to be loaded from the top of the processor stack.
RESDE n	Inserts an RESDE (restore display register extended) into the refresh file. Argument "n" specifies the display register to be loaded from the top of the processor stack.
RLINK	Insert MOV (move) and RTI (return from interrupt) display processor instructions into the LINK interrupt service routine to restart the graphic controller and return the display processor to its previous state.
RMV dx,dy	Causes a relative move by inserting two instructions an LDXR (load X relative) instruction and an MVYR (move Y relative) instruction into the refresh file. Arguments "dx" and "dy" specify the X and Y deltas, respectively.
RTRN	Inserts an RTRN (return from refresh subroutine) instruction into the refresh file.
RTRNE	Inserts on RTRNE (return from refresh subroutine extended) instruction into the refresh file.

Table B-2. Detailed Macro Descriptions (Cont)

MACRO CALL	DESCRIPTION
SAVD n	Inserts an SAVD (save display register) instruction into the refresh file. Argument "n" specifies the display register to be stored on the processor stack.
SAVDE n	Inserts an SAVDE (save display register extended) instruction into the refresh file. Argument "n" specifies the register to be stored on the processor stack.
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 Current Y Position down; negative arguments result in increments that move the Current Y Position up. Argument "nnnn" may vary from -2000 to 1777. Refer to table A-12 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 14 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 Current X Position to the right; negative arguments result in increments that move the Current X Position to the left. Argument "nnnn" may vary from -2000 to 1777. Refer to table A-12 for recommended text increments.
STX	Inserts a CHAR 002,0 (Set Margin control character) into the refresh file.
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,0	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. TXT will accept an equivalent octal number if the argument "0" is present.

Table B-2. Detailed Macro Descriptions (Cont)

MACRO CALL	DESCRIPTION
UPDT	Inserts a UPDT (update Video Controller registers) instruction into the refresh file.
WATE	Inserts a WATE (wait for vertical axes enable toggle and clear pixel memory) instruction into the refresh file.

Table B-3. Typical Program Structure

```
;
 ;
;SAMPLE PROGRAM NO. 1.  A SIMPLE DRIVER, NOT USING THE DISINT
;  AND DISEND MACROS, WITH A PLACE FOR DISPLAY INSTRUCTIONS.
;
.TITLE  SAMPL1
.SBTTL  SAMPL 1 DRIVER
;
;
        .ASECT                                  ;BEGIN ASSEMBLY
        .=2000                                  ;  AT ADDRESS 2000
        .BLKW   30                              ;SAVE SPACE FOR DISPLAY PROCESSOR STACK
BEGIN:  RESET                                   ;CLEAR PROCESSOR BUS
        MOV     #BEGIN,SP                       ;LOAD DISPLAY PROCESSOR STACK POINTER
        CLR     @#FUNS                          ;HALT GRAPHIC CONTROLLER
HTST:   BIT #20, @#SENS BEQ HTST                ;WAIT FOR HALT

        MOV     #START,@#DPC                    ;START GRAPHIC CONTROLLER
LOOPL   WAIT                                    ;DISPLAY PROCESSOR WAIT
        BR      LOOP                            ;WAIT LOOP
START:  LDDZ    <ALL,BLOFF,LINE,BR7>            ;SET Z PARAMETERS
        LDDP                <.NOROTATE,CSO>     ;SET P PARAMETERS
        LDTI    12                              ;SET TEXT INCREMENT TO 12
AGAIN:  IZPR                                    ;INITIALIZE GRAPHIC CONTROLLER
        LDXA    0                               ;MOVE CRT BEAM(S)
        MVYA    0                               ;  TO CENTER
        WATE                                    ;WAIT
;
;       BODY
;        OF
;         USER
;          WRITTEN
;           DISPLAY
;            PROGRAM
;             GOES
;              HERE
;
        JUMP    AGAIN                           ;SHOW THE PICTURE AGAIN
        .END    BEGIN
;
;
;SAMPLE PROGRAM NO. 2.  A SIMPLE DRIVER, USING THE DISINT
;  AND DISEND MACROS, WITH A PLACE FOR DISPLAY INSTRUCTIONS.
;
.TITLE  SAMPL2
.SBTTL  SAMPL2 DRIVER
;
;
        .ASECT                                  ;BEGIN ASSEMBLY
        .=2000                                  ;  AT ADDRESS 2000
        .BLKW   30                              ;SAVE SPACE FOR DISPLAY PROCESSOR STACK
BEGIN:  RESET                                   ;CLEAR PROCESSOR BUS
        MOV     #BEGIN,SP                       ;LOAD DISPLAY PROCESSOR STACK POINTER
        DISINT                                  ;SET PARAMETERS AND START GRAPHIC CONTROLLER
;
;       BODY
;        OF
;         USER
;          WRITTEN
;           DISPLAY
;            PROGRAM
;             GOES
;              HERE
;
        DISEND                                  ;SHOW THE PICTURE AGAIN
        .END    BEGIN
;
;
```

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

All previously developed GCP programs should still run with GCP.

C-2. No user refresh programs should start below address 3000 (octal).

NOTE

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.

Name: _____

Company: _____

Address: _____

_____

Telephone: [ ] _____

Date: _____

CalComp Equipment _____

Part Number _____

Software/Firmware System _____

Version _____

Host computer _____

Host operating system _____ Version _____

Host-Vistagraphic interface _____

My problem is: hardware ☐ software ☐

firmware ☐ manual ☐

Description of problem (or suggestion for improvement):

Related tech manual number _____

THE INTENT AND PURPOSE OF THIS PUBLICATION IS TO PROVIDE ACCURATE AND MEANINGFUL INFORMATION TO SUPPORT EQUIPMENT MANUFACTURED BY CALCOMP/SANDERS. 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

CalComp
Display Products Division
Daniel Webster Highway, South
P.O. Box 868
Nashua, NH 03061

FOLD FOLD

CALCOMP
A Sanders Graphics Company
SANDERS