PICTURE SYSTEM 2/PDP-11

REFERENCE MANUAL

Evans & Sutherland Computer Corporation
580 Arapeen Drive
Salt Lake City, Utah 84108

First Edition November 22, 1976

This manual was prepared by:

M. Mantle
D. Mortenson

All reference to this document should be made to:
Document: E&S #901130-100 NC

PICTURE SYSTEM 2/PDP-11 Reference Manual - First Edition 11/22/76
Amendment #1

Change Notice:

The following errors have been noted in documents:
E&S #901130-100 NC

1. Page 2-85.

   The diagram of the Status Command shows PFORM CCHAR is bit 6 of word 1. It is actually bit 5. The diagram of the Status Command shows CG RESET is bit 5 of word 1. It is actually bit 6.

2. Page 2-88.

   PFORM CCHAR is specified to be bit 6 of word 1 of the Status Command. It is actually bit 5. CG RESET is specified to be bit 5 of word 1 of the Status Command It is actually bit 6.

3. Page 2-107

   Function 04 of the Character Generator CONTROL instruction reads as follows:

   "04 Set the current X,Y position to the saved HOME position. If M=1, then End."

   It should read:

   "04 Set the current X,Y position to the saved HOME position. If M=1, then set X,Y position to the saved carriage return position (CR position) and End.

PICTURE SYSTEM 2/PDP-11 Reference Manual - First Edition 11/22/76
Amendment #2
Change Notice:

The following errors have been noted in document:
E&S #901130 NC

1. Page 2-53.
   The paragraph describing the Pass Formatted:
   FSM2 value (PF-5) states the following:

   "For a 2DRAW command, the X, Y coordinates are taken from the MAP input FIFO and the Z coordinate is taken from the BASE register."

   It should read:

   "For a 2DRAW command, the X, Y coordinates are taken from the MAP input FIFO and the Z coordinate data is undefined.

2. Page 2-61.
   Word 1 of the data format shown on this page is in error. Bit 12 of Word 1 is shown as a 0. It should be a 1 resulting in an FSM2 field <13:11> of 110.

3. Page 2-148.
   The default Alphanumeric Keyboard Data Register (KBDR) is specified to be at location 177620 in the System Control Block. It should read:

   "a. Keyboard Data Register (KBDR) - SCB:177607"

Amendment #2 continued.

4. Page 2-154.
The Map of the System Control Block has incorrect address assignments for the Control Dials/Joystick and Keyboards. It should read:

```
177500-177577  Reserved for Control Dials/Joysticks
    .
    .
    .
177600-177606  KB2-8    KBDR
177607         KB1
```

5. Page 3-12.
Figure 3-5 is incomplete. The Standard and Italicized characters range from .36 cm to 1.88 cm in size as described in Section 2.4.4.1.

6. Page A-4+.
The Appendix A Sample Program contains an error on page 3 of the MACRO-11 assembly language listing.
Page 3, line 25 should read:

```
"RTCCNT      =177744      ;PS2 RTCCNT SCB REGISTER"
```

Page 3, line 28 should read:

```
"DMAPSA      =177747      ;PS2 DMAPSA SCB REGISTER"
```

PREFACE

The PICTURE SYSTEM 2/PDP-11 Reference Manual provides the hardware and software details essential in understanding PICTURE SYSTEM 2 and the PDP-11 interface at a system level.

Chapter 1 contains an overview of the PICTURE SYSTEM 2 hardware at a functional level.

Chapter 2 describes the hardware details of PICTURE SYSTEM 2 and the interface to the PDP-11. The information contained in this chapter is detailed to provide a full understanding of the PICTURE SYSTEM 2 hardware components.

Chapter 3 details the manner in which the PICTURE SYSTEM is used by the Graphics Software Package provided with the system.

Chapter 4 contains the 4x4 matrix transformations used to implement rotation, translation, scaling, windowing, etc. in PICTURE SYSTEM 2.

The appendices contain additional information which pertains to the programming and use of PICTURE SYSTEM 2 and the PDP-11.

Throughout this manual, three number systems will be used--octal, binary, and decimal. To avoid cluttering all numbers with subscripted bases, the following general convention will be used:

Octal - for address locations, contents of addresses, character codes and operation codes for instructions; in most cases there will be words of 6 octal digits.

Binary - for describing a single binary element.

Decimal - for all normal referencing to quantities.

The terms PICTURE SYSTEM 2 and PICTURE SYSTEM are used interchangably throughout this document.

TABLE OF CONTENTS

CHAPTER ONE

1. OVERVIEW OF PICTURE SYSTEM 2

This chapter provides an overview of the hardware components of PICTURE SYSTEM 2. The basic hardware components comprising PICTURE SYSTEM 2 consist of the Picture Controller Interface, PICTURE SYSTEM Data Bus, Picture Processor, PICTURE SYSTEM Memory, Picture Generator, Picture Display and Interactive Devices. These components are used in conjunction with the Picture Controller to comprise a stand-alone computer graphics system. All operations performed by the PICTURE SYSTEM and its components are overlapped to allow for simultaneous throughput in each of the processing units. A functional diagram of PICTURE SYSTEM 2 is shown in Figure 1-1.

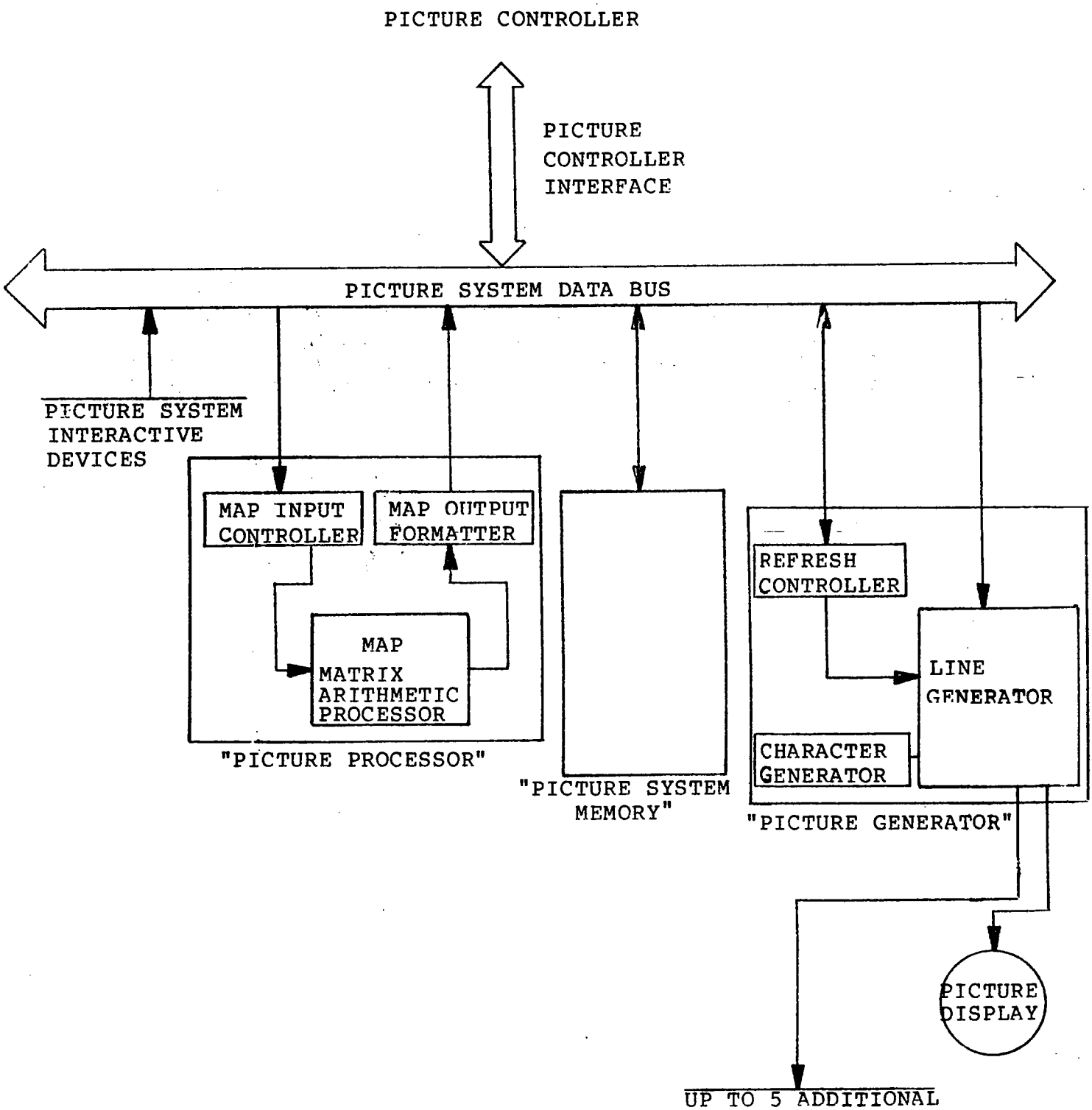


Figure 1-1

PICTURE SYSTEM 2 Components

The Picture Controller is a general-purpose computer, typically a minicomputer such as Digital Equipment Corporation's PDP-11, which directs the display of pictures on PICTURE SYSTEM 2. The Picture Controller communicates with the PICTURE SYSTEM through a Direct I/O and Direct Memory Access component called the Picture Controller Interface.

The Picture Controller Interface allows the PICTURE SYSTEM work station to be positioned up to 500 feet in distance from the Picture Controller.

The Picture Controller is used to:

* Contain the data base describing the object(s) to be viewed.
* Control processing of the object coordinate data by the Picture Processor.
* Perform all input and output required to facilitate graphical interaction.
* Compute parameters for use in simulation of object movement, data representation, etc.
* Perform all standard operating functions required by the operating system under which the control program executes.

All PICTURE SYSTEM 2 components and interactive devices are connected to and interact with the synchronous PICTURE SYSTEM Data Bus. These devices and components, as listed below, are described in detail in the following sections.

* The Picture Controller Interface
* The PICTURE SYSTEM Data Bus
* The Picture Processor
* The PICTURE SYSTEM Memory
* The Picture Generator
* The Interactive Devices

1.1 Picture Controller Interface

The Picture Controller Interface has a double cable which runs from the Picture Controller to the PICTURE SYSTEM work station. This cable allows the PICTURE SYSTEM hardware to be located up to 500 feet from the Picture Controller.

Communication between the Picture Controller and the PICTURE SYSTEM is achieved through the use of two I/O paths:

1. Direct I/O (DIO) path
2. Direct Memory Access (DMA) path

The Direct I/O path is typically used to pass single-word commands to or from the PICTURE SYSTEM. This provides convenient access to the PICTURE SYSTEM registers by the computer.

Using the DMA path, blocks of coordinate data may be transferred to or from the PICTURE SYSTEM without direct supervision by the Picture Controller.

An example of the use of the Direct I/O path would be the transfer of coordinate data to the PICTURE SYSTEM for processing while the DMA path is concurrently transferring the processed data back to the Picture Controller. Another example would be the use of the DMA to transfer data to the Picture Processor for processing at the same time the Direct I/O path is setting up for a current frame to be refreshed. In both cases, these data paths may be used concurrently for information flow.

The PICTURE SYSTEM Interface also provides interrupt capabilities for management of PICTURE SYSTEM 2 components and interactive devices. The PICTURE SYSTEM Interface distributes data to and retrieves data from the various components and devices of the PICTURE SYSTEM via the PICTURE SYSTEM Data Bus.

1.2 PICTURE SYSTEM Data Bus

All PICTURE SYSTEM 2 components and devices connect to and interact with each other on a single, high-speed synchronous data bus. Memory, device registers, and command, control and status registers all exist and are addressable memory locations on the PICTURE SYSTEM Data Bus.

Use of the Data Bus allows coordinate data to be transferred from the Picture Controller to the Picture Processor while data are concurrently being transferred from the Picture Processor to the PICTURE SYSTEM Memory. During this process, data are also being transferred from the PICTURE SYSTEM Memory to the Picture Generator for display. In addition, data may be entered from the Data Tablet or other interactive devices and read by the control program in the Picture Controller.

Data flow is supervised by a bus arbitration system which is integral to the PICTURE SYSTEM Data Bus.

1.3 Picture Processor

The Picture Processor is a high-speed, microprogrammed, digital arithmetic processor capable of accepting two-, three-, or four-dimensional (homogeneous) data; transforming the data; clipping the transformed data at the boundaries of a six-sided window; performing a perspective division and viewport mapping on the clipped data and outputting the transformed, clipped and viewport mapped data for subsequent display. There are three basic units in the Picture Processor:

* MAP Input Controller
* Matrix Arithmetic Processor (MAP)
* MAP Output Formatter

The MAP Input Controller receives data, typically from the Picture Controller, and channels the data to the Matrix Arithmetic Processor. The MAP Output Formatter receives the processed data from the Matrix Arithmetic Processor and outputs the data format for subsequent display. The MAP Output Formatter may alternately output the data in full 16-bit precision for use by the Picture Controller.

The Matrix Arithmetic Processor (MAP) is the major unit of the Picture Processor and consists of a Transformation Matrix, Transformation Matrix Stack, Parameter Register File and Arithmetic Unit.

The Transformation Matrix is a 4x4 element matrix, where each element is a 16-bit word. This 4x4 matrix is used to transform object coordinate data. It can also be concatenated with other 4x4 matrices to obtain a combined transformation.

The Transformation Matrix Stack is a storage area where up to fourteen 4x4 matrices may be "stacked" or saved for future recall.

The Parameter Register File is an array of 16-bit registers into which parameters specifying viewport boundaries, scale factors, etc. are stored and may be retrieved.

The Arithmetic Unit performs all scalar, vector and matrix arithmetic operations in the Picture Processor. This includes subtraction, addition, multiplication, division and normalization.

The MAP utilizes its units to perform digital operations on the data received from the Picture Controller. These operations are:

* To transform two-, three- and four-dimensional data.
* To push the Transformation Matrix onto the Matrix Stack.
* To pop the top 4x4 matrix of the Matrix Stack into the Transformation Matrix.
* To load the Transformation Matrix with data from the Picture Controller or PICTURE SYSTEM Memory.
* To store the contents of the Transformation Matrix into the Picture Controller or PICTURE SYSTEM Memory.
* To concatenate the contents of the Transformation Matrix with a 4x4 matrix in the Picture Controller or PICTURE SYSTEM Memory to obtain a compound transformation.
* To load and store the registers of the Picture Processor.
* To check transformed coordinate data for visibility by comparing with a two- or three-dimensional viewing window. Lines or portions of lines outside the window are removed by a clipping process so that only visible line segments are processed further.
* Three-dimensional data are converted to two dimensions by computing perspective or orthographic views.
* To perform a linear mapping of points from the object's coordinate system into that of the Picture Display or any other coordinate system specified.

1.4 PICTURE SYSTEM Memory

The PICTURE SYSTEM Memory is a dual-port MOS memory (distinct from the Picture Controller's) organized as addressable 16-bit words. This memory is available in increments of 16K words, expandable to 64K words of memory, dependent upon user requirements.

PICTURE SYSTEM Memory may be used in a variety of ways to satisfy the user's application. Typically, a portion of the PICTURE SYSTEM Memory serves as a refresh buffer into which data, still in digital form, is deposited. This data represents information to be shown on the Picture Display. For each frame displayed, the Refresh Controller reads the data from the PICTURE SYSTEM Memory and channels this data to the Line Generator where the data are then converted to analog signals to drive the Picture Display.

The ways in which the PICTURE SYSTEM Memory may be used as a refresh buffer include the single-buffer mode, double-buffer mode and segmented-buffer mode. (See Refresh Controller, Section 1.5.1, for further information.)

1.5 Picture Generator - Picture Display

The Picture Generator is a major component of PICTURE SYSTEM 2 whose primary function is to generate images to be viewed on the Picture Display.

The Picture Generator receives digital data information and converts this data into analog signals which are used to draw the image on the Picture Display.

The Picture Generator consists of three basic units:

* the Refresh Controller
* the Character Generator
* the Line Generator

1.5.1 Refresh Controller

The Refresh Controller is the unit of the Picture Generator that controls the refreshing of images on the Picture Display. The Refresh Controller reads data from the PICTURE SYSTEM Memory, a refresh buffer, and channels this data to the Line Generator for display. Under program supervision, the Refresh Controller is used to manage the organization of the PICTURE SYSTEM Memory. It also contains special-purpose hardware to facilitate memory segmentation and management.

In single-buffer mode, the entire refresh buffer is used to store a single display frame. In this mode, refresh may be initiated from a partially-updated display frame consisting of some lines from the new frame and some lines from the previous frame.

In double-buffer mode, one-half of the refresh buffer is designated as an old frame and one-half a new frame. Refresh is then initiated from the old frame while the new frame is being constructed. When construction of the new frame is completed, the frame buffers are swapped and the newly-constructed frame is displayed. The space occupied by the old frame becomes available for new frame construction.

The segmented-buffer mode provides the most general use of the refresh buffer for the display and updating of data. Typically, a frame consists of portions which need not be updated as frequently as others. Ideally, these portions should be updated as separate parts, or segments, of the frame. The Refresh Controller facilitates the use of the refresh buffer in this mode by allowing each of the separate portions of the refresh buffer to be given a name by which the segment may be replaced, appended to, deleted, etc.

The Refresh Controller also improves the utilization of the refresh buffer by providing, in segmented-buffer mode, for the reclamation of unused portions of the refresh buffer that have been left by deleted segments. This prevents fragmentation of the refresh buffer into small, unuseable areas.

1.5.2 Character Generator

The Character Generator is the unit of the Picture Generator that accepts and interprets character codes from the Line Generator, producing strokes which are then channeled back to the Line Generator for display. The microprogrammable Character Generator provides the basic capabilities to interpret the full 128 ASCII character set for generation of the 95 displayable ASCII character subset and for performing character positioning commands (i.e., carriage returns, line feeds, etc.) The Character Generator also allows the user to establish the left and top margins for use with the character positioning codes.

As a standard feature, the Character Generator is capable of creating regular and italicized characters in eight sizes, ranging from .08 cm (.03") to 1.88 cm (.74") in size. It is also capable of producing subscript and superscript characters which are approximately two-thirds the currently selected character size. A unique capability of the Character Generator is its ability to terminate generation of characters when a character string extends beyond the right boundary of the Picture Display. This hardware feature frees the user from concern of character wrap-around on the display.

The Character Generator is capable of being microprogrammed by the PICTURE SYSTEM user to produce alternate character fonts (or special symbols) at varying sizes and orientations. This ability allows characters to be produced at any orientation (i.e., horizontal, 90-degrees counter-clockwise,

45-degrees counter-clockwise, etc.) while maintaining the correct response to the character positioning commands.

The Character Generator has a memory, half of which is pre-programmed to interpret the 128 ASCII characters as described above, and half is available for the user to program various character fonts and special symbols.

1.5.3 Line Generator

The Line Generator receives data detailing coordinate point positions, status information which describes the modes of operation, and character codes that are passed to the Character Generator for interpretation. This information is used to produce the final image seen by the viewer.

Coordinate point positions are specified by Move or Draw commands in the Picture Display coordinate system. Each coordinate point, defined as X, Y and intensity (Z), causes the Line Generator to position the electron beam of the Picture Display to the location specified. If the command is a Move, the beam will then be positioned without intensification. If the Line Generator is currently in Dot mode, the beam will be positioned and then intensified. If the command is a Draw, a line will be drawn from the current beam location, positioned from a previous Move or Draw command, to the coordinate specified. The Line Generator has the capability of varying the intensity of a line between its endpoints. The line drawn will be in the line texture (dashed, etc.) currently selected for the Line Generator.

A status command to the Line Generator specifies the modes in which lines and characters are to be displayed. These modes are:

* Blink
* Texture
* Color
* Display Select

Setting the Blink mode indicates that all subsequent lines and characters output to the display are to blink until this mode is reset.

The Texture mode is used to indicate the manner in which all subsequent lines and characters are to be drawn by the Line Generator. Line textures available are: solid line mode; long dashed line mode; long-short dashed line mode; long-short-short dashed line mode and short dashed line mode.

For all of the above dashed line textures, the Line Generator ensures that dashes begin and end at the endpoints on the line.

Conversely, the Line Generator may be set to a textured line mode whereby connected line segments may be displayed without concern for individual line endpoints. This textured line mode effectively allows the appearance of a smoother curve by de-emphasizing the individual line segment endpoints.

Color selection is used in conjunction with the beam penetration monitor to initiate changes in color for all subsequent lines and characters which are output to the display after this mode is initiated.

In connection with the Color mode selection, an intensity selection is also used to control the drawing speed of the Line Generator. This change in drawing speed is utilized to present an appearance of equal intensity for the various colors available (i.e., red, red-orange, orange, yellow and green).

The Display Select mode is used in multiple Picture Display configurations in choosing to which Picture Display all subsequent lines and characters are to be output. The Display Select capabilities allow one or more images to be viewed on any combination of the six possible Picture Displays.

Character codes are not interpreted by the Line Generator, but are passed to the Character Generator. The Character Generator interprets the character codes, generating individual strokes which are then channeled back to the Line Generator for display.

The Line Generator is capable of displaying lines and characters at four distinct intensity levels under program control. The depth-cue feature increases the number of intensity levels which may be displayed from 4 to 64. This fea-

ture also allows the intensity of a single line to vary, from bright to dim according to its Z value, imparting the illusion of depth to three-dimensional images.

1.5.4 Picture Display

The Picture Display receives analog signals from the Picture Generator which are used for electron beam positioning and intensity control. The Picture Generator controls beam positioning and the drawing of all vectors, characters and dots on the Picture Display.

1.6 PICTURE SYSTEM Interactive Devices

All PICTURE SYSTEM 2 interactive devices are interfaced directly to the PICTURE SYSTEM Data Bus. Data may be input from the interactive devices by the control program using the Direct I/O path of the Picture Controller Interface. These interactive devices may be used under interrupt control or may be polled directly by the user program in the Picture Controller.

The following interactive devices are supported by PICTURE SYSTEM 2:

1. Tablet
2. Control Dials
3. Function Switches & Lights
4. Alphanumeric Keyboard

The use of these graphical input devices provide all the capabilities typically required for graphical interaction with PICTURE SYSTEM 2. Appropriate use of these interactive devices, along with the dynamic qualities of PICTURE SYSTEM 2, provide the user with the tools required for a three-dimensional, truly-interactive graphics system.

1.6.1 Tablet

The Tablet serves as the standard, general-purpose graphic input device in PICTURE SYSTEM 2. The Tablet can be used for positioning or pointing to the picture elements by use of a stylus, or pen, whose x,y coordinates are read by the Picture Controller. A "cursor" may be drawn on the Picture Display to indicate the position of the pen on the tablet. In conjunction with the Picture Processor, the tablet and pen can perform the interactive functions usually reserved for such graphic input devices as light pens, joysticks and function switches.

1.6.2 Control Dials

Control Dials permit the user to control size, position and orientation of objects or other functions required by the

application.

1.6.3 Function Switches & Lights

Function Switches & Lights provide the user the capability to utilize switches for functions assigned under program control. The lights may be used to indicate function switch setting or to display programmed information back to the user.

1.6.4 Alphanumeric Keyboard

The Alphanumeric Keyboard is a standard 61-key, 128-character keyboard used for text or data input to the Picture Controller for graphical interaction or other functions required by the application.

CHAPTER TWO

2. PICTURE SYSTEM 2

PICTURE SYSTEM 2 consists of:

1. The PICTURE SYSTEM 2 Data Bus (PSBUS) and Memory (PSMEM)
2. The PICTURE SYSTEM 2/PDP-11 I/O Interface
3. The Picture Processor
4. The Picture Generator
5. The Picture Display
6. Interactive Devices

Figure 2-1 shows an overview of the system and the interconnection of each of these units. The following sections of this chapter detail the functions performed, command and data formats, and command and status register definitions of each unit.

Throughout this chapter, repeated reference is made to the terms INIT, PSRESET, and RESET. INIT is the name given the UNIBUS initialization signal issued by the PDP-11 on power-up, execution of the PDP-11 RESET instruction and the pressing of the START switch on the computer console. PSRESET is the name given the PSBUS initialization signal issued on PICTURE SYSTEM power-up or the setting of the PSRESET bit in the I/O status register (see Section 2.2.2). Each PICTURE SYSTEM 2 component has its own RESET capability. In each of the individual component descriptions this is referred to as RESET. PSRESET performs the equivalent of the RESET for all of the components of PICTURE SYSTEM 2. PSRESET is not issued by the PDP-11 INIT signal.

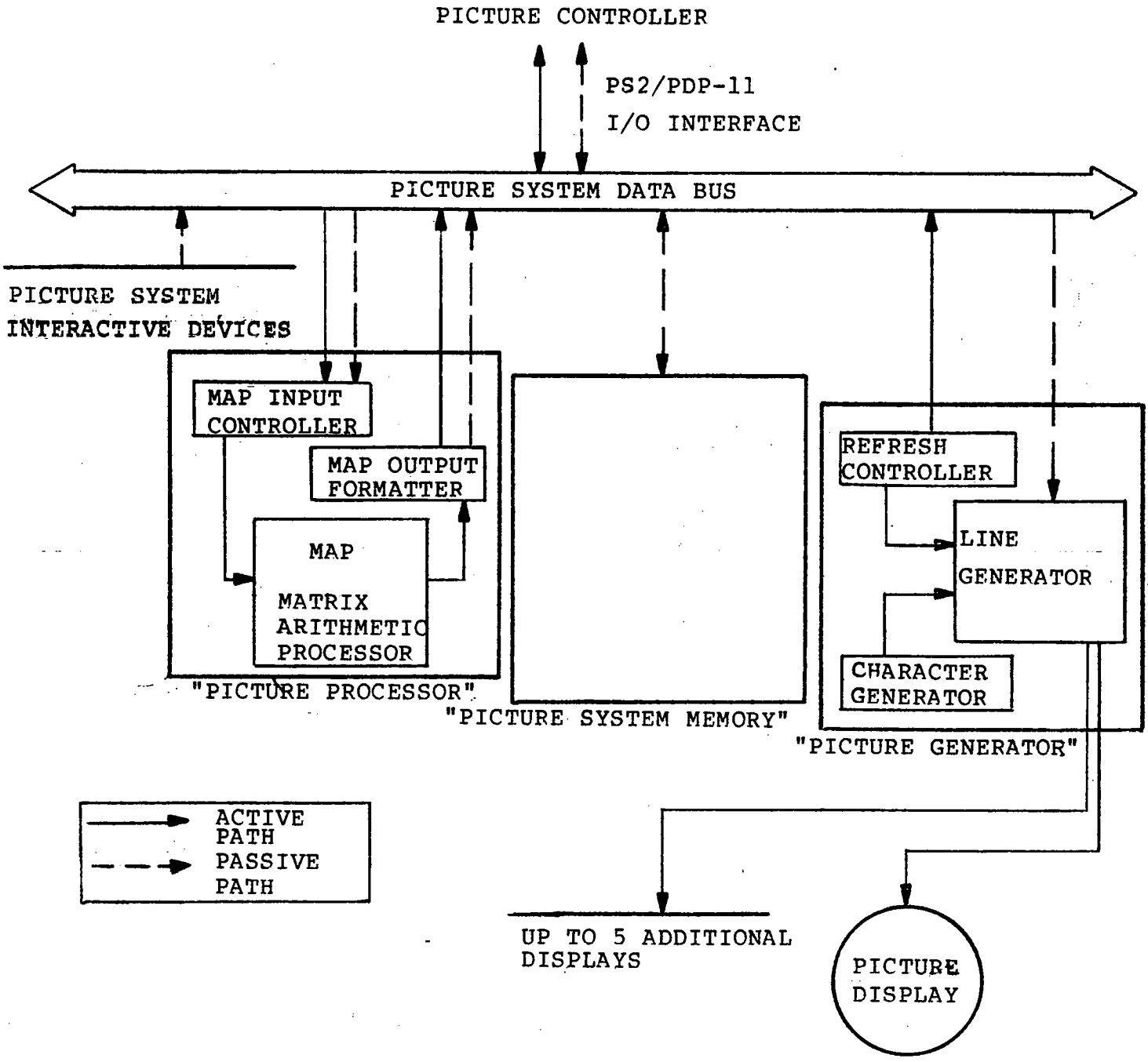


Figure 2-1

PICTURE SYSTEM 2 Block Diagram

2.1 The PICTURE SYSTEM Data Bus (PSBUS)

All PICTURE SYSTEM components and devices connect to and communicate with each other on a single, high-speed synchronous data bus (PSBUS). Addresses, data and control information are channeled through the 59 lines of the bus. Memory, device registers, and command, control and status registers are referenced as addressable memory locations on the PSBUS. All devices on the PSBUS (including memory) are considered as either "Active" or "Passive" devices. An Active device can initiate and control the transfer of data on the PSBUS to or from a Passive device. A Passive device will read or write data only when requested to do so by an Active device. An example of an Active device is the PICTURE SYSTEM 2/PDP-11 Direct I/O Interface which actively tranfers data to or from the PICTURE SYSTEM memory. The PICTURE SYSTEM Memory, in this example, is the Passive device.

PSMEM always passive

2.1.1 PICTURE SYSTEM Memory (PSMEM)

The PICTURE SYSTEM memory is a Passive device which consists of a dual-port MOS memory organized as 16-bit words. PSMEM is configured in increments of 16K words of memory dependent upon the system configuration and user requirements. Each memory location has a unique address from 0 to 177777 as shown in Figure 2-2.

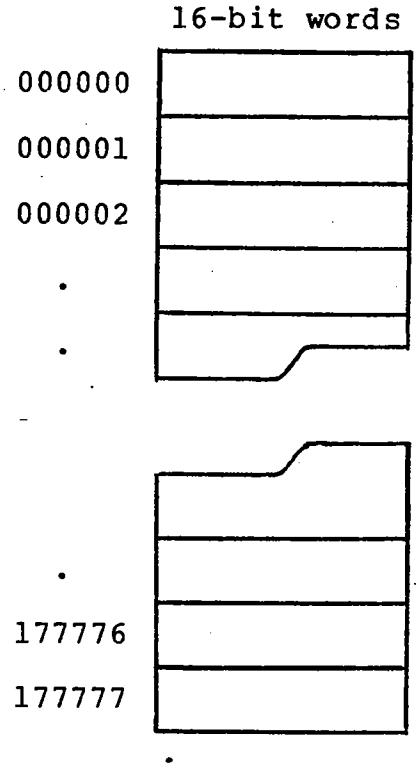


Figure 2-2
PICTURE SYSTEM Memory Addresses

Certain memory locations have been reserved for system use as command and control registers and peripheral device registers. Specifically, the top 256 memory addresses (locations 177400-177777) are reserved. This area is referred to as the System Control Block (SCB). All other memory locations may be used as refresh buffer areas for transformed data which is to be displayed or for display list regions for data which is to be transformed for subsequent display. Thus, a maximum of 63.75K 16-bit words are available for general program use. It should be noted that the SCB is always located at 177400-177777; hence, actual loss of general purpose memory occurs only when the system has 64K of memo-

ry. Additionally, the SCB is always present on the system, including a system with no general-purpose memory, regardless of the actual memory configuration. Figure 2-3 shows the maximum memory configuration of the PICTURE SYSTEM.

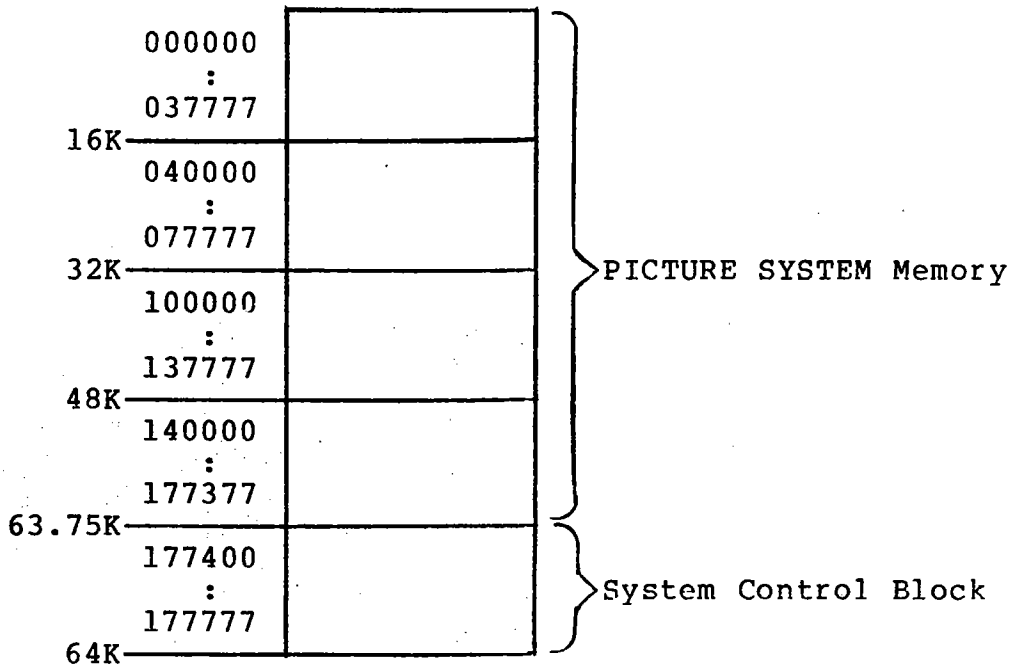


Figure 2-3
PICTURE SYSTEM Memory

2.1.2 System Control Block (SCB)

The SCB is a 256-word block of PICTURE SYSTEM memory (177400-177777) reserved for device control registers, Matrix Arithmetic Processor registers, Maintenance registers, etc. Figure 2-4 shows the SCB and the blocks of registers contained within the SCB. The individual register specifications are described in detail throughout this chapter.

177400 : 177727	PICTURE SYSTEM 2 Device Registers
177730 : 177743	Picture Generator Maintenance and Control Registers
177744 177745	Real Time Clock Control Registers
177746	DMA 2 PS Address
177747	DMA 1 PS Address
177750 : 177757	Picture Processor Maintenance and Control Registers
177760 : 177767	System and Device Interrupt Control Registers
177770 : 177777	Passive Device Input and/or Output Ports

Figure 2-4
Register Blocks within the SCB

2.1.3 PICTURE SYSTEM Data Transfers

All data transfers to or from the PICTURE SYSTEM or from device to device within the PICTURE SYSTEM are performed by an Active device transferring data to a Passive device or by an Active device transferring data from a Passive device. An Active device is one which can initiate and control the transfer of data to or from a Passive device. A Passive device is one which can accept data from, or provide data for transfer to, an Active device. Associated with each Active/Passive data transfer is a request/acknowledge system. This system allows each word of data requested for transfer by an Active device to be transferred to or from the Passive device before the next data transfer request will be issued by the Active device. Associated with each Active device is a PICTURE SYSTEM address register which is used to address the Passive device to be used for the data transfer.

All PICTURE SYSTEM devices are either Active or Passive. Although some devices may be either Active or Passive, no device can be both Active and Passive simultaneously. Most PICTURE SYSTEM devices are Passive. An example of a Passive device is the PICTURE SYSTEM memory which waits "Passively" until requested by an Active device to transfer a word of data onto the PSBUS. The memory will transfer the data onto the bus and will then wait for the next data transfer request.

More than one Active device may address a Passive device at any given time. There are five Active devices in the standard PICTURE SYSTEM:

1. The PDP-11/PICTURE SYSTEM Direct I/O Interface
2. The PDP-11/PICTURE SYSTEM DMA I/O Interface
3. The Picture Processor MAP Input Controller
4. The Picture Processor MAP Output Formatter
5. The Picture Generator Refresh Controller

Each of these Active devices has the ability to initiate and control the transfer of data to or from a Passive device. The PICTURE SYSTEM Address Register associated with each Active device is used to address the Passive device which is the partner in the data transfer. In the case of PICTURE SYSTEM memory, this Address Register is used to indicate the location data is to be transferred to or from, and the Address Register is incremented by the Active device for the next transfer request. If the address register is addressing the Passive I/O port of a Passive device (SCB:177770-177777), then the data transfer occurs. Note that the incrementation of the Address register is inhibited so that the next data transfer request is to the same Pas-

sive I/O port.

The specific functions of each Active and Passive device will be detailed in subsequent sections of this manual.

2.2 PICTURE SYSTEM 2/PDP-11 I/O Interface

The PDP-11 computer communicates with the PICTURE SYSTEM over two I/O paths:

1. The Direct I/O (DIO) path,
2. The Direct Memory Access (DMA) path.

Using the Direct I/O path, the PDP-11 computer can transfer a single word of data to or from the PICTURE SYSTEM. This provides convenient access to the registers of the SCB from the PDP-11.

Using the DMA path, blocks of data may be transferred to or from the PICTURE SYSTEM without direct supervision by the PDP-11 CPU.

The Direct I/O and DMA input and output paths to the PICTURE SYSTEM are Active devices. Each can initiate and control the transfer of data via the control registers which reside in the PDP-11 and the SCB of the PICTURE SYSTEM. Additionally, the DMA input path to the PDP-11 may be used as a Passive device with input being directed by an Active device to the DMA Passive Input Port (SCB:177770).

The Direct I/O and DMA registers are described in the following sections.

2.2.1 Direct I/O Registers

a. PICTURE SYSTEM Data Register (PSDATA): 767660

The PSDATA is a 16-bit Read/Write register used to write (or read) data, one word at a time, to the PICTURE SYSTEM. Each word written to (or read from) the PICTURE SYSTEM is transferred to/from the PSMEM location currently addressed by the DIOPSA register (see below). Each time the PSDATA register is referenced, the DIOPSA is incremented by one. If the reference was a read, the contents of the next sequential location are then transferred to the PSDATA register. If the reference was a write, the contents of the PSDATA register are then transferred to the location specified by the DIOPSA. In either case, after the reference to the PSDATA register, the DIOPSA is incremented. This allows sequential locations of PICTURE SYSTEM memory, or the SCB, to be easily read or written. The incrementation of the DIOPSA register may be inhibited by setting the PSAHOLD bit in the IOST register (see below) or by having the DIOPSA register address one of the eight Passive I/O ports in the SCB (177770-177777). Data in this register is valid only upon completion of the Direct I/O transfer as indicated by the DIOREADY bit in the IOST register (see below). The PSDATA is a word register and should not be addressed using byte instructions.

b. DIO PICTURE SYSTEM Address Register (DIOPSA): 767662

The DIOPSA is a 16-bit write-only register. It is initially loaded with the PICTURE SYSTEM memory address to be read or written by the Direct I/O path. Thereafter, each read or write reference to the PSDATA register will cause the DIOPSA to be incremented by one. The incrementation of this register can be inhibited by setting the PSAHOLD bit in the IOST register (see below) or by addressing one of the eight Passive I/O ports in the SCB (177770-177777). The contents of the DIOPSA can only be read in the following manner:

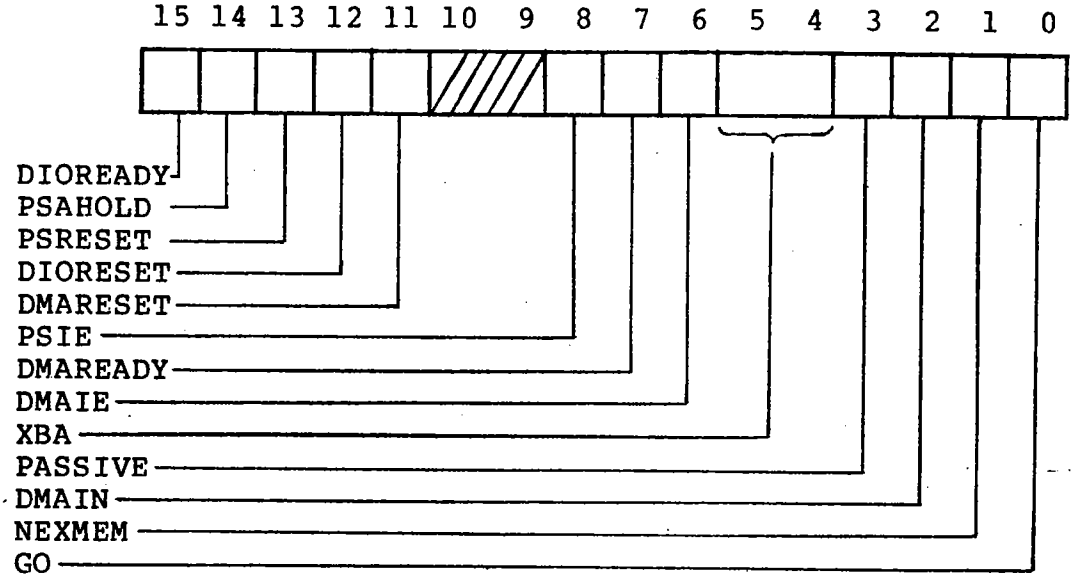
1. Read the DIOPSA register (the contents read will be undefined). This indicates that the current DIO PICTURE SYSTEM address is to be transferred to the PSDATA register.

2. Wait for the Direct I/O transfer to complete by testing the DIOREADY bit in the IOST register.

3. Read the PSDATA register. This register will now contain the location in the PICTURE SYSTEM currently being addressed by the Direct I/O path. In this case, reading PSDATA will not cause the DIOPSA to increment.

The DIOPSA is a word register and should not be addressed using byte instructions. It is cleared by PSRESET.

2.2.2 I/O Status Register (IOST): 767670



This register is used to provide status indicators of the Direct I/O and DMA I/O operations, to reset the PICTURE SYSTEM and the I/O channel to an initial state and to initiate a DMA transfer to or from the PICTURE SYSTEM. This register is not byte addressable and must be addressed using word instructions.

BIT	NAME	FUNCTION
15	DIOREADY	Cleared upon initiation of a Direct I/O transfer (i.e., any reference to the PSDATA or DIOPSA registers). Set to indicate that the Direct I/O channel is ready to perform another operation and that the previous operation has completed. Set by INIT, PSRESET or DIORESET. Read only.
14	PSAHOLD	Set to inhibit incrementing of the DIOPSA register so that subsequent data transferred to/from the PICTURE SYSTEM is to/from the same PICTURE SYSTEM memory location. Cleared by INIT or PSRESET. Read/Write.

13	PSRESET	When set, causes a pulse to the PICTURE SYSTEM which resets all PICTURE SYSTEM registers and I/O processes to their initial state. This provides a mechanism for initializing the PICTURE SYSTEM. Setting PSRESET causes a RESET pulse to be issued to the PICTURE SYSTEM (equivalent to the RESET button on the backpanel) and also performs the functions of DIORESET and DMARESET. Write only. Always reads as a zero.
12	DIORESET	When set, causes all Direct I/O processes to and from the PICTURE SYSTEM to be reset to their initial state. This is used to initialize the Direct I/O processes. Write only. Always reads as a zero.
11	DMARESET	When set, causes all Direct Memory Access I/O processes to and from the PICTURE SYSTEM to be reset to their initial state. This is used to initialize the DMA I/O processes. Write only. Always reads as a zero.
10-9	not used	
8	PSIE	PSIE (PICTURE SYSTEM Interrupt Enable) is set to allow any interrupt request from the PICTURE SYSTEM to cause an interrupt at the PDP-11. This bit is used in conjunction with the interrupt enable registers of the PICTURE SYSTEM (see Section 2.5, Interrupt Control, for details). Cleared by INIT or PSRESET. Read/Write.
7	DMAREADY	Set to indicate that the DMA has completed the last transfer and is ready to accept a new command. Set by INIT, PSRESET, DMARESET or

		NEXMEM; cleared by GO. Set on word count overflow of the DMAWC register (177777 => 000000). Causes an interrupt if bit 6 (IE) is set. Forces the DMA to release control of the UNIBUS and prevents further DMA cycles. Read only.
6	DMAIE	DMAIE (DMA Interrupt Enable) is set to allow DMAREADY or NEXMEM = 1 to cause an interrupt. Cleared by INIT or PSRESET. Read/Write.
5-4	XBA	XBA (Extended Bus Address) specifies the UNIBUS address bits 17 and 16 that, in conjunction with DMABA, form the 18-bit address to be used for direct memory transfers. Cleared by INIT or PSRESET. XBA17 and XBA16 increment when DMABA overflows. Read/Write.
3	PASSIVE	This bit is used to set the DMA into a mode of operation where it will function as a Passive device for transfer of data from the PICTURE SYSTEM to the PDP-11. All data directed to the DMA Passive Input Port register in the SCB will be transferred to the PDP-11 and stored at the current address specified by DMABA and XBA<17:16>. The DMA can function as a Passive device if data are to be transferred back to the PDP-11. Before the passive input transfer can be performed, the number of words to be transferred (word count) and the base address in PDP-11 memory must be set in the DMAWC and DMABA registers and the transfer initiated by setting the GO bit. Cleared by INIT or PSRESET. Read/Write.
2	DMAIN	DMAIN (DMA Input) is used to indicate the current transfer

		direction of the DMA. Cleared by INIT or PSRESET. Read/Write. If DMAIN=0, the data transfer direction is out--from the PDP-11 to the PICTURE SYSTEM. If DMAIN=1, the data transfer is in--from the PICTURE SYSTEM to the the PDP-11.
1	NEXMEM	NEXMEM (Nonexistent Memory) is set to indicate that a UNIBUS master, the DMA, did not receive a SSYN response 10 usec after asserting MSYN. Cleared by INIT, PSRESET, DMARESET or loading with a 0; cannot be loaded with a 1.
0	GO	This bit is set to initiate the DMA transfer. Setting this bit clears DMAREADY. Note that no other bits in this word are able to be modified by the same instruction which causes this bit to be set. Write only. Always reads as a 0.

2.2.3 Direct Memory Access Registers

This section deals with the mechanism used to pass blocks of data between the PDP-11 and the PICTURE SYSTEM. Note that blocks of data can be transferred in either direction.

The block data transfer path is a Direct Memory Access interface unit. To pass data to or from the PICTURE SYSTEM, a block of PDP-11 memory containing the data, or which will receive the data, is specified by loading registers in the DMA interface. The location in the PICTURE SYSTEM to or from which data is to be transferred must also be specified by loading the DMAPSA (see below).

a. DMA Word Count Register (DMAWC): 767664

The DMAWC is a 16-bit register. It is initially loaded with the two's complement of the number of words to be transferred and increments up towards zero after each bus cycle. When overflow occurs (all 1's to all 0's), the DMA READY bit of the IOST is set and the bus cycles stop. DMAWC is a word register; byte instructions should not be used when loading this register. Cleared by INIT or PSRESET.

b. DMA Bus Address Register (DMABA): 767666

The DMABA is a 15-bit R/W register. Bit 0 is always a zero, and is a read-only bit. Along with bits 5 and 4 of the IOST (XBA17 and XBA16), the DMABA is used to specify the address used when the DMA accesses the UNIBUS. The register is incremented (by 2) after each bus access, advancing the address to the next sequential word location on the bus. If DMABA overflows (177776 to 0), the bus extension bits of the IOST are incremented and the transfer continues. DMABA is a word register; byte instructions should not be used when loading this register. Cleared by INIT or PSRESET.

c. DMA PICTURE SYSTEM Address Register (DMAPSA)-SCB:177747

The DMA PICTURE SYSTEM Address Register is a 16-bit R/W register in the PICTURE SYSTEM SCB, which indicates the next word of PICTURE SYSTEM memory to be accessed, whether as a read or write or an input to, or output from, the memory. After each word is transferred, this address is then incremented so that the next word transferred will be to (or from) the next sequential

location. If the DMAPSA is currently addressing one of the eight Passive I/O ports in the SCB (177770-177777), incrementation of the DMAPSA will be inhibited. Cleared by PSRESET.

d. DMA Passive Input Port (DMAPIP)-SCB:177770

The DMA Passive Input Port is a 16-bit write-only location in the PICTURE SYSTEM SCB to which data may be directed by an Active device for DMA transfer to the PDP-11. In order to use the DMAPIP correctly, the DMA must be set up and initiated in the PDP-11 and the Active device's address register should then be set to address the DMAPIP using the Direct I/O path to the PICTURE SYSTEM. Once this has been done, all subsequent data output by the Active device will be transferred to the PDP-11 without direct intervention by the CPU until the entire block of data which was set up to be transferred has been completed.

2.3 The Picture Processor

The Picture Processor is a microprogrammed, high-speed, digital arithmetic processor capable of accepting two-, three-, or four-dimensional data, transforming the data by a previously loaded or concatenated Transformation Matrix, clipping the transformed data at the boundaries of a six-sided window, performing a perspective division and viewport mapping on the clipped data, and outputting the transformed, clipped and viewport mapped data to the refresh buffer as a portion of a segmented display file for subsequent display on a CRT. There are three basic units of the Picture Processor:

1. The MAP Input Controller
2. The Matrix Arithmetic Processor (MAP)
3. The MAP Output Formatter

These units function together in a synchronous pipelined manner to provide maximum throughput for the Picture Processor.

The Picture Processor is microprogrammed to perform the functions described throughout this section. The Picture Processor is, however, capable of being provided as a general-purpose, microprogrammable digital processor. When used in this manner, the microprogram is loaded, using the Picture Processor Maintenance Registers described in Section 2.3.4.

The Picture Processor input, output and status is controlled by eight registers in the SCB. These registers, shown in Figure 2-5, are described in the following sections which detail the three basic units of the Picture Processor.

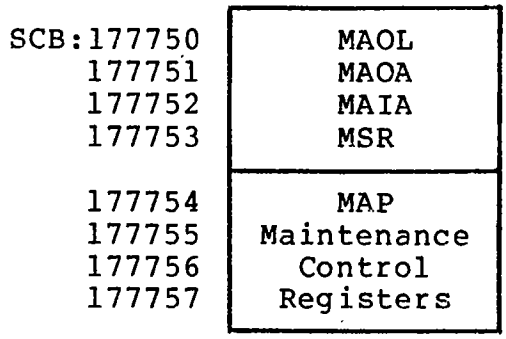
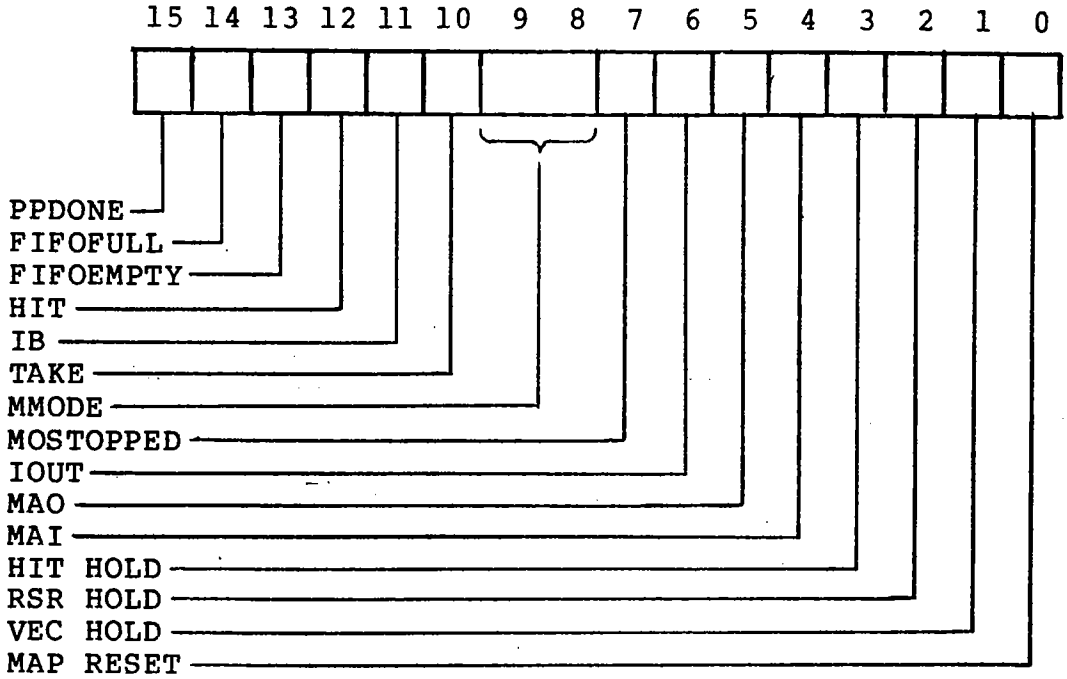


Figure 2-5
Picture Processor I/O Control, Status and Maintenance Registers

MAP Status Register (MSR)-SCB:177753



The MSR is the basic control register of the Picture Processor. It is used to provide operating mode information to the MAP, and status of the MAP back to the control program.

BIT	NAME	FUNCTION
15	PPDONE	PPDONE (Picture Processor Done) is set to indicate that the MAP Output Formatter is finished with its processing and the MAP itself is in the HOLD state waiting for another processing request. Read only. Set by RESET.
14	FIFOFULL	When this bit is set, it indicates that the MAP input FIFO is full and has 4 words of data buffered for input to the MAP. Read only. Cleared by RESET.
13	FIFOEMPTY	When this bit is set, it indicates that the MAP input FIFO is empty. Read only. Set by

RESET. The MAP input FIFO is emptied (and this bit is set) by loading the MAP Active Input Address register.

12 HIT

This bit is cleared at the beginning of the processing of each input vector. While processing, if any part of the vector is within the window, HIT is set. If the HIT Interrupt Enable (see Section 2.5.2) is set, this bit will cause an interrupt to occur. Read only unless the write is enabled by the TAKE bit (10) being set. Cleared by RESET.

11 IB

IB is used to designate where the INPUT, SAVE, BASE and MOUT x,y,z and w coordinates are located within the MAP Register File. Each time a new vector is input, IB is complemented. INPUT, SAVE, BASE and MOUT X,Y,Z and W are valid only in between and after successive vector commands (i.e., a Matrix concatenation destroys the current INPUT, BASE, SAVE and MOUT X,Y,Z and W coordinates.) IB is cleared by RESET and can be set or cleared from software.

If IB=0 before INPUT then:

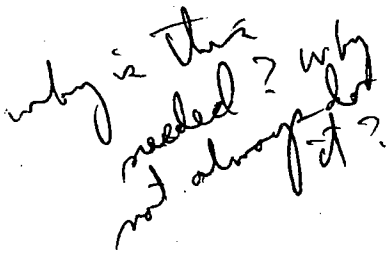
ABSOLUTE -
Data input written into MAP 0-3.
RELATIVE -
Previous information in MAP 4-7. New input plus previous written into MAP 0-3.
ORIGIN OFFSET -
BASE information in MAP 0-3. New input plus BASE written into MAP 0-3. BASE information moved to MAP 4-7.
SET BASE -
Data input written into MAP 0-3 and into MAP 4-7.

If IB=1 before INPUT then:

ABSOLUTE -
Data input written into MAP 4-7.
RELATIVE -
Previous information in MAP 0-3. New input plus previous written into MAP 4-7.
ORIGIN OFFSET -
BASE information in MAP 4-7. New input plus BASE written into MAP 4-7. BASE information moved to MAP 0-3.
SET BASE -
Data input written into MAP 0-3 and into MAP 4-7.

Read only unless the write is enabled by the TAKE bit (10) being set. Cleared by RESET.

10 TAKE



This bit is used to enable the writing of bits 11 and 12 (IB and HIT). If this bit is set when the MSR is written, then bits 11 and 12 are written into the register; otherwise bits 11 and 12 of the MSR remain unmodified. Write-only. Always reads as a zero.

9,8 MMODE

The MMODE (MAP Mode) bits are used to control the mode in which the MAP processes data and the Output Formatter outputs the data. Set to 00 by RESET. The bits indicate the mode as follows:

00

Display Mode
In this mode, all two-, three-, or four-dimensional data are transformed, normalized, clipped, viewport mapped and output formatted for input to the Picture Generator. All Pass Formatted data will be formatted for input to the Picture Generator.

01

Sixteen-Bit Precision Mode
In this mode, all two-, three-, or four-dimensional data are transformed, normalized, clipped, viewport mapped and formatted as a linear display list with an RSR (see Section 2.3.2.2) for each set of X,Y,Z and W coordinates output. All data that are to be merely passed through the MAP are output unmodified. All Control Commands are executed by the MAP and the data output according to the command. All data are output as absolute 3D data (i.e., X,Y,Z). All "Pass" data are output in 3D Pass with only the first and second words being valid (i.e., the words corresponding to the X and Y coordinates).

10

Transform/Normalize Mode
In this mode, all two-, three-, or four-dimensional data are transformed and normalized only and output as X,Y,Z and W coordinates--each significant to 16-bits. The clipping, perspective division and viewport mapping is not performed upon data when in this mode. All other forms of data (Pass, etc.) should not be used when in this mode.

11

Transform Only Mode
In this mode, all two-, three-, or four-dimensional data are transformed only and output as X,Y,Z and W coordinates--each significant to 16-bits. The clipping, perspective division and viewport mapping is not performed upon data when in this mode. All other forms of data (Pass, etc.) should not be used when in this mode. It should be noted while in this mode it is possible to have the matrix

		multiplication produce overflow such that the data output by the MAP as 16-bits does not represent the same number kept internally in the MAP to 24 bits of precision. This overflow can be read and compensated for by the control program in the host computer.
7	MOSTOPPED	When MAP Output Stopped (MOSTOPPED) is set, it indicates that output of the MAP has stopped since the MAP Active Output Address register is equal to the MAP Active Output Limit register. This condition indicates that the area of PICTURE SYSTEM memory allocated for output data (i.e., refresh buffer, etc.) is full. The condition can only be dismissed by resetting either the MAP Active Output Address (MAOA) or the MAP Active Output Limit (MAOL) register. The setting of this bit will cause an interrupt to the host computer if the MOSTOP interrupt enable bit is set in the System Interrupt Enable register (see Section 2.5.2). Read only. Cleared by resetting MAOA or MAOL so that MAOA is not equal to MAOL. Set by MAOA=MAOL and RESET.
6	IOUT	When Inhibit Output (IOUT) is set, it inhibits the MAP Output Formatter from outputting any further data. Input to the MAP may proceed as if output were not inhibited. This bit is useful in "hit testing" where data passing through a hit window would appear misplaced if output by the Picture Processor for display. Cleared by RESET.
5	MAO	When the MAP Active Output (MAO) bit is set, the MAP Output Formatter functions as an

		Active device, initiating and controlling the output of data from the MAP to the Passive device currently selected by the MAP Active Output Address register--typically, PICTURE SYSTEM memory. Cleared by RESET.
4	MAI	When the MAP Active Input (MAI) bit is set, the MAP Input Controller functions as an Active device filling the MAP input FIFO with data from the Passive device currently selected by the MAP Active Input Address register. Clearing this bit is the only way in which to halt the MAP Input Controller. Cleared by RESET.
3	HIT HOLD	When this bit is set, the MAP will stop the processing of new data after a hit occurs, as indicated by the HIT bit (12). Processing of new data by the MAP can be continued only by clearing the HIT Request bit in the System Request register (see Section 2.5.2). Cleared by RESET.
2	RSR HOLD	When this bit is set, the MAP will stop taking data from the input FIFO after execution of the current MAP RSR command has completed. The next RSR command to be processed will be the first item in the FIFO, if any data is in the FIFO. After this bit is set, PPDONE (bit 15) indicates that the Picture Processor has completed the last RSR operation and is now in the HOLD state. Cleared by RESET.
1	VEC HOLD	When this bit is set, the MAP will stop taking data from the input FIFO when processing for the current vector is complete. This is true unless the current vector is the last

		vector of an RSR command. In this case, the first associated vector of the next RSR command will also be processed before the MAP holds. After this bit is set, PPDONE (bit 15) indicates that the Picture Processor has completed the last vector and is now in the HOLD state. Cleared by RESET.
0	MAP RESET	When this bit is set, it causes a RESET signal to be issued to the MAP, resetting it to its initial power-up state. Write only. Always reads as a zero.

2.3.1 The MAP Input Controller

The MAP Input Controller is a PICTURE SYSTEM device which can function as either an Active or Passive device. When functioning as either, the operation of the Input Controller is comparatively simple--it transfers 16-bit words of data, either actively or passively received, to a four-word FIFO (First In First Out) buffer. As the FIFO becomes full, the Input Controller waits for the MAP to empty a word of data from the FIFO before the next word of data is transferred to the FIFO. There are two status indicators of the current state of the MAP input FIFO. These indicators are "FIFO FULL" and "FIFO EMPTY" and exist as bits in the MAP Status Register.

As previously stated, the MAP Input Controller can function as either an Active or Passive device. The Active/Passive state of the Input Controller is selected by a bit in the MAP Status Register (MSR). When the MAP Input Controller is selected as an Active device, there is an associated MAP Active Input Address register (MAIA) which holds the address of the location in PSMEM from which the next word of data to be sent to the MAP Input FIFO is taken.

a. MAP Active Input Address (MAIA)-SCB:177752

The MAIA is a 16-bit R/W register which holds the PSMEM address to be used to fetch the next word of data for input to the MAP. After each word of data is transferred to the MAP Input FIFO, this address is incremented by the MAP Input Controller so that the next word of data is taken from the next sequential address of PICTURE SYSTEM memory. This register should not be used to address any Passive device other than PICTURE SYSTEM memory. Incrementation of this device cannot be inhibited by addressing one of the eight Passive I/O Ports (SCB:177770-177777).

If the MAP Input Controller is not selected as an Active device, it is, therefore, a Passive device and all data directed to its Passive Input Port (MPIP, SCB:177777) is stored in the MAP input FIFO for input to the MAP.

b. MAP Passive Input Port (MPIP)-SCB:177777

The MAP Passive Input Port is a 16-bit write-only location in the PICTURE SYSTEM SCB to which data may be directed by an Active device for input to the MAP. In order to use the MPIP, the Active device's output ad-

dress register should be set to address the MPIP using the Direct I/O path to the PICTURE SYSTEM and the MAP Input Controller selected for Passive input. Once this has been done, all subsequent data output by the Active device will be transferred to the MAP without direct intervention by the host CPU.

2.3.2 The Matrix Arithmetic Processor (MAP)

The Matrix Arithmetic Processor consists of a Transformation Matrix, Transformation Matrix Stack, Parameter Register File, and Arithmetic Unit.

The Transformation Matrix is a 4x4 element matrix, where each element is a 16-bit word. This 4x4 matrix is used to tranform object coordinate data. It can also be concatenated with other 4x4 matrices to obtain a combined transformation.

The Transformation Matrix Stack is a storage area where up to fourteen 4x4 element matrices may be "stacked" or saved for future recall.

The Parameter Register File is an array of registers into which parameters specifying viewport boundaries, scale factors, etc. are stored and may be retrieved.

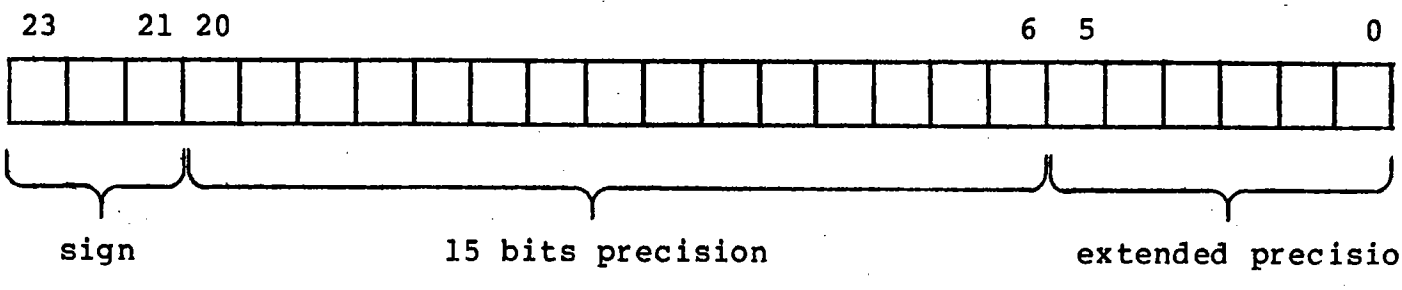
The Arithmetic Unit performs all scalar, vector and matrix arithmetic operations in the Picture Processor. This includes subtraction, addition, multiplication, division and normalization.

The MAP utilizes these units to perform digital operations on the data received from the host computer.

2.3.2.1 MAP Internal Registers

This secton describes the 256 registers that are internal to the MAP and are used to contain the Transformation Matrix, Matrix Stack, parameters for various functions, and working storage during command execution. Figure 2-6 shows these registers and the addresses assigned to them. These addresses are specified using the "Operand" field of the RSR Control Commands (see Section 2.3.2.2).

Each of the 256 MAP registers are 24-bit registers of the following format:



The high three bits (23-21) represent the sign of the number. The lower 21 bits (20-0) are used to represent the number to 21 bits of significance. When data is input to or output from the MAP, it is transferred as 16 bits (1 bit sign, 15 bits data) to or from bits 21-6 of the specified MAP register. When the transfer is from a MAP register, bits 21-6 are transferred. When the transfer is to a MAP register, bits 5-0 are loaded with 0, bits 21-6 are loaded with the 16-bits of data transferred to the MAP, bits 23-22 are sign extended from bit 21. All 24 bits of each of the 256 MAP registers may be LOADed or STOREd by using the extended LOAD and STORE commands (see Section 2.3.2.2).

It should be noted that all MAP arithmetic operations are performed treating data as two's complement signed (fixed point) fractions. Since the basic word length of PICTURE SYSTEM 2 is 16 bits, the algebraically largest number that can be represented is 1-2**15 (i.e., 32767 decimal or 77777 octal), and the algebraically smallest number that can be represented is -1 (-32768 decimal or 100000 octal). In binary notation (with the binary point separating the sign bit from the fraction):

```
0.111111...  is the algebraically largest number.
0.000000...  is the unigue representation for zero.
1.000000...  is the algebraically smallest number.
```

The closest approximation to +1 is the fraction 0.111111..., which is close enough to +1 for practical cases. Thus, the MAP internal representation of 24 bits is an extention of the 16-bit two's complement fractions described above. Three bits of sign (23-21) are used since during matrix multiplication four numbers are summed together, which can require up to 3 bits to maintain the correct sign of the summation. Normalization is always performed to the binary point.

0	INPUTx/BASEx
1	INPUTy/BASEy
2	INPUTz/BASEz
3	INPUTw/BASEw
4	BASEx/INPUTx
5	BASEy/INPUTy
6	BASEz/INPUTz
7	BASEw/INPUTw
10	MOUTx/SAVEx
11	MOUTy/SAVEy
12	MOUTz/SAVEz
13	MOUTw/SAVEw
14	SAVEx/MOUTx
15	SAVEy/MOUTy
16	SAVEz/MOUTz
17	SAVEw/MOUTw
20	Viewport X 1/2 Size
21	Viewport X Center
22	Viewport Y 1/2 Size
23	Viewport Y Center
24	Viewport Z Size
25	Viewport Z Front
26	Working Register
27	Transformation Matrix Address
30	NEWCLIPx/CLIPSAVEx
31	NEWCLIPy/CLIPSAVEy
32	NEWCLIPz/CLIPSAVEz
33	NEWCLIPw/CLIPSAVEw
34	CLIPSAVEx/NEWCLIPx
35	CLIPSAVEy/NEWCLIPy
36	CLIPSAVEz/NEWCLIPz
37	CLIPSAVEw/NEWCLIPw
40	Transformation Matrix
:	and Matrix Stack for
377	14 4x4 Matrices

Figure 2-6
MAP Parameter Register File

In Figure 2-6, the "/" designation (i.e., INPUT/BASE) is used to indicate that the particular MAP register may be used for either the INPUT or BASE register depending upon the state of the IB bit in the MAP Status Register. In all of the "toggled" registers (i.e., INPUTx/BASEx) the first register name (INPUTx) is selected when IB=0 and the second name (BASEx) is selected when IB=1. Following is a description of each register and its usage.

INPUT (INPUT), Registers 0-3 or 4-7

The INPUTx, INPUTy, INPUTz and INPUTw registers are used to hold the X,Y,Z and W data input to the MAP for 2D, 3D or 4D DRAW commands. Alternates with BASE.

BASE (BASE), Registers 4-7 or 0-3

The BASEx, BASEy, BASEz and BASEw registers are used to hold the current X,Y,Z and W base coordinates for 2D, 3D or 4D DRAW commands. For Relative and Origin Offset draws, the BASE register is used to compute the absolute coordinates for the input data. Alternates with INPUT.

Matrix Output (MOUT), Registers 10-13 or 14-17

The MOUTx, MOUTy, MOUTz and MOUTw registers are used to hold the normalized matrix output X,Y,Z and W coordinates at the completion of a 2DDRAW, 3DDRAW or 4DDRAW command execution. MOUT contains the data as it exists after it has been multiplied by the Transformation Matrix, but prior to any clipping that may have taken place. Alternates with SAVE.

SAVE (SAVE), Registers 14-17 or 10-13

The SAVEx, SAVEy, SAVEz and SAVEw registers are used to hold the unclipped X,Y,Z and W coordinates of the previous point. The unclipped coordinates are necessary to properly compute the next intersection of the clipping plane if required. Alternates with MOUT.

Viewport X Half-Size (VIEWXH), Register 20

This register is used to specify the half-width of the X viewport boundaries to which data that lies within the clipping boundaries will be mapped by the viewport mapping process. This value is typically 0-2K, causing the data to be mapped to the boundaries of the Picture Display. If a negative value is placed here, the mapping will be such that a mirror image is produced about the y-axis of the viewport.

Viewport X Center (VIEWXC), Register 21

This register is used to specify the center of the viewport in the x-axis. This value, in conjunction

with VIEWXH, is used to specify the left and right viewport boundaries. The boundaries are computed as follows:

```
Viewport Left boundary  = VIEWXC - VIEWXH
Viewport Right boundary = VIEWXC + VIEWXH
```

Viewport Y Half-Size (VIEWYH), Register 22

This register is used to specify the half-width of the Y viewport boundaries to which data that lies within the clipping boundaries will be mapped by the viewport mapping process. This value is typically 0-2K, causing the data to be mapped to the boundaries of the Picture Display. If a negative value is placed here, the mapping will be such that a mirror image is produced about the x-axis of the viewport.

Viewport Y Center (VIEWYC), Register 23

This register is used to specify the center of the viewport in the y-axis. This value, in conjunction with VIEWYH, is used to specify the bottom and top viewport boundaries. The boundaries are computed as follows:

```
Viewport Bottom boundary = VIEWYC - VIEWYH
Viewport Top boundary    = VIEWYC + VIEWYH
```

Viewport Z Size (VIEWZS), Register 24

This register is used to specify the size of the Z viewport boundaries to which data that lies within the clipping boundaries will be mapped by the viewport mapping process. This value is typically 63-0, causing the data to be mapped so that it may be displayed as depth-cued information.

Viewport Z Front (VIEWZF), Register 25

This register is used to specify the front of the viewport in the Z-axis. This value, in conjunction with VIEWZS, is used to specify the hither and yon viewport intensities. The intensities are computed as follows:

```
Viewport Hither intensity = VIEWZF
Viewport Yon intensity    = VIEWZF + VIEWZS
```

Working Register, Register 26

This register is used as a working register whose contents are undefined at any given time.

Transformation Matrix Address (TMADR), Register 27

This register is used to contain the current MAP register address (0-377) which is to be used as the Transformation Matrix. This register is used as a Matrix Stack pointer in that a Matrix PUSH (or Matrix POP) causes the address to increment (or decrement) by 16. This register always contains the address of the 16th element of the current Transformation Matrix.

New Clip (NC), Registers 30-33 or 34-37

The NCx, NCy, NCz and NCw registers are used to contain the clipped X,Y,Z and W coordinates of the new clipped point if the point required clipping as indicated by the SV and NV bits of the MAP Maintenance Status Register (see Section 2.3.4). Alternates with CSAVE.

Clip Save (CSAVE), Registers 34-37 or 30-33

The CSAVEx, CSAVEy, CSAVEz, CSAVEw registers are used to contain the clipped X,Y,Z and W coordinates of the previous point. Alternates with NC.

Transformation Matrix (TM), Registers 40-377

The Transformation Matrix is used to contain the 4x4 matrix by which all 2D, 3D and 4D coordinates are transformed. The Transformation Matrtix may be any 16 consecutive MAP registers from 40-377. The Transformation Matrix must, however, begin on an even 16-word boundary, thus restricting its beginning address to MAP register 40, 60, 100, 120...360. The address of the current Transformation Matrix is always indicated by the Transformation Matrix Address (TMADR), register 27. Matrices are stacked in the PICTURE SYSTEM by copying the current contents of the TM to TM+16 and then incrementing TMADR by 16. See Section 2.3.2.2 for details of the commands which modify the TM. Although allocated as Matrix Stack space, MAP registers 40-377 need not be used only for matrix storage, but may also be used to store other data or as auxillary (not Matrix) stack area (see Section 2.3.2.2).

2.3.2.2 Picture Processor Commands

In order to properly process data sent to the Picture Processor, a command must be sent to the MAP indicating the operation to be performed. The Repeat Status Register (MMRSR-SCB:177755) is used to supply commands to the Picture Processor. It is called a "Repeat" Status Register since portions of its contents may be automatically modified in a predetermined manner after the specified command has been executed. In certain cases, the command is repeated after the modification without required program intervention. The basic format of the RSR command is shown in Figure 2-7.

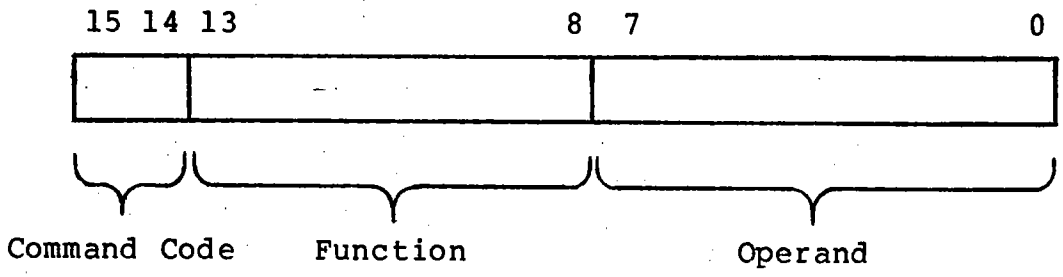


Figure 2-7
RSR Basic Command Format

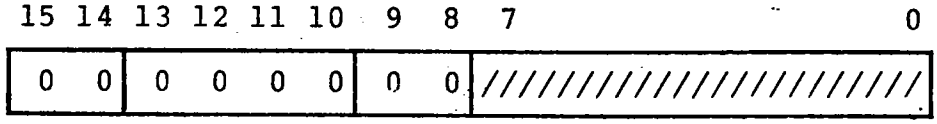
As the figure shows, the basic command format consists of 2 bits of command code (15,14), 6 bits of a Function Field, and 8 bits of an Operand Field. The meaning of the function and operand fields are dependent upon the command currently being processed. There are two basic command codes:

a. RSR Control Commands
b. RSR Drawing Commands

These Command Codes and the associated Function and Operand fields are described in detail in the following sections.

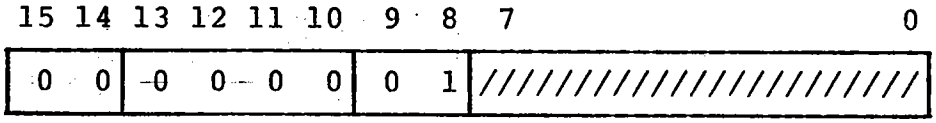
a. RSR Control Commands

HALT



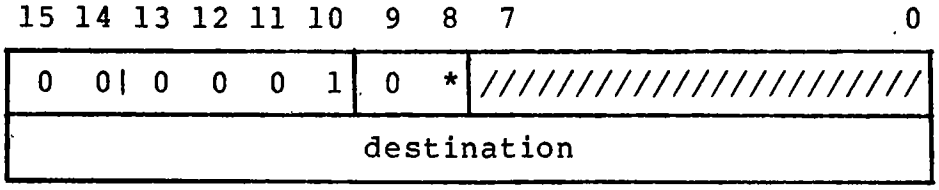
HALT causes the MAP input operation to cease, setting the PPDONE bit (15) in the MAP Status Register (MSR) when the MAP Output Formatter has completed its last operation.

NO-OP



NO-OP causes the next RSR command to be input to the MAP without other action being taken.

JUMP



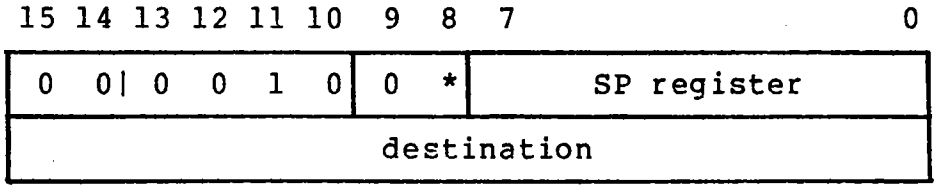
JUMP causes the next word of data input to the MAP to be placed in the MAP Active Input Address (MAIA) register, thereby causing all subsequent input to come from the effective PICTURE SYSTEM memory address specified by the destination word. The MAP must be in operation as an Active device for any further data to be transferred from PSMEM. After the PICTURE SYSTEM memory address is transferred to the MAIA, the input FIFO is cleared so that the next data processed by the MAP is from the PSMEM location specified. Note that the PICTURE SYSTEM memory address specified should contain the next RSR command to be processed. If this command is executed by the MAP when the MAP Input Controller is not selected as an Active device, the JUMP REQ bit is set in the System Interrupt Request register (SCB:177762). This will cause an interrupt to the host computer if the interrupt is currently enabled in the System Interrupt Enable register (SCB:177763). The MAP will then be in the HOLD state with the PPDONE bit (15) in the MSR set.

Bit 8 of the RSR JUMP command is used to determine whether the JUMP is to an absolute address in PICTURE SYSTEM memory (Absolute) or to a location relative to the current MAP Active Input Address (Relative).

If bit 8=0: The JUMP is performed as an absolute JUMP to the address specified by destination.

If bit 8=1: The JUMP is performed as a relative JUMP to the effective address computed as MAIA + destination.

PUSHJ



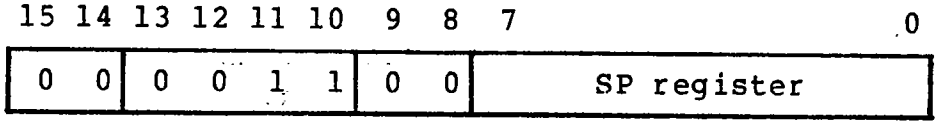
PUSHJ functions exactly as the RSR JUMP command except that the current address in the MAIA is saved in the stack whose stack pointer is specified in the operand field (SP register) of the PUSHJ command. The MAP register being addressed by the operand as the stack pointer is incremented before the current MAIA is saved.

Bit 8 of the RSR PUSHJ command is used to determine whether the JUMP, performed after the current address is pushed onto the stack, is to an absolute address in PICTURE SYSTEM memory (Absolute) or to a location relative to the current MAP Active Input Address (Relative).

If bit 8=0: the PUSHJ is performed as an absolute PUSHJ to the address specified by destination.

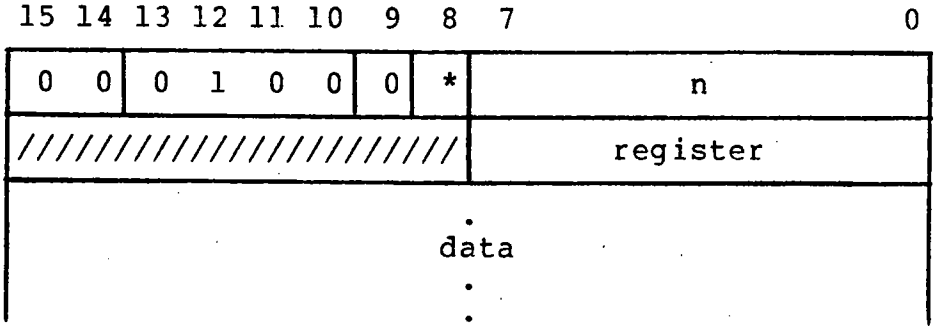
If bit 8=1: The PUSHJ is performed as a relative PUSHJ to the effective address computed as MAIA + destination.

POPJ



POPJ causes the current contents of the top of the stack, whose stack pointer is specified in the operand field (SP register), to be placed in the MAIA register . This causes all subsequent input to come from the PICTURE SYSTEM memory address that was on the top of the stack. The MAP must be in operation as an Active device for any further data to be transferred from PSMEM. After the PICTURE SYSTEM memory address is transferred to the MAIA, the input FIFO is cleared so that the next data processed by the MAP is from the PSMEM location taken from the top of the stack. The MAP register being addressed as the stack pointer is decremented after the MAIA is loaded with the top of the stack. If this command is executed by the MAP when the MAP Input Controller is not selected as an Active device, the JUMP REQ bit is set in the System Interrupt Request register (SCB:177762). This will cause an interrupt to the host computer if the interrupt is currently enabled in the System Interrupt Enable register (SCB:177763). The MAP will then be in the HOLD state with the PPDONE bit (15) in the MSR set.

LOAD



The LOAD command is used to load the 256 MAP registers with data. The operand field of the command (n) specifies the two's complement of the number of MAP registers to be loaded. The next word of data specifies the first MAP register to be loaded. The data which follows is then sequentially loaded into the MAP registers.

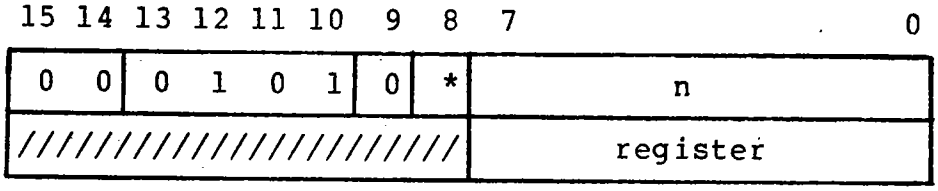
If bit 8=0: Each word of data is loaded into each 24-bit MAP register as bits 21-6 of the register. Bits 23-22 are sign extended from the sign of the 16-bit word (bit 21) which was loaded. Bits 5-0 are zeroed.

If bit 8=1: Each 24-bit MAP register is loaded with 24-bits of data taken from the data input in the following manner:

MAP register bit 23←data word i bit 1.
MAP register bit 22←data word i bit 0.
MAP register <21-6>←data word i+1 <15-0>.
MAP register <5-0>←data word i <15-10>.

In this way, 32 bits of data are transferred for each 24-bit MAP register. Bits 9-2 of data word i are discarded.

STORE

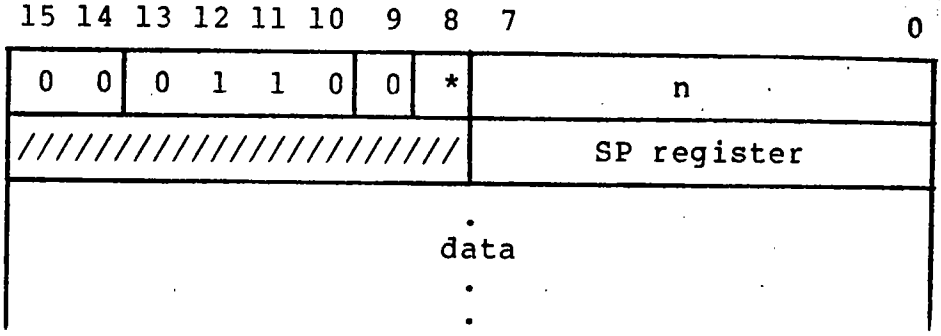


The STORE command is used to store the contents of the 256 MAP registers. The operand field of the command (n) specifies the two's complement of the number of MAP registers to be stored. The next word of data specifies the first MAP register to be stored. The contents of the MAP registers are sequentially transferred to the location specified by the MAP Active Output Address (MAOA) register. Bit 8 of the RSR specifies how the data are to be taken from the MAP registers:

If bit 8=0: Bits 21-6 of each MAP register selected are transferred to the location specified by the MAOA. Note that in this mode, the data stored may not truly reflect the contents of the register. This is especially true if the register contains unnormalized matrix data.

If bit 8=1: Bits 23-22 and 5-0 of each MAP register selected are transferred to bits 1-0 and 15-10 of the location specified by the MACA. Bits 21-6 of each MAP register are transferred to bits 15-0 of the location specified by MAOA+1.

LOADSTK



The LOADSTK (Load Stack) command is used to load data into a stack area of the 256 MAP registers. The operand field of the command (n) specifies the two's complement of the number of MAP registers to be loaded. The next word of data (SP register) specifies the MAP register which contains the MAP address of the current top of the stack. The data which follows are then loaded into the top n words of the stack within the MAP. The stack pointer is unmodified by this operation.

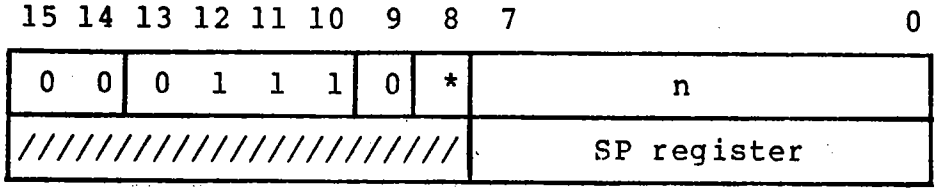
If bit 8=0: Each word of data is loaded into each 24-bit MAP register as bits 21-6 of the register. Bits 23-22 are sign extended from the sign of the 16-bit word (bit 21) which was loaded. Bits 5-0 are zeroed.

If bit 8=1: Each 24-bit MAP register is loaded with 24-bits of data taken from the data input in the following manner:

MAP register bit 23←data word i bit 1.
MAP register bit 22←data word i bit 0.
MAP register <21-6>←data word i+1 <15-0>.
MAP register <5-0>←data word i <15-10>.

In this way, 32 bits of data are transferred for each 24-bit MAP register. Bits 9-2 of data word i are discarded.

STORESTK

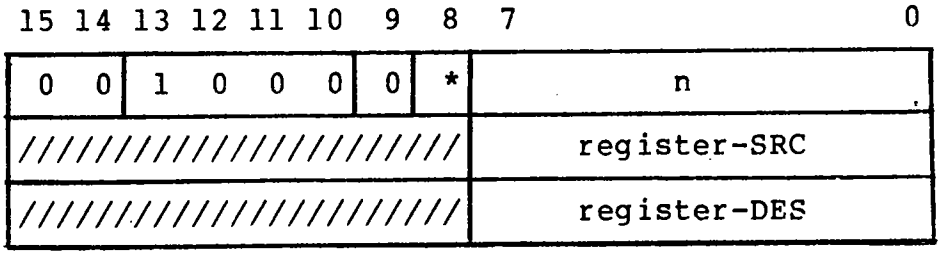


The STORESTK (Store Stack) command is used to store data from a stack area of the 256 MAP registers. The operand field of the command (n) specifies the two's complement of the number of MAP registers to be stored. The next word of data (SP register) specifies the MAP register which contains the MAP address of the current top of the stack. The n MAP registers on top of the stack are then output by the MAP. The stack pointer is unmodified by this operation.

If bit 8=0: Bits 21-6 of each MAP register selected are transferred to the location specified by the MAOA. Note that in this mode, the data stored may not truly reflect the contents of the register. This is especially true if the register contains unnormalized matrix data.

If bit 8=1: Bits 23-22 and 5-0 of each MAP register selected are transferred to bits 1-0 and 15-10 of the location specified by the MAOA. Bits 21-6 of each MAP register are transferred to bits 15-0 of the location specified by MAOA+1.

XFER

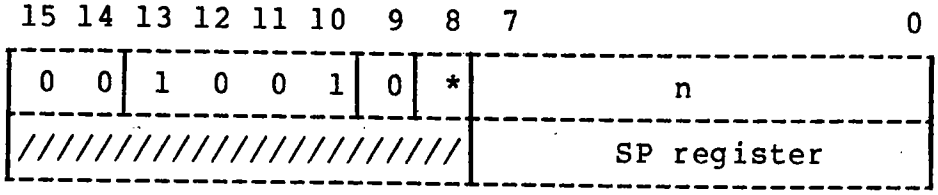


The XFER (Transfer) command is used to transfer the contents of the MAP registers from one location to another. The operand field of the command (n) specifies the two's complement of the number of MAP registers to be transferred. The next word of data (register-SRC) specifies the first MAP register to be transferred. The next word of data (register-DES) specifies the first MAP register to which data are to be transferred.

If bit 8=0: The transfer operation is conducted as a move. The contents of n sequential MAP registers are moved, beginning from the source specified to the n sequential locations which begin at the destination register specified.

If bit 8=1: The transfer operation is conducted as a swap. The contents of n sequential MAP registers, beginning from the source register specified, are exchanged with the contents of the n sequential registers which begin with the specified destination register.

PUSH

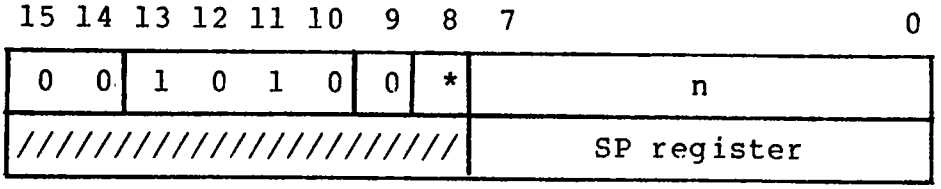


The PUSH command is used to transfer the contents of the top n words of a stack area of the 256 MAP registers onto the top of the selected stack. The operand field of the command (n) specifies the two's complement of the number of MAP registers to be transferred. After the contents of the registers have been transferred onto the stack, the MAP register selected as the stack pointer is incremented to point to the last word of data transferred onto the stack.

If bit 8=0: The MAP register selected as the stack pointer is 27--the Transformation Matrix Address (TMADR).

If bit 8=1: The next word of data input (SP register) is taken as the address of a MAP register which will be used as the stack pointer.

POP

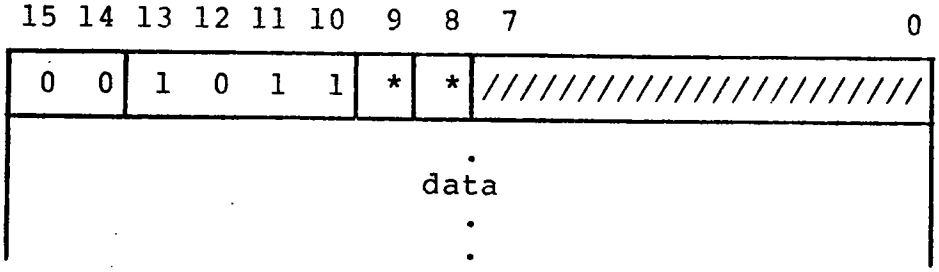


The POP command is used to decrement by n the MAP register which is functioning as a stack pointer. The operand field of the command (n) specifies the two's complement of the amount to decrement the stack pointer.

If bit 8=0: The MAP register selected as the stack pointer is 27--the Transformation Matrix Address (TMADR).

If bit 8=1: The next word of data input (SP register) will be used as the address of a MAP register to be used as the stack pointer.

MATCON



The MATCON (Matrix Concatenation) command is used to cause 16 words of data to be input which are then used as four rows of four elements to form a 4x4 matrix. This input matrix is then post-multiplied by the current Transformation Matrix (as indicated by the TMADR register) and the resultant normalized, compound matrix stored into the current Transformation Matrix.

If bit 8=0: The input matrix is post-multiplied by the Transformation Matrix, with the resultant matrix stored in the 16 words of the matrix stack immediately above the current Transformation Matrix. The 16 MAP registers above the Transformation Matrix are used to hold the intermediate or temporary results of the matrix multiplication.

If bit 8=1: The input matrix is post-multiplied by the Transformation Matrix, with the resultant matrix stored in the 16 words of the Matrix Stack immediately above the current Transformation Matrix. The TMADR is then incremented by 16, thereby pushing the previous contents of the Transformation Matrix onto the Matrix Stack and leaving the newly-concatenated compound matrix in the current Transformation Matrix.

If bit 9=0: The current Transformation Matrix is treated as a matrix whose rows contain the elements Aij (i.e., A11, A12, A13, A14, A21, A22,...). The resultant matrix is also treated as a matrix whose rows contain the elements Aij.

If bit 9=1: The current Transformation Matrix is treated as a matrix whose rows contain the elements Aji (i.e., A11, A21, A31, A41, A12, A22,...) or the transpose of the typical Transformation Matrix. The resultant matrix is also

treated as a matrix whose rows contain the elements Aji. This ability to treat the Transformation Matrix as the transpose allows the Picture Processor to effectively perform pre- or post-multiplication of matrices. This may be explained by the statement of some elementary axioms of matrix algebra:

(a) [A][B] not equal [B][A] (in general)
(b) [A]' = transpose of [A]
(c) ([A]')' = [A]
(d) [A]'[B]' = [BA]'

In (d) above, [B] is the Transformation Matrix and [A] is the matrix which is input. Then: if [A] is input as a transposed matrix and bit 9=1, [B] will be treated as a transposed matrix resulting in a compound temporary matrix [BA]' which, upon storing in the Transformation Matrix, will again be transposed. This results in the compound matrix ([BA]')' or [BA] in the Transformation Matrix. Matrix pre-multiplication may be performed in this manner.

b. RSR Drawing Commands

The Drawing command is used to specify the type of data to be input and the manner in which the data are to be processed. The Drawing command's basic format is shown in Figure 2-8.

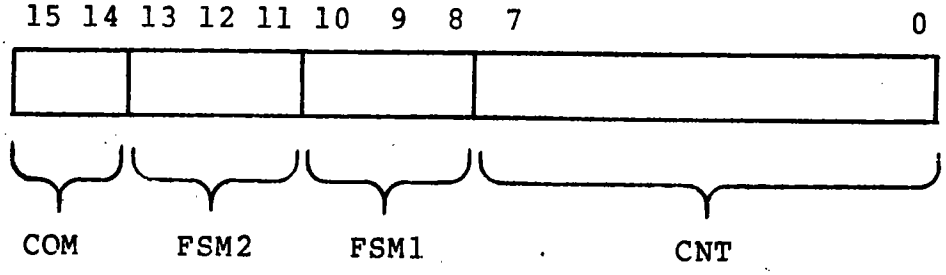


Figure 2-8
Basic Format of the Drawing RSR Commands

Bits of the Drawing RSR are divided into three fields:

1. Command (COM) - specifies what command is to be executed. This field is never automatically modified.

2. Finite State Machines (FSM1,FSM2) - these bits give further specification of the command to be executed. This field is automatically updated after each command execution.

3. Count (CNT) - these bits determine how many times the command should be executed before another RSR is required. This field is automatically incremented after each command execution until CNT=0; at this time, the next RSR to be executed is input.

The above fields are described in the following sections.

Command (COM)

COM=01-2DDRAW

Two words are accessed from the MAP input FIFO and are then processed by the Picture Processor as specified by the Finite State Machine bits. CNT is then incremented.

COM=10-3DDRAW

Three words are accessed from the MAP input FIFO and are then processed by the Picture Processor as specified by the Finite State Machine bits. CNT is then incremented. Note that for FSM2=5 (Pass Formatted), only two words of data will be output.

COM=11-4DDRAW

Four words are accessed from the MAP input FIFO and are then processed by the Picture Processor as specified by the Finite State Machine bits. CNT is then incremented. Note that for FSM2=5 (Pass Formatted), only two words of data will be output.

Finite State Machines (FSM1,FSM2)

FSM1- The FSM1 bits are used to describe the type of draw desired. Upon completion of the command execution, the bits are updated. The type, update definitions and FSM1 sequences initiated are listed below:

FSM1 OCTAL VALUE	TYPE	VALUE AFTER UPDATE	SEQUENCES
0	Moveto(M)	1	(M,D,M,D,...)
1	Drawto(D)	0	(D,M,D,M,...)
2	Moveto(M)	3	(M,D,D,D,...)
3	Drawto(D)	3	(D,D,D,D,...)
4	Moveto(M)	4	(M,M,M,M,...)
5	not used	4	
6	not used	7	
7	not used	6	

Interpretation of the various types of FSM1 follows:

Moveto: specifies a point in the coordinate system, normally used as the beginning point of a line. Note that FSM1=4 (M,M,M,...) is used to specify data which are to be processed as dots.

Drawto: indicates that a line is to be drawn from the last specified point to the point being speci-

fied.

FSM2- The FSM2 bits are used to specify whether the data accompanying DRAW commands are to be interpreted as Absolute, Relative (added to previous data), Offset (added to previously set origin) or as data which are to be Passed through the MAP, unprocessed, for output to memory or the Picture Generator. Upon completion of the command execution, the bits are updated. The interpretation, update definitions and FSM2 sequences initiated are listed below:

FSM2 OCTAL VALUE	TYPE	VALUE AFTER UPDATE	SEQUENCES
0	Set Base(SB)	1	(SB,O,O,O,...)
1	Offset(O)	1	(O,O,O,O,...)
2	Absolute(A)	3	(A,R,R,R,...)
3	Relative(R)	3	(R,R,R,R,...)
4	Absolute(A)	4	(A,A,A,A,...)
5	Pass Formatted(PF)	5	(PF,PF,PF,PF,...)
6	Pass Conditional(PC)	6	(PC,PC,PC,PC,...)
7	Pass(P)	7	(P,P,P,P,...)

Interpretation of the various types of FSM2 follows:

Set Base: specifies that the next set of coordinates input are to be loaded into the BASE (and INPUT) register of the MAP. For a 2DDRAW, only the x and y registers are loaded; for a 3DDRAW only, the x, y and z registers are loaded; for a 4DDRAW only, the x, y, z and w registers are loaded. For subsequent DRAW commands, the INPUT and BASE registers are used to compute the X, Y, Z and W coordinates as absolute data. A 4DDRAW Set Base should be done for 2DDRAW or 3DDRAW commands to establish the Z and W, or just W, coordinates for the input data.

Offset: specifies that subsequent data input for this command are to be processed as an offset from the current contents of the BASE register, thus treating the BASE register as an origin for subsequent data. For a 2DDRAW command, the X, Y coordinates are taken from the MAP input FIFO and the X, Y Base register values are added to them to form the X, Y input coordinates. The Z, W coordinates are taken from the BASE register. For a 3DDRAW command, the X, Y, Z coordinates are taken

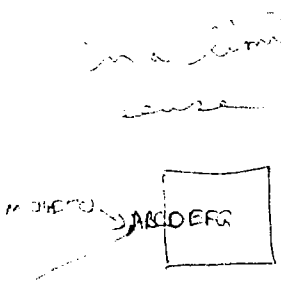
from the MAP input FIFO and the X, Y, Z, W Base register values are added to them to form the X, Y, Z, W input coordinates. For all DRAW commands, the BASE register is not updated.

Absolute: specifies that this set of coordinates is to be treated as absolute data. For a 2DDRAW command, the X, Y coordinates are taken from the MAP input FIFO and the Z, W coordinates are taken from the BASE register. For a 3DDRAW command, the X, Y, Z coordinates are taken from the MAP input FIFO and the W coordinate is taken from the BASE register. For a 4DDRAW command, the X, Y, Z, W coordinates are taken from the MAP input FIFO. For all DRAW commands, the new X, Y, Z, W coordinates are used to update the BASE register.

Relative: specifies that this set of coordinates is to be added to the current contents of the BASE register to form a set of absolute coordinates which are relative to the BASE register (i.e., the previous point). For all DRAW commands, the new X, Y, Z, W coordinates are used to update the BASE register.

Pass Formatted: specifies that subsequent data input for this command are to be passed through the MAP unprocessed, but formatted on output for input to the Picture Generator. For a 2DDRAW command, the X, Y coordinates are taken from the MAP input FIFO and the Z coordinate is taken from the BASE register. For 3DDRAW and 4DDRAW commands, the X, Y, Z coordinates are taken from the MAP input FIFO. For all DRAW commands the data is output as 12 bits of X, 12 bits of Y and 6 bits of Z information in the format specified in Section 2.3.3. The BASE register is not updated by a Pass Formatted command. The FSM1 field is used to determine whether the data is a Moveto or Drawto command.

Pass Conditional: specifies that subsequent data input for this command are to be passed through the MAP unprocessed, but output (unformatted in any manner) only if the last point processed by the MAP was not clipped. This allows data to be output from the MAP only if the last point would be "seen". This mode is used to conditionally send characters to the Picture Generator only if the Moveto used to position the character string was also output. Use of this mode allows character strings to be "clipped" if they are currently not in the field-of-view. If the data is to be output, this command functions identically to the



Pass command.

Pass: specifies that subsequent data input for this command is to be passed through the MAP unprocessed, and the output unformatted in any manner. This command is used to output character strings and status commands to the Picture Generator. See Section 2.4.3 for detailed data formats for the Picture Generator.

Count (CNT): this 8-bit field is used to specify how many times the command specified by the COM bits is to be repeatedly executed (with additional data each time). The field is treated as a two's complement number from -1 (377) to -256 (000) (the sign bit is always implied). The command is executed repeatedly, with the field incremented at the end of each command execution until it goes from -1 (377) to -256 (000). At that time, the MAP will attempt to input a new RSR from the MAP input FIFO unless the RSR HOLD bit is set in the MSR. When an RSR command is input from the FIFO, it is automatically placed in the MMRSR register (SCB:177755). The MMRSR register is never directly loaded by the host computer.

2.3.3 The MAP Output Formatter

The MAP Output Formatter is a PICTURE SYSTEM device which can function as either an Active or Passive device. In either operation, the function of the Output Formatter is to output the data directed to it by the MAP in the format required by the RSR command which was just executed. The Active/Passive state of the Output Formatter is selected by a bit in the MAP Status Register (MSR). When the MAP Output Formatter is selected as an Active device, there is an associated MAP Active Output Address register (MAOA) which holds the address of the location in PSMEM into which the next word of data output is to be stored.

a. MAP Active Output Limit (MAOL) - SCB:177750
The MAOL is a 16-bit R/W register which holds the upper limit address within PSMEM that is to be used by the MAP Output Formatter. The Output Formatter will output data up to, but not including, the address contained in MAOL. See MAOA below.

b. MAP Active Output Address (MAOA) - SCB:177751
The MAOA is a 16-bit R/W register. The next word of data output by the Output Formatter will be stored in the PSMEM location addressed by MAOA. This register is incremented after each word of data is stored into PSMEM. When MAOA=MAOL, the MAP Output Stopped (MOSTOPPED) bit in the MSR is set, indicating that no more data can be output by the Output Formatter. This condition can be cleared by resetting MAOA or MAOL so that MAOA is not equal to MAOL.

If the MAP Output Formatter is not selected as an Active device, it functions as a Passive device and all data output must be taken by an Active device from its Passive Output Port (MPOP, SCB:177776).

c. MAP Passive Output Port (MPOP) - SCB:177776
The MAP Passive Output Port is a 16-bit read-only location in the PICTURE SYSTEM SCB from which data output by the MAP may be taken by an Active device. In order to use the MPOP, the Active device's input address register should be set to address the MPOP using the Direct I/O path to the PICTURE SYSTEM and the MAP Output Formatter selected for passive output. Once this has been completed, all subsequent data output by the MAP will be transferred to the Active device without direct intervention by the host CPU.

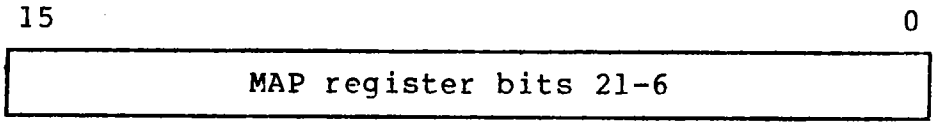
The MAP Output Formatter may be selected to inhibit the output of any data from the MAP by setting the Inhibit Output (IOUT) bit in the MSR. When this bit is set, input to the MAP may proceed, with the data being pro-

cessed as if output were not inhibited.

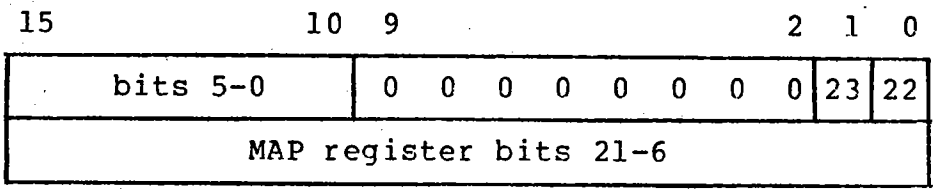
The exact data format for each word output is determined by the RSR command which was executed to produce the output from the MAP. Details of the data formats for each RSR command follows.

RSR Control Commands

The only control commands which cause data to be output from the MAP are the STORE and STORESTK commands. The output data format for these commands is identical. If the Precision bit of the RSR command (bit 8) is zero, one 16-bit word of data is output for each MAP register to be stored as shown below.



If the Precision bit of the RSR command (bit 8) is one, two 16-bit words of data are output for each MAP register to be stored as shown below.



In this extended precision mode, all 24 bits of each MAP register are output with MAP register bits 23, 22 and 5-0 are stored into bits 1,0 and 15-10 of the first word output; MAP register bits 21-6 are stored in the second word output.

RSR 2DDRAW, 3DDRAW and 4DDRAW Commands

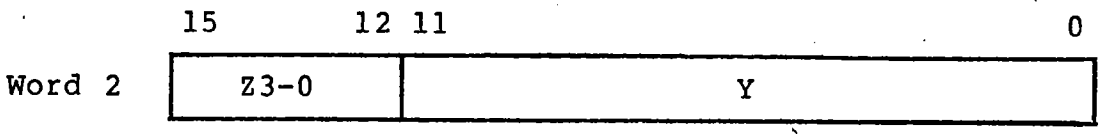
The output data formats for the 2DDRAW, 3DDRAW and 4DDRAW commands are identical. The 2D, 3D or 4D specification is used merely to determine the number of coordinates to be input to the MAP. Thus, the output data for these command formats may be detailed together. The output data formats for the DRAW commands are controlled by the MAP mode bits of the MSR (MMODE, bits 9,8). The status of these bits determine how the MAP is to process the two-, three- or four-dimensional data which are input and in what format the data are to be output. The output data formats for each of the different modes are detailed below.

MMODE=00 - Display Mode

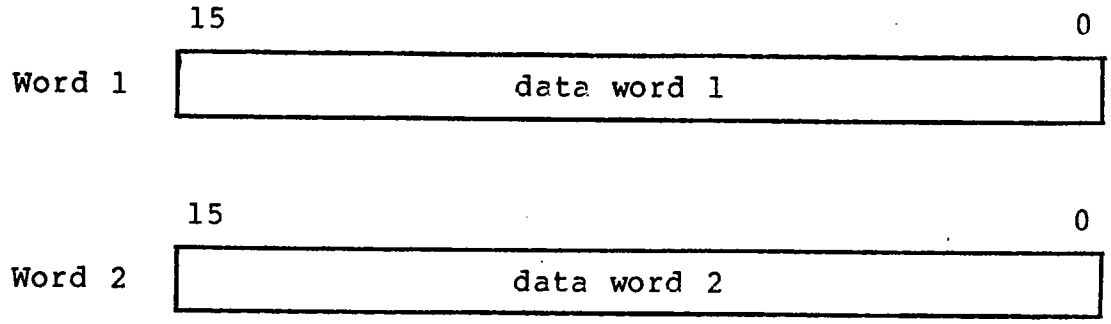
This mode causes all data output for DRAW commands to be formatted for input to the Picture Generator. The data format is:

For all DRAW commands when FSM2=0-5, two 16-bit words of data are output for each line segment endpoint output by the MAP as shown below:

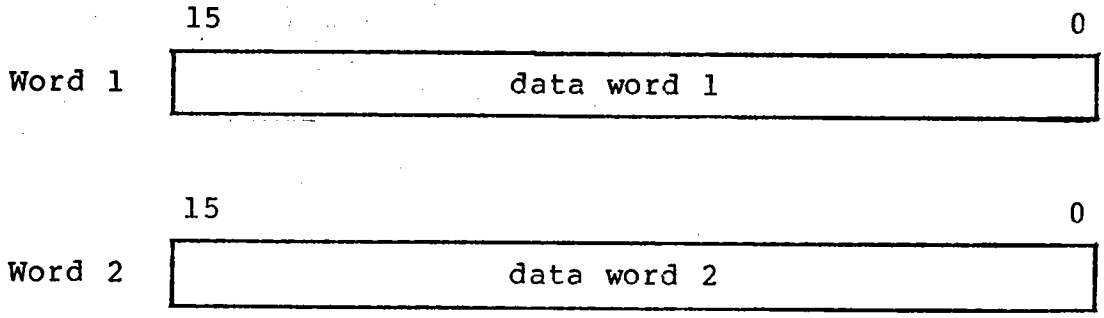
*=0: Move
*=1: Draw



For all DRAW commands when FSM2=6, two 16-bit words of data are output for each DRAW command execution only if the last data point processed by the MAP was unclipped. The two words of data output are identical to the first two words of data input for this command as shown below. If more than two words of data are input by the MAP (i.e., for 3DDRAW or 4DDRAW commands), any extra words of data input are discarded.



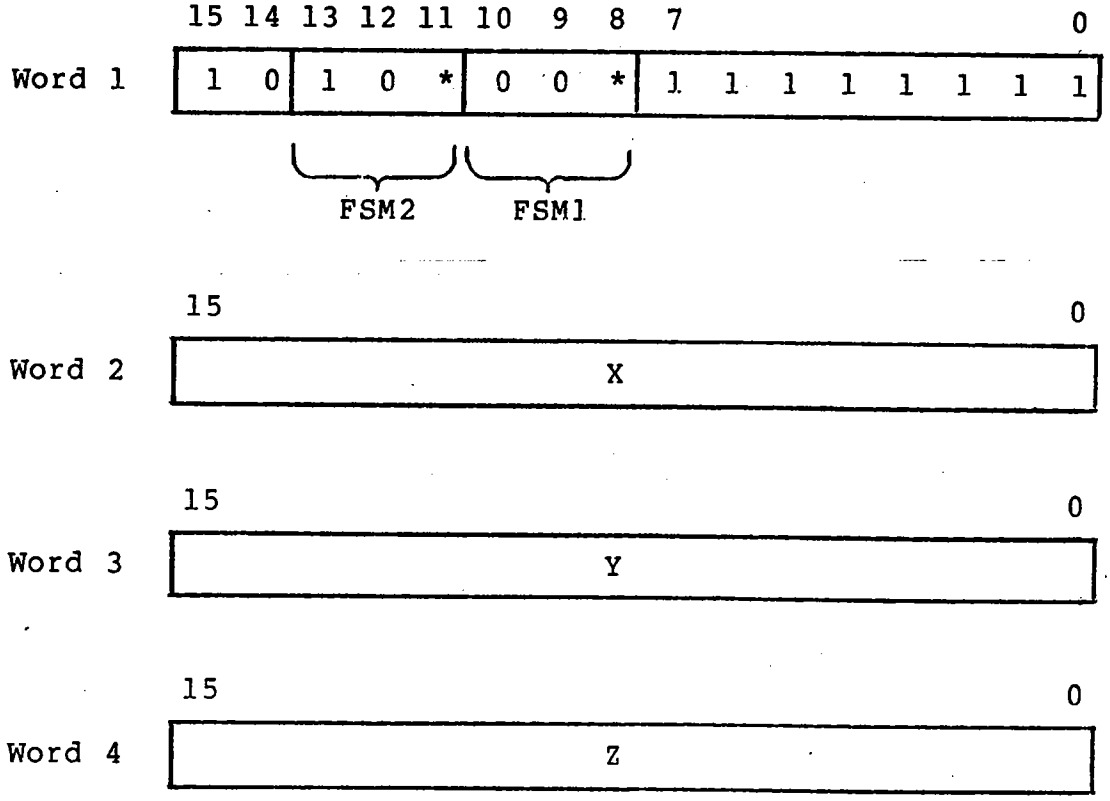
For all DRAW commands when FSM2=7, two 16-bit words of data are output for each DRAW command execution. The two words of data output are identical to the first two words of data input for this command as shown below. If more than two words of data are input by the MAP (i.e., for 3DDRAW or 4DDRAW commands), any extra words of data input are discarded.



MMODE=01 - Sixteen Bit Precision Mode

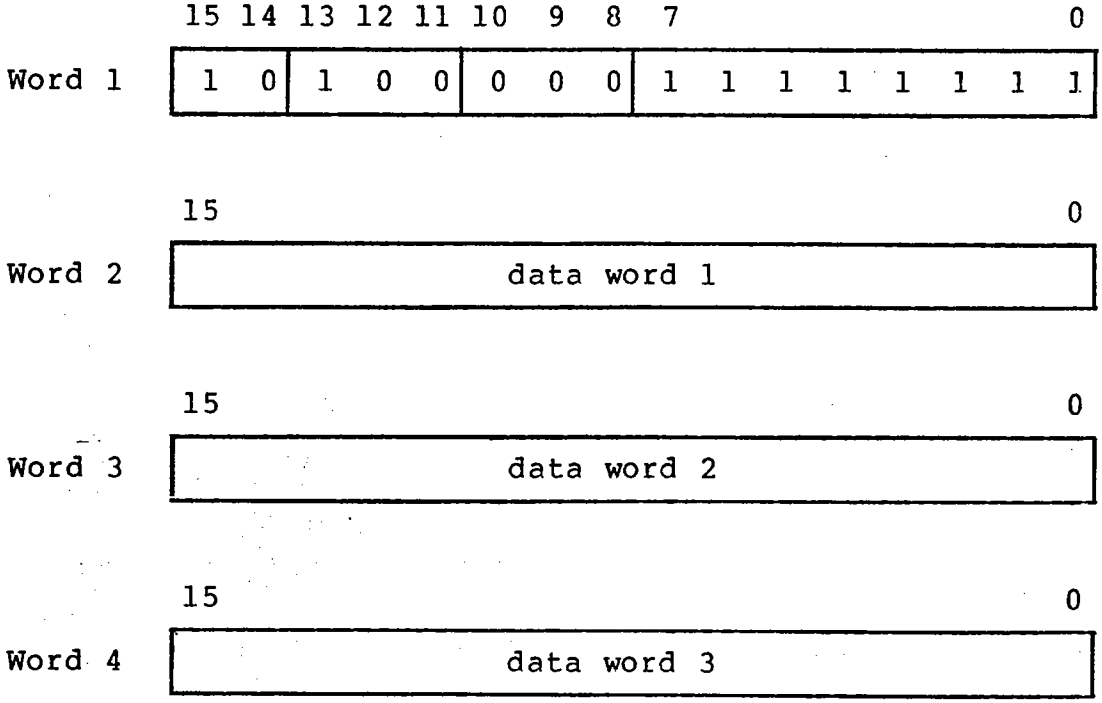
This mode causes all data output for DRAW commands to be formatted as a 3DDRAW RSR command which describes how the three words of data are to be interpreted. Four words of data are always output for each DRAW command execution. The data format is:

For all DRAW commands when FSM2=0-5, four 16-bit words of data are output for each line segment endpoint output by the MAP. The data output will always be absolute with FSM2=4, except when in Pass Formatted mode where FSM2=5. FSM1=0 for a Move and FSM1=1 for a Draw as shown below.

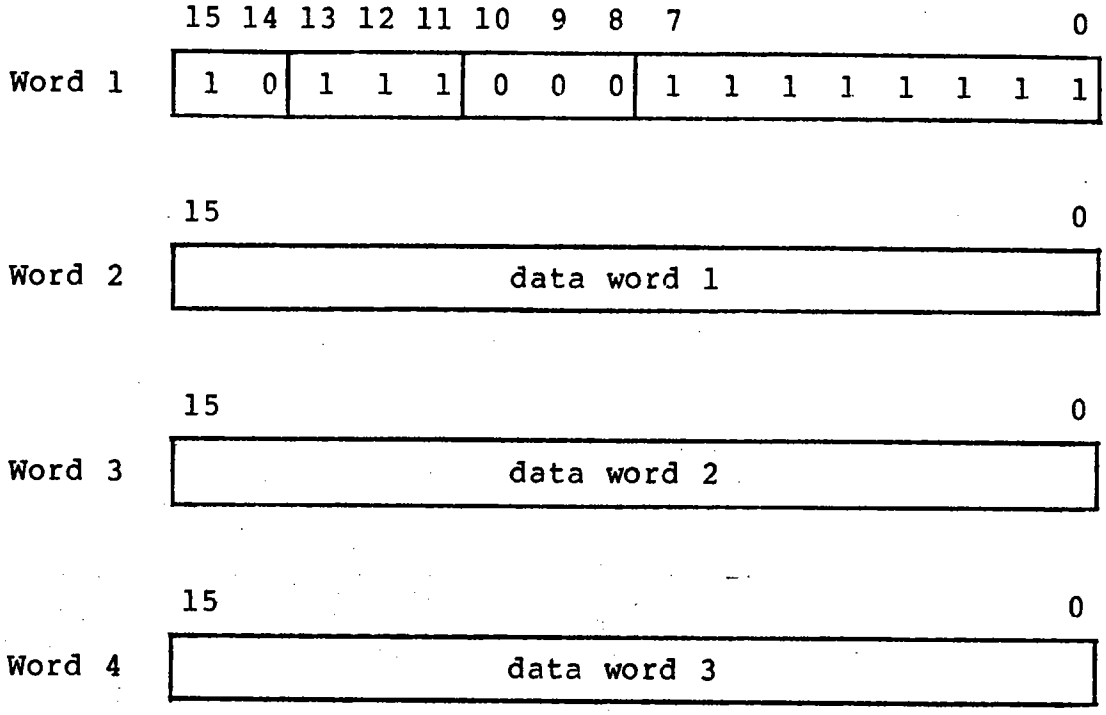


For all DRAW commands when FSM2=6, four 16-bit words of data are output for each DRAW command execution only if the last data point processed by the MAP was unclipped. There are four words of data output; the first of which is an RSR command. The remaining words consist of the first three words of data that were input for this command as shown below. If less than three words of data were input by the MAP (i.e., for a 2DDRAW command), the fourth word of data output will be invalid data. If more than three words of data were input by the MAP

(i.e., for a 4DDRAW command), the extra word of data input is discarded.



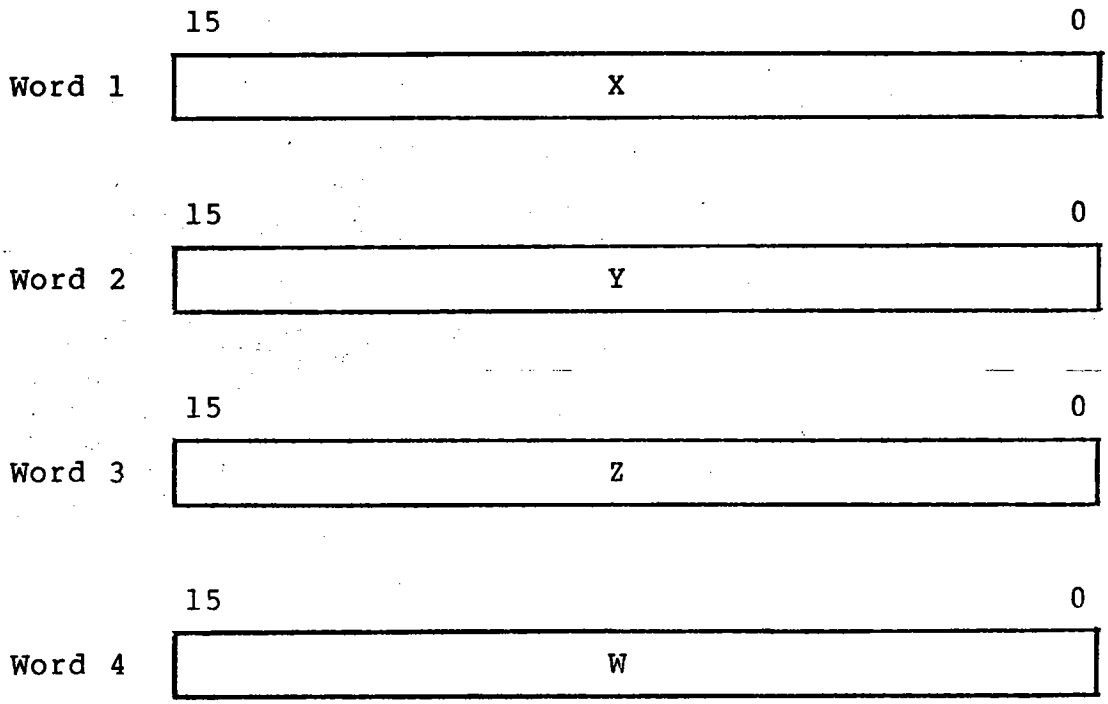
For all DRAW commands when FSM2=7, four 16-bit words of data are output for each DRAW command execution. There are four words of data output; the first of which is an RSR command. The remaining three words consist of the first three words of data that were input for this command as shown below. If less than three words were input by the MAP (i.e., for a 2DDRAW command), the fourth word of data output will be invalid data. If more than three words of data were input by the MAP (i.e., for a 4DDRAW command), any extra words of data input are discarded.


15 14 13 12 11 10 9 8 7 0
Word 1
1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1
15 0
Word 2
data word 1
15 0
Word 3
data word 2
15 0
Word 4
data word 3

MMODE=10 - Transform/Normalize Mode

This mode causes all data output for DRAW commands to be output as four words of data which represent the transformed and normalized X, Y, Z, W coordinates of the point just processed by the MAP. The data format is:

For all DRAW commands when FSM2=0-4, four 16-bit words of data are output for each line segment endpoint input to the MAP. The output format is shown below:



For all DRAW commands when FSM2=5-7, four 16-bit words of data are output for each DRAW command execution. The four words of data output are the same four words of data that were input for this command as shown for FSM2=0-4. If less than four words of data were input by the MAP (i.e., for 2DDRAW or 3DDRAW commands), the third and fourth (or just fourth) words of data output will be invalid data.

MMODE=11 - Transform Only Mode

This mode causes all data output for DRAW commands to be output as four words of data which represent the transformed X, Y, Z, W coordinates of the point just processed by the MAP. The data format is:

For all DRAW commands when FSM2=0-4, four 16-bit words of data are output for each line segment endpoint input to the MAP. The data output consists of 1 sign bit and 15 bits of significance. It should be noted that the process of transforming the input coordinates may cause the true sign of a coordinate to be reflected by more than 1 sign bit. This is accounted for in the MAP by designating 3 bits of sign for each coordinate processed, with the result typically normalized so that 1 sign bit reflects the true sign of the number. However, when using this mode, the number is not normalized; hence, the 16-bits of data output for each coordinate may not reflect the true sign of the number. The output format is identical to that shown for MMODE=10.

For all DRAW commands when FSM2=5-7, four 16-bit words of data are output for each DRAW command execution. The four words of data output are the same four words of data that were input for this command as shown for MODE=10. If less than four words of data were input by the MAP (i.e., for 2DDRAW or 3DDRAW commands), the third and fourth (or just fourth) words of data output will be invalid data.

2.3.4 Picture Processor Maintenance Registers

The maintenance registers of the Picture Processor provide access to the internal registers and data paths of the Picture Processor. Using these maintenance registers allows the Picture Processor microinstructions to be executed in single-step mode and also allows the microcode instruction register (or "DOIT" register) to be read and written providing complete control of the Picture Processor. Additionally, maintenance registers provide the way in which to load the Picture Processor microcode when it is equipped with the writeable control store feature. Other than the initial clearing of the MAP HLT and MAP SNGL STP bits of the MMSR register (SCB:177754, see below), these registers need not be accessed during normal operation.

The Picture Processor is controlled by eight registers in the SCB as shown in Figure 2-9. The maintenance registers (SCB:177754-177757) are described in detail in the following sections.

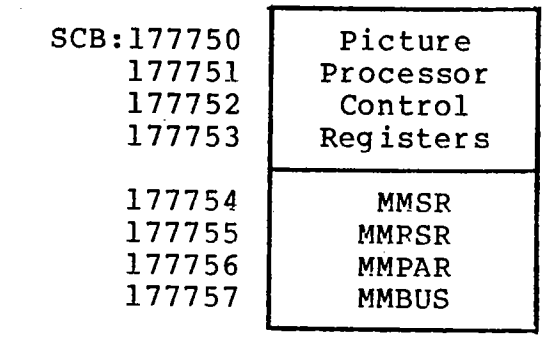
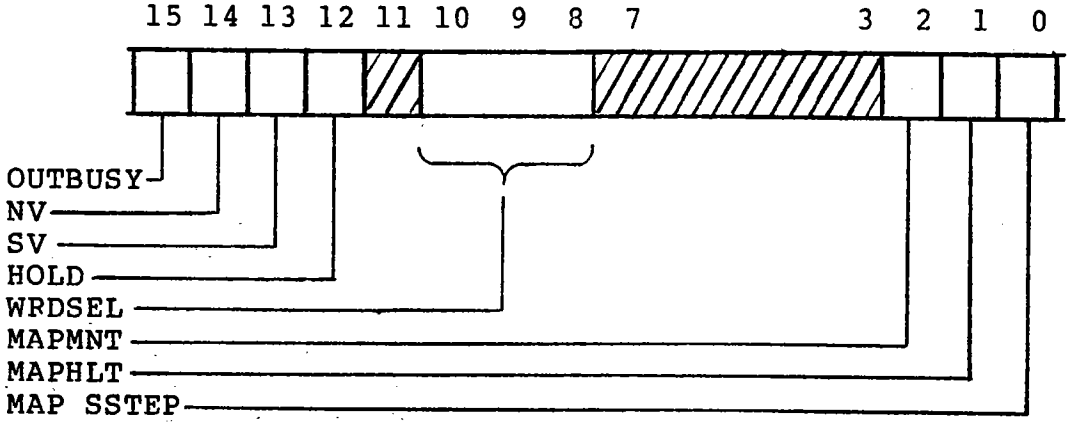


Figure 2-9
Picture Processor Control and Maintenance Registers

a. MAP Maintenance Status Register (MMSR)-SCB:177754



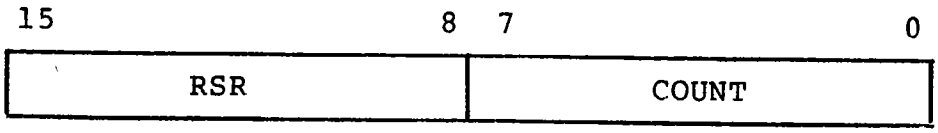
The MMSR is used to provide status and control of the Picture Processor for maintenance purposes.

BIT	NAME	FUNCTION
15	OUTBUSY	Setting OUTBUSY indicates the MAP Output Formatter is busy. Read only. Cleared by RESET.
14	NV	NV is a status indicator which, when set, indicates that the new point has been clipped. NV, in conjunction with SV (bit 13, see below), can be used to determine how the current line has been clipped. (See SV for further details.) Read only. Cleared whenever a new point is input by the MAP for processing and RESET.
13	SV	SV is a status indicator which, when set, indicates that the saved point (the unclipped x,y,z and w coordinates of the previously processed point) has been clipped. This bit, in conjunction with NV, can be used to determine how the current line has been clipped (if at all). The bits are valid only at the end of the processing of a point when the MAP is in

		the HOLD state. Read only. Cleared by RESET whenever a new point is input by the MAP for processing.
12	HOLD	HOLD is a status indicator which, when set, indicates that the MAP is in the HOLD state awaiting another RSR command for processing. Read only. Set by the MAP Output Formatter after the last point has been processed and RESET.
11	not used	
10-8	WRDSEL	WRDSEL (Control Store Word Select) is used to select which part of the 96-bit "DOIT" register can be read or written via the MMBUS register. The WRDSEL field is used to address the DOIT register in the following manner:
	0	DOIT bits 95-80
	1	DOIT bits 79-64
	2	DOIT bits 63-48
	3	DOIT bits 47-32
	4	DOIT bits 31-16
	5	DOIT bits 15-0
	6	Write DOIT register to Writeable Control Store word addressed by PRMADR field of MMPAR.
	7	Read Control Store word addressed by PRMADR field of MMPAR.
		When the WRDSEL field overflows (7=>0), the PRMADR field of the MMPAR register is incremented by one allowing sequential locations of the Control Store to be read or written easily. WRDSEL is used in conjunction with MAPMNT (bit 2, see below). Cleared by RESET.
7-3	not used	

2	MAPMNT	When MAPMNT is set, it enables the Control Store "DOIT" register to be read or written by the MMBUS register. The WRDSEL field is used to select which bits of the "DOIT" register are to be accessed. MAPMNT must be set for the "DOIT" register to be accessed. Set by RESET. Must be programmably cleared before normal use of the Picture Processor may proceed.
1	MAPHLT	Setting MAPHLT disables the Picture Processor system clock which normally issues a clock pulse each 150 nsec. This freezes the Picture Processor for diagnostic purposes. Set by RESET. Must be cleared before normal use of the Picture Processor may proceed.
0	MAP SSTEP	This bit is set to issue a single clock pulse to the Picture Processor. Write only. Always reads as a zero.

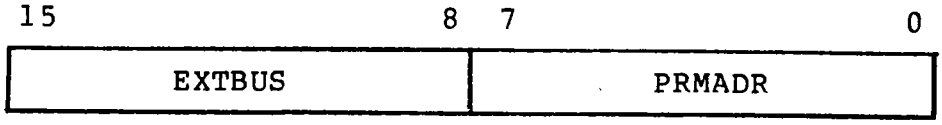
b. MAP Maintenance RSR (MMRSR)-SCB:177755



The MAP Maintenance RSR (Repeat Status Register) is a read-only register containing the current instruction being processed by the MAP. The fields of this register are detailed below.

BIT	NAME	FUNCTION
15-8	RSR	This field will contain the current Repeat Status Register command being executed by the MAP. (See Section 2.3.2.2 for RSR command details.) Does not clear by RESET.
7-0	COUNT	This field contains the current count of the MAP RSR instruction only when the MAP is in the HOLD state (see MMSR, bit 12). When the MAP is not in the HOLD state, this field contains the current address into the MAP RAMs. Read only. Cleared by RESET.

c. MAP Maintenance PROM Address Register (MMPAR)-SCB:177756



The MAP Maintenance PROM Address Register provides the fields which allow the 24-bit data paths to be accessed to their full precision and the Control Store to be addressed.

BIT	NAME	FUNCTION
15-8	EXTBUS	This field contains the extended bits of the MAP B-Bus (a 24-bit data path). EXTBUS contains bits 5-0 and 23,22 of the B-Bus in bits 15-8 of the MMPAR register.
7-0	PRMADR	This field contains the next state address of the Control Store MAP state machine. Read only. Cleared by RESET. When the Picture Processor is in maintenance mode (MAPMNT=1), PRMADR becomes a R/W field.

d. MAP Maintenance B-Bus (MMBUS)-SCB:177757

This register contains bits 21-6 of the main internal B-Bus of the MAP. MMBUS, in conjunction with EXTBUS, may be used to read the 24-bit B-Bus. When the Picture Processor is in maintenance mode (MAPMNT=1), the "DOIT" register may be read or written by reading or writing MMBUS (WRDSEL selects which 16 bits of the DOIT register are to be accessed). Read/Write. Cleared by RESET. Any access to this register will increment the WRDSEL field of the MMSR after the operation.

The following table details the data which may be read or written using this register for each of the WRDSEL values. The DOIT bit <95-0> and the name of the particular function performed for each bit are also included. If the contents of a bit field is not specified, it should be considered undefined.

WRDSEL	MMBUS Register Contents
0	MMBUS<15>=DOIT<95> NBADR(7)
	MMBUS<14>=DOIT<94> NBADR(6)
	MMBUS<13>=DOIT<93> NBADR(5)
	MMBUS<12>=DOIT<92> NBADR(4)
	MMBUS<11>=DOIT<91> NBADR(3)
	MMBUS<10>=DOIT<90> NBADR(2)
	MMBUS<09>=DOIT<89> NBADR(1)
	MMBUS<08>=DOIT<88> NBADR(0)
	MMBUS<07>=DOIT<87> *LDSUB
	MMBUS<06>=DOIT<86> *SELSUBRTN
	MMBUS<05>=DOIT<85> *RSRUPDAT
	MMBUS<04>=DOIT<84> *RSRLOAD
	MMBUS<03>=DOIT<83> *GATFIFO
	MMBUS<02>=DOIT<82> *OUTPUTSET
	MMBUS<01>=DOIT<81> *OUTPUTLOAD
	MMBUS<00>=DOIT<80> *SETPNT
1	MMBUS<15>=DOIT<79> *RAMWRT
	MMBUS<14>=DOIT<78> *MBSEL
	MMBUS<13>=DOIT<77> *COUNTSEL
	MMBUS<12>=DOIT<76> *DTREGA
	MMBUS<11>=DOIT<75> *DTREGB
	MMBUS<10>=DOIT<74> *DTREGOUT
	MMBUS<09>=DOIT<73> *MMPSELG
	MMBUS<08>=DOIT<72> *SELMSH
	MMBUS<07>=DOIT<71> *MNTOUT
	MMBUS<06>=DOIT<70> *MBCBSEL
	MMBUS<05>=DOIT<69> *MDACBSEL
	MMBUS<04>=DOIT<68> *RACBSEL
	MMBUS<03>=DOIT<67> *MBLOAD
	MMBUS<02>=DOIT<66> *MDALOAD
	MMBUS<01>=DOIT<65> *RALOAD
	MMBUS<00>=DOIT<64> *LOGICF
2	MMBUS<15>=DOIT<63> *CARRIN
	MMBUS<14>=DOIT<62> *FUNSEL0
	MMBUS<13>=DOIT<61> *FUNSEL1
	MMBUS<12>=DOIT<60> *FUNSEL2
	MMBUS<11>=DOIT<59> *FUNSEL3
	MMBUS<10>=DOIT<58> *MRALOAD
	MMBUS<09>=DOIT<57> *NMROUT
	MMBUS<08>=DOIT<56> *MRASEL
	MMBUS<07>=DOIT<55> REG 7
	MMBUS<06>=DOIT<54> REG 6
	MMBUS<05>=DOIT<53> REG 5
	MMBUS<04>=DOIT<52> REG 4
	MMBUS<03>=DOIT<51> REG 3
	MMBUS<02>=DOIT<50> REG 2
	MMBUS<01>=DOIT<49> REG 1
	MMBUS<00>=DOIT<48> REG 0

```
3       MMBUS<15>=DOIT<47>  *NSRLOAD
        MMBUS<14>=DOIT<46>  *NSRMINLD
        MMBUS<13>=DOIT<45>  *BUSADRSEL
        MMBUS<12>=DOIT<44>  *CNTSELA
        MMBUS<11>=DOIT<43>  *CNTSELB
        MMBUS<10>=DOIT<42>  *CHECKSTATUS
        MMBUS<09>=DOIT<41>  *DISPATCH
        MMBUS<08>=DOIT<40>  *TRANSPOSE
        MMBUS<07>=DOIT<39>  *AC0LOAD
        MMBUS<06>=DOIT<38>  *AC0INC
        MMBUS<05>=DOIT<37>  *AC1LOAD
        MMBUS<04>=DOIT<36>  *AC1INC
        MMBUS<03>=DOIT<35>  *AC2LOAD
        MMBUS<02>=DOIT<34>  *AC2INC
        MMBUS<01>=DOIT<33>  *AC3LOAD
        MMBUS<00>=DOIT<32>  *AC3INC

4       MMBUS<15>=DOIT<31>  *R1ROMOUT
        MMBUS<14>=DOIT<30>  *R2ROMOUT
        MMBUS<13>=DOIT<29>  *LDFLAGS
        MMBUS<12>=DOIT<28>  *USENORM
        MMBUS<11>=DOIT<27>  *J1B
        MMBUS<10>=DOIT<26>  *K1B
        MMBUS<09>=DOIT<25>  *USEJ1B
        MMBUS<08>=DOIT<24>  *USEK1B
        MMBUS<07>=DOIT<23>  *USESV
        MMBUS<06>=DOIT<22>  *USENV
        MMBUS<05>=DOIT<21>  *JV
        MMBUS<04>=DOIT<20>  *CLRV
        MMBUS<03>=DOIT<19>  *D
        MMBUS<02>=DOIT<18>  *C
        MMBUS<01>=DOIT<17>  *B
        MMBUS<00>=DOIT<16>  *A

5       MMBUS<15>=DOIT<15>
        MMBUS<14>=DOIT<14>
        MMBUS<13>=DOIT<13>
        MMBUS<12>=DOIT<12>
        MMBUS<11>=DOIT<11>
        MMBUS<10>=DOIT<10>
        MMBUS<09>=DOIT<09>
        MMBUS<08>=DOIT<08>
        MMBUS<07>=DOIT<07>  *GATCOUNT
        MMBUS<06>=DOIT<06>  *EXTSEL
        MMBUS<05>=DOIT<05>  *HLTINT
        MMBUS<04>=DOIT<04>  *GATMRI
        MMBUS<03>=DOIT<03>  *LDMRI
        MMBUS<02>=DOIT<02>  *LOADEXT
        MMBUS<01>=DOIT<01>  *CLREXT
        MMBUS<00>=DOIT<00>  *JHIT
```

2.4 The Picture Generator

The Picture Generator is the unit of the PICTURE SYSTEM that controls the drawing of lines and characters on the Picture Display. There are four basic units of the Picture Generator:

1. The Line Generator Input Controller
2. The Refresh Controller
3. The Line Generator
4. The Character Generator

These units function together to input data from the refresh buffer area of PICTURE SYSTEM memory (or to accept data directly from an Active device--the MAP Output Formatter, for example) and to convert the data into the appropriate analog signals to drive the CRT(s).

The Picture Generator input and status is controlled by fourteen registers in the SCB. These registers, shown in Figure 2-10, together with the four basic units of the Picture Generator, are described in detail in the following sections.

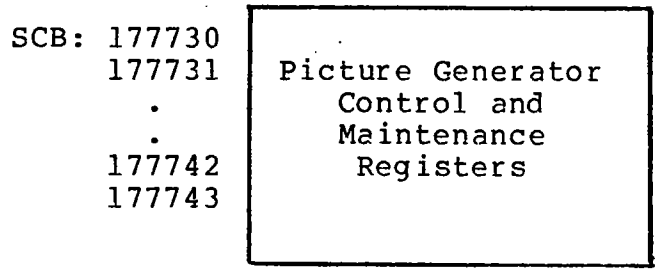


Figure 2-10
Picture Generator Input Control, Status and Maintenance Registers

2.4.1 The Line Generator Input Controller

The Line Generator Input Controller is a Passive device whose function is simply to transfer 16-bit words of data passively received to a 8-word Line Generator FIFO (First In First Out) buffer. As the FIFO becomes full, the Input Controller waits for the Line Generator to empty two 16-bit words from the FIFO before the next word of data is transferred to the FIFO.

a. Line Generator Passive.Input Port (LGPIP)-SCB:177775

The Line Generator Passive Input Port is a 16-bit write-only location in the PICTURE SYSTEM SCB to which data may be directed by an Active device for input to the Picture Generator. In order to use the LGPIP, the active device's output address register should be set to address the LGPIP using the Direct I/O path to the PICTURE SYSTEM. Once this has been done, all subsequent data output by the Active device will be transferred to the Line Generator without direct intervention by the host CPU.

2.4.2 The Refresh Controller

The Refresh Controller is an Active device whose function is to control the refreshing of pictures on the Picture Display. The Refresh Controller reads data from the area of PICTURE SYSTEM memory designated as refresh buffer area and transfers the data to the Line Generator Input Controller for subsequent display by the Line Generator. The Refresh Controller is also equipped with special-purpose segmentation registers to facilitate refresh buffer segmentation.

Refresh and segmentation is controlled by eight registers in the SCB. These registers, shown in Figure 2-11, are described in detail below:

SCB:	177730	RFCSN
	177731	RFSN
	177732	RFAWA
	177733	RFAWL
	177734	RFAIA
	177735	RFASA
	177736	RFAIL
	177737	RFSR

Figure 2-11
Refresh Status and Control Registers

a. Refresh Current Segment Name (RFCSN)-SCB:177730

The RFCSN is a 16-bit R/W register which holds the name of the segment of the refresh buffer currently being refreshed by the Picture Generator. Whenever a Segment Name is encountered by the Refresh Controller, that name is loaded into the RFCSN. If RFCSN=RFSN, a segment match is indicated by setting the MATCH REQ bit in the System Interrupt Request register. If the MATCH HOLD bit in the RFSR is set, the Refresh Controller will then wait for the MATCH REQ bit to be reset before the refresh cycle continues.

b. Refresh Segment Name (RFSN)-SCB:177731

The RFSN is a 16-bit R/W register which holds the name of a segment which may be searched for during the traversing of the refresh buffer by the Refresh Controller while refreshing the display file. The search mode is enabled by setting the SEARCH bit in the RFSR. A search may be done in any of the modes available, as specified by the RFSR SEARCH MODE bits. When a search match occurs (RFCSN=RFSN), the MATCH REQ bit in the System Interrupt Request register is set, indicating that a match has been found. Note: the match need not be equal, depending upon the status of the SEARCH MODE bits in the RFSR.

c. Refresh Active Writeback Address (RFAWA)-SCB:177732

The RFAWA is a 16-bit R/W register, and is used in the "writeback" operation. When the Input Controller is in writeback mode, it reads, from the PSMEM, the word of data addressed by RFAIA, and writes it back into the location indicated in the RFAWA. Writeback mode is selected by setting the WRITEBACK bit in the RFSR and is used to automatically reclaim the refresh buffer area allocated to deleted segments. This register is automatically incremented after each word is written. When RFAWA=RFAWL, the RFHOLD bit in the RFSR is set, indicating that writeback cannot proceed. This condition may be cleared by resetting the WRITEBACK bit in the RFSR or resetting RFAWA or RFAWL so that RFAWA is not equal to RFAWL. The WBSTOP REQ bit in the System Interrupt Request Register must then be acknowledged.

d. Refresh Active Writeback Limit (RFAWL)-SCB:177733

The RFAWL is a 16-bit R/W register which holds the upper address limit within PSMEM that is to be written

by the Refresh Controller during writeback. The Refresh Controller will write data up to, but not including, the address contained in RFAWL.

e. Refresh Active Input Address (RFAIA)-SCB:177734

The RFAIA is a 16-bit R/W register which holds the PSMEM address used to fetch the next word of data for input to the Picture Generator. After each word of data is transferred to the Line Generator Input Controller, this register is incremented so that the next word of data is taken from the next sequential location of PICTURE SYSTEM memory. When RFAIA=RFAIL, the RFSTOPPED bit in the Refresh Status Register (RFSR, see below) is set, indicating that no more data can be input by the Refresh Controller. This condition may be cleared by setting the RFSTART bit in the RFSR which then loads the RFAIA with the contents of the Refresh Active Start Address register, RFASA. This register should not be used to address any Passive device other than PICTURE SYSTEM memory. Incrementation of this register cannot be inhibited by addressing one of the eight passive I/O ports (SCB:177770-177777).

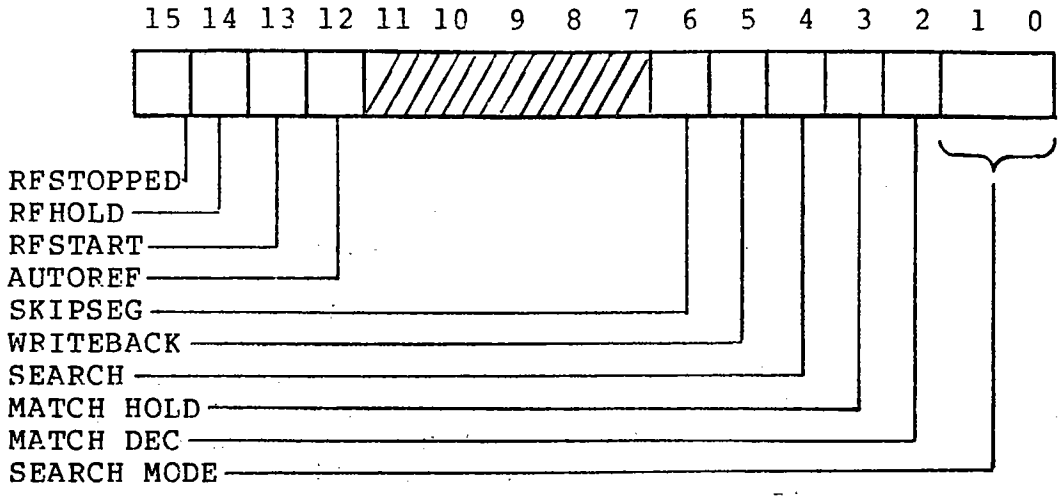
f. Refresh Active Start Address (RFASA)-SCB:177735

The RFASA is a 16-bit R/W register which holds the initial PSMEM address which is to be loaded into the RFAIA upon the start of a refresh cycle. The contents of this register is loaded into the RFAIA by the Refresh Controller whenever the RFSTART bit in the RFSR is set or by the Refresh Controller when an automatic refresh cycle is begun (see RFSR, AUTOREF).

g. Refresh Active Input Limit (RFAIL)-SCB:177736

The RFAIL is a 16-bit R/W register which holds the upper limit address within PSMEM that is to be used by the Refresh Controller. The Refresh Controller will input data up to, but not including, the address contained in RFAIL.

h. Refresh Status Register (RFSR)-SCB:177737



The RFSR is a 16-bit R/W register which is used to provide operating mode information to the Refresh Controller and status of the Refresh Controller back to the control program.

BIT	NAME	FUNCTION
15	RFSTOPPED	Setting RFSTOPPED (Refresh Stopped), indicates that the Refresh Controller has stopped reading data from the refresh buffer and is waiting for another refresh cycle to begin. Read only. Set by RESET and cleared by setting RFSTART.
14	RFHOLD	Setting RFHOLD (Refresh Hold), indicates that the Refresh Controller has stopped reading data from the refresh buffer since a MATCH REQ or the writeback limit has been met (RFAWA=RFAWL). Read only. Cleared by RESET or acknowledgement of MATCH REQ or WBSTOP REQ.
13	RFSTART	This bit is set to initiate execution of a refresh cycle by the Refresh Controller. Setting this bit causes the

		contents of the RFASA register to be loaded into the RFAIA. All subsequent data read by the Refresh Controller will be input to the Picture Generator for display until RFAIA=RFAIL or a Refresh Status Halt is encountered. Write only. Always reads as a zero.
12	AUTOREF	This bit is set to place the Refresh Controller into a self-operating mode of control where refreshing will proceed, without software control, at the interval of the line frequency specified by the Real Time Clock Counts register (SCB:177745). The Refresh Controller reads the contents of PSMEM from the location specified by the RFASA register to the location specified by the RFAIL register, or until a Refresh Control command with the HALT bit set (see Section 2.4.3c) is encountered. If the Refresh Controller has not completed an automatic refresh cycle when the clock interval elapses, the next refresh cycle will not begin until the clock interval again elapses. See Section 2.6.1 for further details on refresh interval selection. Cleared by RESET.
11-7	not used	
6	SKIPSEG	Setting SKIPSEG (Skip Segment), causes the Refresh Controller to skip all further data in the refresh buffer until RFSTOPPED or RFHOLD is set. This bit, in conjunction with the MATCH HOLD and SEARCH bits, allows segments to be skipped or blanked during refresh. Cleared by RESET.

5	WRITEBACK	When WRITEBACK is set, each word of data read by the Refresh Controller will be input to the Line Generator Input Controller as usual and also written back into PSMEM in the location specified by the RFAWA register. After each word is written, the RFAWA register is incremented allowing sequential PSMEM locations to be written. When RFAWA=RFAWL, the Refresh Controller is placed into the HOLD state (with RFHOLD set) until the condition is acknowledged and cleared.
4	SEARCH	When SEARCH is set, the Refresh Controller compares all segment names encountered during a refresh cycle with the name contained in RFSN. If a match occurs (as determined by the SEARCH MODE bits, see below), the MATCH REQ bit is set in the System Interrupt Request register. If the MATCH HOLD bit is set, the Refresh Controller will halt with the RFHOLD bit set.
3	MATCH HOLD	When MATCH HOLD is set, the Refresh Controller will terminate input to the Line Generator Input Controller whenever a MATCH REQ is issued and SEARCH (bit 4) is set. The RFAIA will be addressing the PSMEM location following the segment name that was searched for. Cleared by RESET.
2	MATCH DEC	When MATCH DEC is set, the Refresh Controller, when in a SEARCH operation with the MATCH HOLD bit set, will automatically reposition the RFAIA to point at the segment name encountered rather than after it, as is the normal mode of operation when MATCH DEC = 0. This causes the RFAIA to be

		decremented by two on a segmented name match. Cleared by RESET.
1,0	SEARCH MODE	These bits determine how the Segment Name in the RFSN is to be used in searching for a valid match. Set to 00 by RESET. The bit combinations and interpretations are as follows:
	00	Search for any name match. In this mode, any segment name encountered will cause the MATCH REQ to be issued.
	01	Search for right match. In this mode, any segment name encountered whose eight least-significant bits are equal to the eight least-significant bits of the RFSN register will cause a MATCH REQ to be issued.
	10	Search for left match. In this mode, any segment name encountered whose eight most-significant bits are equal to the eight most-significant bits of the RFSN register will cause a MATCH REQ to be issued.
	11	Search for exact match. In this mode, any segment name encountered which is equal to the segment name in the RFSN register will cause a MATCH REQ to be issued.

2.4.3 The Line Generator

The Line Generator is the unit of the Picture Generator that removes the digital data, input by the Input Controller, from the Line Generator input FIFO and converts the data into the appropriate analog signals to drive the display(s). The Line Generator removes two 16-bit words from the input FIFO and interprets the data as a 32-bit word which specifies:

a. A Move or Draw command with X, Y and Z (intensity) information.

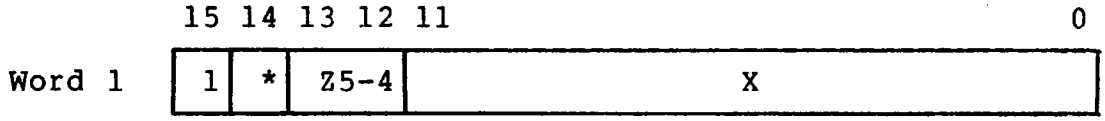
b. A Status command to be interpreted by the Line Generator to change line textures, line colors, etc.

c. A Refresh Control command for use by the Refresh Controller in performing refresh and segmentation operations.

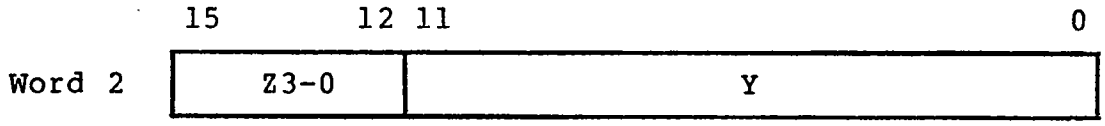
d. A Character command with four character codes to be interpreted by the Character Generator.

The data formats for each of these Line Generator words are detailed in the following sections.

a. MOVE or DRAW Command



*=0: MOVE
*=1: DRAW



A Move command to the Line Generator causes the beam to be positioned at the X,Y position specified and at the intensity specified by the 6 bits of Z information. The beam may or may not be intensified depending upon the current status of the Line Generator.

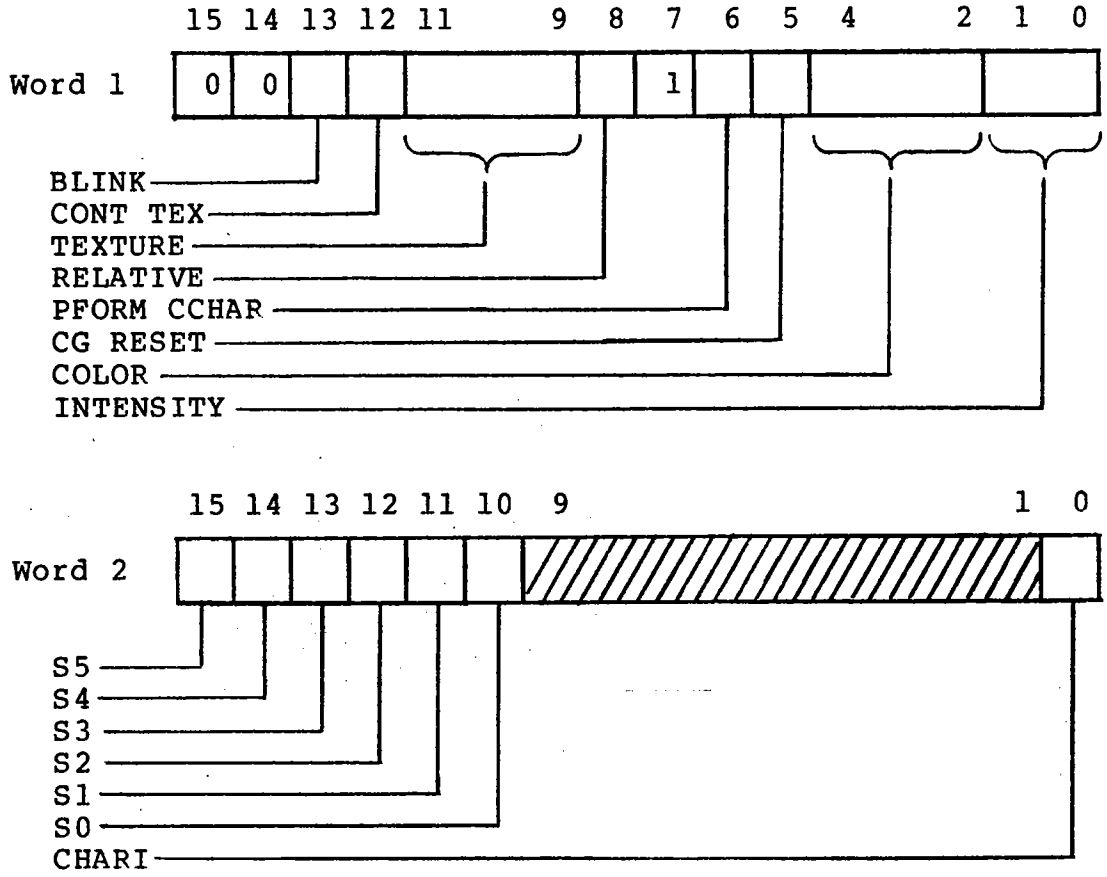
A Draw command to the Line Generator causes a line to be drawn to the X,Y position specified and at an intensity which varies exponentially from the current beam intensity to that specified by the 6 bits of Z information. The texture of the line (i.e., long dashed, long-short dashed, etc.) is determined by the current status of the Line Generator.

Note that if the Line Generator is currently in relative mode, as selected by the RELATIVE bit of the Status command, the X, Y and Z data are interpreted as relative data and the Move or Draw is done relative to the current postion.

	BIT	NAME	FUNCTION
Word 1:	15	none	This bit specifies that this is a Move or Draw command. Bit 15 must be a one.
	14	MOVE/DRAW	The state of this bit determines whether the Line Generator is to position the beam to the coordinates specified or to draw a line to the coordinates specified. MOVE/DRAW=0 for a Move; MOVE/DRAW=1 for a Draw.

	13-12	Z5-4	These bits, in conjunction with bits 15-12 of Word 2, (Z3-0) specify the intensity of this line endpoint.
	11-0	X	These bits specify the absolute 12-bit value of the X coordinate of this line endpoint.
Word 2:	15-12	Z3-0	These bits, in conjunction with bits 13-12 of Word 1 (Z5-4), specify the intensity for this line endpoint.
	11-0	Y	These bits specify the absolute 12-bit value of the Y coordinate of this line endpoint.

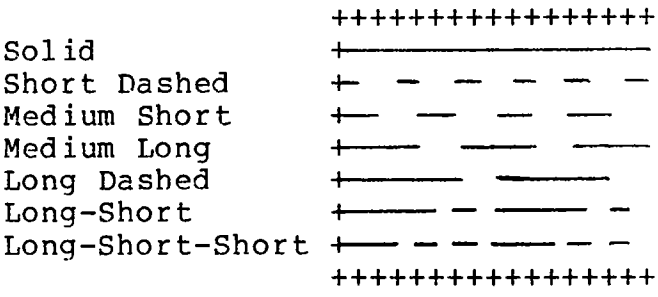
b. Status Command



A Status command to the Line Generator specifies whether to put the Line Generator into blink mode, whether to display line textures as continuous textures regardless of the placement of line segments, the current line texture, the color and intensity that lines are to be displayed, what scopes are to be selected for display, and character status for the Character Generator.

	BIT	NAME	FUNCTION
Word 1:	15,14	none	These bits (in conjunction with bit 7) specify that this is a Status command. Bits 15 and 14 must be zero.
	13	BLINK	This bit is set to indicate that all successive lines and characters are to blink on the display.

12	CONT TEX	CONT TEX (Continuous Texture) is set to indicate that successive textures of dashed lines are to be displayed without regard to individual line endpoints for connected line segments. When this mode is not enabled, then for all line textures the endpoints for every line segment will be displayed. This mode causes a decreased line drawing speed and should be used only when necessary.
11,10,9	TEXTURE	These bits specify how all subsequent lines are to be displayed. Lines may be drawn in any of the following textures:
	000	Solid Lines All subsequent Move commands will cause the beam to be positioned to the specified X,Y coordinates, but not intensified. All subsequent Draw commands will cause a solid line to be drawn to the X,Y coordinates specified.
	001	Dot Mode All subsequent Move commands will cause the beam to be positioned to the X,Y coordinates named and intensified as specified by the Z coordinate. Draw commands are not affected.
	010	Short Dashed All subsequent Draw commands will cause a short dashed line to be drawn to the specified coordinates.
	011	Medium Short Dashed All subsequent Draw commands will cause a medium-short dashed line to be drawn to the specified coordinates.

	100	Medium Long Dashed Lines All subsequent Draw commands will cause a medium-long dashed line to be drawn to the specified coordinates.
	101	Long Dashed Lines All subsequent Draw commands will cause a long dashed line to be drawn to the specified coordinates.
	110	Long-Short Dashed Lines All subsequent Draw commands will cause a line of alternating long and short dashes to be drawn to the specified coordinates.
	111	Long-Short-Short Dashed Lines All subsequent Draw commands will cause a line to be drawn to the specified coordinates which consist of a repeating long-short-short dashed line pattern. The different line textures are shown below: 
8	RELATIVE	When RELATIVE is set, the Line Generator functions in a mode where all MOVE or DRAW commands are relative. The X and Y information is added to the current X and Y position to form the value used for the new beam position. Z is not treated as relative data.
7	none	This bit (in conjunction with bits 15 and 14) specifies that this is a Status command. Bit

7 must be a one.

6	PFORM CCHAR	When PFORM CCHAR (Perform Control Character) is set, it puts the Character Generator into a mode where transfer control characters (ASCII character codes 26 and 27) are performed rather than displayed. This is equivalent to a not-show control character function. (See Section 2.4.4 for details.)
5	CG RESET	When CG RESET (Character Generator Reset) is set, it reinitializes the Character Generator to its power-up state.
4-2	COLOR	These bits specify the color status for the scopes currently selected. The value of these bits specifies the color of all subsequent data drawn. A value of 000, 001 or 010 must be used when a monochrome display is selected. Values 011-111 are used when a beam penetration color monitor is selected. Note that the INTENSITY bits must be appropriately selected for each color selection.

Color	Selected
000	(Monochrome Display)
001	(Monochrome Display)
010	(Monochrome Display)
011	Red
100	Red-Orange
101	Orange
110	Yellow
111	Green

1-0	INTENSITY	These bits specify the relative intensity that all subsequent lines and characters are to be drawn. The gain in intensity is produced by decreasing the drawing speed of the Line Generator. This is normally used in conjunction with

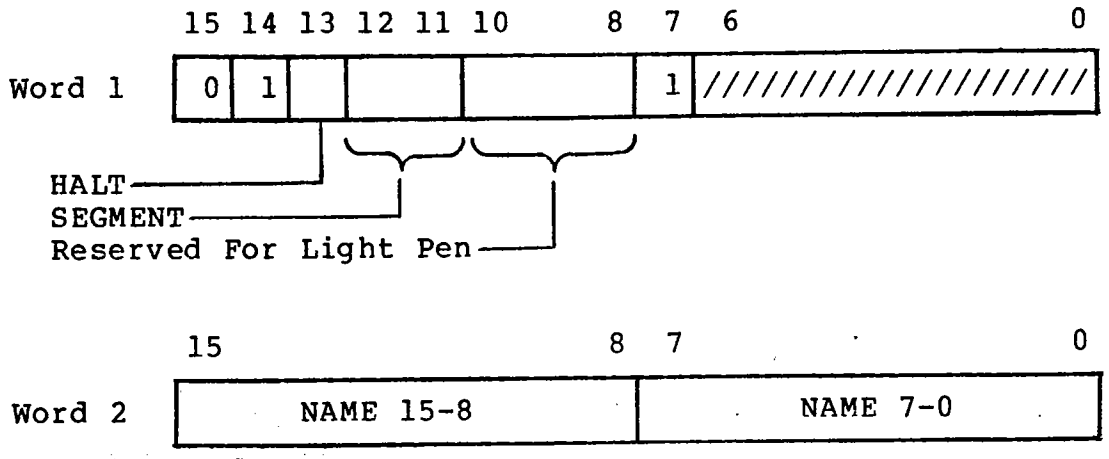
beam penetration monitors to produce lines of equal intensity for the different colors available.

Intensity Selected	
00	normal intensity (used for monochrome display).
01	2x normal intensity (used when selecting Green and Yellow).
10	4x normal intensity (used when selecting Orange and Red-Orange).
11	8x normal intensity (used when selecting Red).

	BIT	NAME	FUNCTION
Word 2:	15	S5	This bit is set to indicate that the Picture Display (Scope 5) will display all data subsequently drawn. This is driven by the cable from backpanel slot 3, channel B.
	14	S4	Same as bit 15, but for Scope 4. This is driven by the cable from backpanel slot 3, channel A.
	13	S3	Same as bit 15, but for Scope 3. This is driven by the cable from backpanel slot 2, channel B.
	12	S2	Same as bit 15, but for Scope 2. This is driven by the cable from backpanel slot 2, channel A.
	11	S1	Same as bit 15, but for Scope 1. This is driven by the cable from backpanel slot 1, channel B.
	10	S0	Same as bit 15, but for Scope 0. This is driven by the cable from backpanel slot 1, channel A.
	9-1	unused	
	0	CHARI	CHARI (Character Intensity) is set to specify that the intensity characters are to be displayed is to be taken from the Intensity field of the Character Generator Font Parameter Stack (see Section 2.4.4.4 for specific details). When CHARI is not set, the intensity characters are displayed is the intensity associated with the last Move or Draw command which positioned for the character string.

how can I delete a segment, or entire one?

c. Refresh Control Command



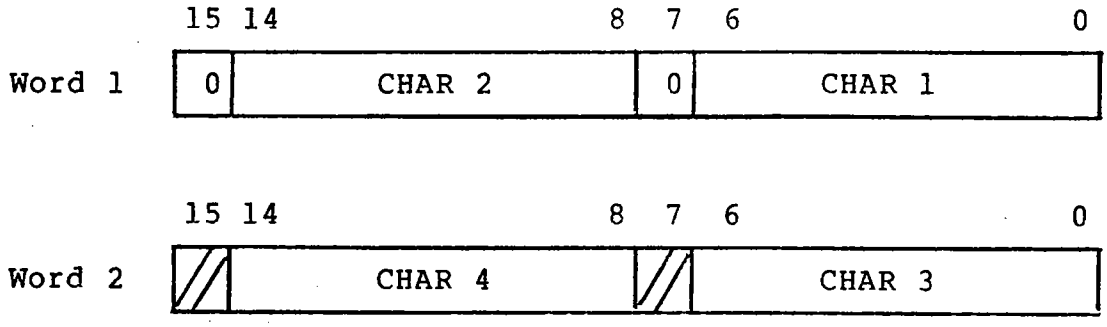
A Refresh Control command causes the function specified by bits 13-11 to be performed. If no function is enabled (bits 13-11=0), these two words will be passed onto the Line Generator Input FIFO and the next two 16-bit words fetched from PSMEM. The Line Generator performs no function upon receiving a Refresh Control command (unless a Light Pen operation is specified as part of the Refresh Control command).

	BIT	NAME	FUNCTION
Word 1:	15-14	none	These bits (in conjunction with bit 7) specify that this is a Refresh Controller command. Bit 15 must be a zero and bit 14 must be a one.
	13	HALT	This bit is used to indicate the end of a refresh segment and that all further input to the Line Generator is to cease. The encounter of a HALT command by the Refresh Controller is equivalent to the RFAIA=RFAIL.
	12-11	SEGMENT	These bits are set to specify a segment name and how that segment name is to be used. The bit combinations and interpretations are as follows:

00	No function.
01	SEGMENT Jump. This command causes the Refresh Controller to jump to the absolute address in PSMEM specified by Word 2. This is done by loading the RFAIA with the contents of Word 2. It should be noted that when a SEGMENT Jump command is encountered by the Refresh Controller when WRITE-BACK mode is enabled, the Jump will not be written back; however, the Jump will be performed, thereby compacting the physically disjoint data segments into a contiguous segment.
10	SEGMENT Name. This command causes the Refresh Controller to load the segment name specified by Word 2 into the RFCSN register. If the SEARCH bit in the RFSR is set, the Refresh Controller compares the RFCSN with the RFSN register which was loaded under program control. If the RFCSN is equivalent to RFSN, a MATCH is declared by the Refresh Controller and the MATCH REQ bit set in the System Interrupt Request register.
11	SEGMENT Name-Blanked. This command functions in the same manner the SEGMENT Name (10) command functions, but sets the Refresh Controller to a mode whereby this command and all subsequent data will not be passed on to the Line Generator for display until the next SEGMENT Name command is encountered. This allows segments to be maintained in the refresh buffer but not displayed.

	10-8		Reserved for Light Pen use.
	7	none	This bit (in conjunction with bits 15 and 14) specifies that this is a Refresh Controller command. Bit 7 must be a one.
	6-0	unused	
Word 2:	15-8	NAME 15-8	These bits constitute the 8 most-significant bits of the Segment Name. The SEARCH MODE bits in the RFSR register may be set so that in searching for a segment name these bits are either masked off or included in the comparison.
	7-0	NAME 7-0	These bits constitute the 8 least-significant bits of the segment name. The SEARCH MODE bits in the RFSR register may be set so that in searching for a segment name the bits are either masked off or included in the comparison. This ability provides subfields within segment names in either the most- or least-significant bytes of the segment name.

d. Character Command



A Character command to the Line Generator causes four character codes to be passed to the Character Generator for interpretation into strokes which are passed back to the Line Generator for display.

	BIT	NAME	FUNCTION
Word 1:	15	none	This bit (in conjunction with bit 7) specifies that this is a Character command. Bit 15 must be a zero.
	14-8	CHAR 2	These bits specify the character to be displayed at the current beam position.
	7	none	This bit (in conjunction with bit 15) specifies that this is a Character command. Bit 7 must be a zero.
	6-0	CHAR 1	These bits specify the character to be displayed at the current beam position.
Word 2:	15	none	This bit is or'ed with the least-significant of the Character Style bits and used with the seven bits of character code to specify the beginning location in the selected Character Memory (PROM or RAM) that the character stroke command begins.

14-8	CHAR 4	These bits specify the character to be displayed at the current beam position.
7	none	This bit is or'ed with the least-significant of the Character Style bits and used with the seven bits of character code to specify the beginning location in the selected Character Memory (PROM or RAM) that the character stroke command begins.
6-0	CHAR 3	These bits specify the character to be displayed at the current beam position.

2.4.4 The Character Generator

The Character Generator is the unit of the Picture Generator that accepts character codes from the Line Generator and interprets these codes, producing strokes (i.e., Line Generator Move and Draw commands). These strokes are then channeled back to the Line Generator for display. There are three basic units of the Character Generator:

1. The Character Control
2. The Character Memory
3. The Character Multiplier

These units function together to cause any of the 95 ASCII characters to be drawn in any of eight standard sizes for regular or italicized characters. The Character Generator may also be programmed to provide many additional sizes and orientations. These units may also cause alternate character fonts and control information to be loaded. The following sections describe the functions of the Character Generator and detail the individual units.

2.4.4.1 Standard Character Sizes and Definitions

The standard Character Generator comes with the ROM Character Memory programmed to interpret the 128 ASCII character set shown in Table 2.4-1 and display the 95 displayable ASCII character subset shown in Table 2.4-2. The ROM scaling coefficients provide eight different sizes of characters in horizontal orientation and eight different sizes in horizontal italicized orientation. The character sizes provided are:

1. .36 cm (.14 inches)
2. .08 cm (.03 inches)
3. .15 cm (.06 inches)
4. .25 cm (.10 inches)
5. .40 cm (.16 inches)
6. .68 cm (.27 inches)
7. 1.14 cm (.45 inches)
8. 1.88 cm (.74 inches)

These sizes give the approximate height of a capital letter (A-Z) based upon a 28.6 x 28.6 cm (11.2 x 11.2 inch) screen viewing area. Size 4 provides 132 characters across a full screen. Size 5 provides 80 characters across a full screen. The Character Generator is initialized to size 1 by RESET. It should be noted that subscript and superscript characters are available only in sizes 3-8. Subscript or superscript character codes (30-33) used when size 1 or 2 is selected will result in invalid character size selection.

Control of the Character Generator is achieved by a set of control characters which are used to regulate character sizes, subscript, superscript, italics, and character line positioning. Control characters and their functions are detailed in Table 2.4-3.

Table 2.4-1
128 ASCII Character Set

ASCII CODE	CHARACTER	ASCII CODE	CHARACTER	ASCII CODE	CHARACTER
000	^P (NULL)	040	space	120	P
001	^A (SOH)	041	!	121	Q
002	^B (STX)	042	"	122	R
003	^C (ETX)	043	#	123	S
004	^D (EOT)	044	$	124	T
005	^E (ENQ)	045	%	125	U
006	^F (ACK)	046	&	126	V
007	^G (BEL)	047	'	127	W
010	^H (BS)	050	(	130	X
011	^I (HT)	051	)	131	Y
012	^J (LF)	052	*	132	Z
013	^K (VT)	053	+	133	[
014	^L (FF)	054	,	134	\
015	^M (CR)	055	-	135	]
016	^N (SO)	056	.	136	^
017	^O (SI)	057	/	137	_
020	^P (DLE)	060	0	140	`
021	^Q (DC1)	061	1	141	a
022	^R (DC2)	062	2	142	b
023	^S (DC3)	063	3	143	c
024	^T (DC4)	064	4	144	d
025	^U (NAK)	065	5	145	e
026	^V (SYN)	066	6	146	f
027	^W (ETB)	067	7	147	g
030	^X (CAN)	070	8	150	h
031	^Y (EM)	071	9	151	i
032	^Z (SUB)	072	:	152	j
033	^[(ESC)	073	;	153	k
034	^\ (FS)	074	<	154	l
035	^] (GS)	075	=	155	m
036	^^ (PS)	076	>	156	n
037	^_ (US)	077	?	157	o
		100	@	160	p
		101	A	161	q
		102	B	162	r
		103	C	163	s
		104	D	164	t
		105	E	165	u
		106	F	166	v
		107	G	167	w
		110	H	170	x
		111	I	171	y
		112	J	172	z
		113	K	173	{
		114	L	174	|
		115	M	175	}
		116	N	176	~
		117	O	177	del

Table 2.4-2
95 Displayable ASCII Characters

ASCII CODE	CHARACTER	ASCII CODE	CHARACTER
040	space	120	P
041	!	121	Q
042	"	122	R
043	#	123	S
044	$	124	T
045	%	125	U
046	&	126	V
047	'	127	W
050	(	130	X
051	)	131	Y
052	*	132	Z
053	+	133	[
054	,	134	\
055	-	135	]
056	.	136	^
057	/	137	_
060	0	140	`
061	1	141	a
062	2	142	b
063	3	143	c
064	4	144	d
065	5	145	e
066	6	146	f
067	7	147	g
070	8	150	h
071	9	151	i
072	:	152	j
073	;	153	k
074	<	154	l
075	=	155	m
076	>	156	n
077	?	157	o
100	@	160	p
101	A	161	q
102	B	162	r
103	C	163	s
104	D	164	t
105	E	165	u
106	F	166	v
107	G	167	w
110	H	170	x
111	I	171	y
112	J	172	z
113	K	173	{
114	L	174	|
115	M	175	}
116	N	176	~
117	O	177	del

Table 2.4-3
Control Character Codes and Functions

ASCII CODE	MNEMONIC	FUNCTION
002	STX	Start of Text. Causes the current character position to be updated as the current top and left margin position.
003	LFT (ETX)	Push-Load Font Stack. Causes the six low-order bits of the next two characters received to be assembled into a 12-bit number. The number is pushed onto the Font Stack as the Intensity, Style and Coefficient parameters. The coefficient offset is set to zero. (See Section 2.4.4.4.)
004	PFT (EOT)	Pop the Font Stack. Causes the Font Parameters to be set to those parameters specified in the last Push-Load Font Stack (LFT) instruction. (See Section 2.4.4.4.)
010	BS	Backspace. Causes the current character position to be updated to the previous character position on the line.
011	HT	Horizontal Tab. Causes the current character position to be updated to the next tab stop on the line. Tab stops are defined by default to be at the following space positions: 8,16,24,32,40,48,....
012	LF	Line Feed. Causes the current character position to be updated to the next line without modifying the relative character position within the lines.
013	VT	Vertical Tab. Causes eight line feeds to be performed.

014	FF	Form Feed. Causes the current character position to be updated to the current top and left margin position.
015	CR	Carriage Return. Causes the current character position to be updated to the current left margin position.
026	SXF (SYN)	Start Transfer. Causes the subsequent characters received (until an End Transfer, 027) to be used as special data for the Character Generator rather than interpreted as character codes. This mode is used to load the RAM Character Memory, and to perform other initialization functions. This operation is only performed if the current Line Generator Status was set with the PFORM CCHAR bit set (see Section 2.4.3b).
027	EXF (ETB)	End Transfer. Causes the mode of the Character Generator to be reset so that all subsequent characters received will be interpreted as characters. Used to reset the mode enabled by SXF (026). This operation is only performed if the current Line Generator Status was set with the PFORM CCHAR bit set (see Section 2.4.3b).
030	SUP (CAN)	Superscript. Causes the current character position to be updated to the raised superscript position and the current character size to be reduced to the next smallest size.
031	RSP (EM)	Reset Superscript. Causes the current character position to be updated to the current line position and the current character size to be increased to the next larger size.

032	SUB (SUB)	Subscript. Causes the current character position to be updated to the lowered subscript position and the current character size to be reduced to the next smaller size.
033	RSB (ESC)	Reset Subscript. Causes the current character position to be updated to the current line position and the current character size to be increased to the next larger size.
034	SIT (FS)	Set Italics. Causes the current size characters to be placed into italics mode whereby all subsequently displayed characters will be slanted to the right.
035	RIT (GS)	Reset Italics. Causes the italics mode to be reset so that characters are displayed in their normal orientation.
036	CUR (RS)	Cursor. Causes a blinking underline to be displayed at the current character position with the current character position unmodified.

2.4.4.2 The Character Control

The Character Control is the unit of the Character Generator that interprets the character codes received from the Line Generator. The Character Control provides for four selectable character styles of which two standard styles are provided. One or two programmable styles may be implemented by the user in the RAM Character Memory and the RAM Font Parameter Stack (see Section 2.4.4.3 for further details). For any style, the interpretation of the characters entails executing the commands contained in the Character Memory for each character, scaling the generated strokes by the Character Multiplier and channelling the scaled strokes to the Line Generator for display.

The Character Generator input and status is controlled by Status commands received by the Line Generator. (See Section 2.4.3b for details.)

2.4.4.3 The Character Memory

The Character Memory consists of two 1024 12-bit word memories which contain the information defining strokes for each of the characters which may be interpreted by the Character Generator. One memory is a 1024-word PROM which has been microcoded to provide the characters and functions described in Section 2.4.4.1, and one memory is a 1024-word Random Access Memory (RAM) which may be loaded with one or two user-defined font descriptions or special symbols.

The Character Memory contains character generation instructions (one per 12-bit word) which define the strokes to be produced for each character. When a character is received by the character control, the character code, in conjunction with the currently-selected font, is used to address the PROM (or RAM). The character generation instructions are then executed until a character termination instruction is executed which causes the Character Control to fetch the next character. Each character is defined as a series of line segments (Moves and Draws) on a 15x15 graph as shown in Figure 2-12.

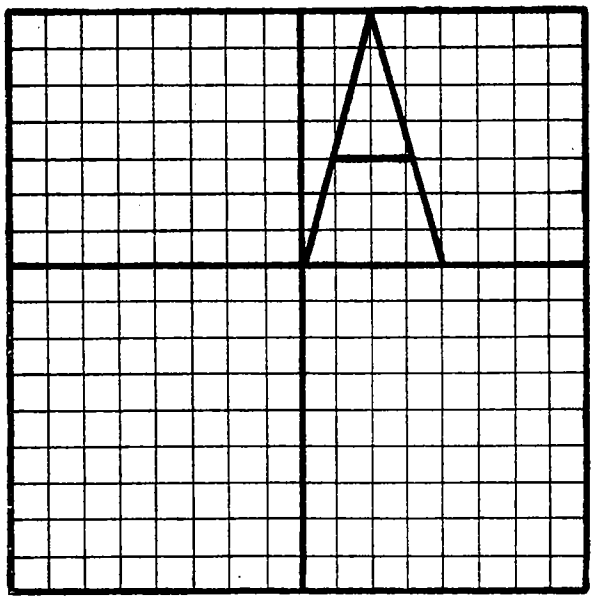
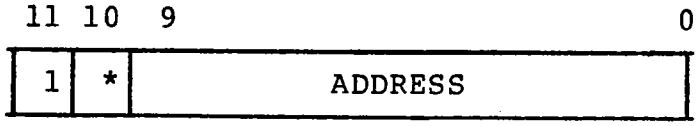


Figure 2-12
Character Generation Definition Graph

The instructions which are used for the character definitions are described as follows:

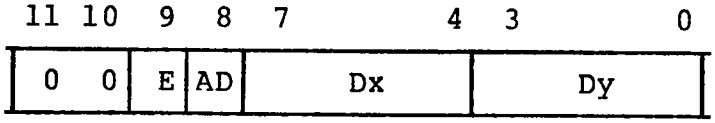
a. JMS



*=0: ADDRESS is within PROM
*=1: ADDRESS is within RAM

The JMS (JuMp Subroutine) instruction causes the next character generation instruction to be taken from the ADDRESS specified (PROM or RAM as indicated by bit 10). The first 128 (or 256) words of each Character Memory contains JMS instructions. When a character code is input, the code (0-127 or 0-255) is used as the starting address in the character memory for the generation of the strokes for the character. This starting address must contain a JMS instruction whose associated ADDRESS is the starting location of the actual stroke generating instruction. The initial JMS (in locations 0-127 or 0-255) is used as a simple jump in this instance rather than a subroutine jump.

b. RMOVE

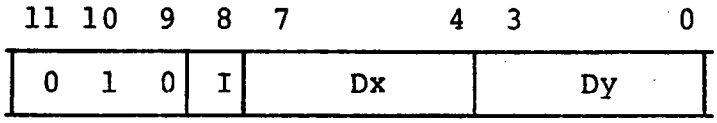


E=0: Not End
E=1: End
AD=0: Do Not Add Displacement to Dx and Dy
AD=1: Add Displacement to Dx and Dy

The RMOVE instruction causes the current character position to be updated to the current position plus the Delta x and y values ($-9<Dx,Dy<8$). If E (End, bit 9) is set, then the current character interpretation is ended. If AD (Add Displacement, bit 8) is set, the current contents of the X and Y Displacement registers (see Section 2.4.4.4) are added to the Dx and Dy values to form the final position for the next character. The RMOVE instruction (with E=1) is used only after all other strokes of the character are processed to perform the final positioning for the next character to be displayed.

Note that when the "Add Displacement" operation is performed, the Displacement values are added directly to internal position registers. The resulting position change on the screen is unaffected by the Character Multiplier parameters (see Section 2.4.4.4).

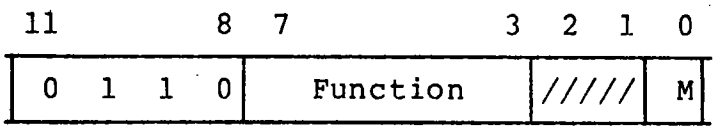
c. CHDRAW/CHMOVE



I=0: CHMOVE (Do Not Intensify)
I=1: CHDRAW (Intensify)

The CHDRAW/CHMOVE instruction is used to specify the individual strokes which compose the character to be generated. This CHMOVE differs from the RMOVE instruction in that the current character position is not updated and is therefore used only for intra-character positioning. During CHDRAW/CHMOVE instructions, the beam position is changed by an analog integrator in the line-generating hardware. The position is not "fixed" to a digitally-controlled position again until an RMOVE instruction is executed. For this reason, more than one RMOVE instruction may be required in a very complex character description to avoid "drift". If I (Intensify, bit 8) is set, a stroke is produced from the current beam position to the relative Delta x and y position (-9<Dx,Dy<8). The character definitions consist mostly of CHDRAW/CHMOVE instructions.

d. CONTROL



Function= Function to be performed (see below)
M= Modifier - Function dependent (see below)

The CONTROL command is used to perform the miscellaneous functions required for the Character Generator to be the powerful and versatile device that it is. The Function field is used to name the particular function to be performed. The Modifier bit is generally used to indicate that the current character generation is complete (End), but may also be used to specify other modification to the function to be performed. These fields are detailed below:

Function	Description
00	Reset Character generator. If M=1, then End.
01	Null. Character Memory no-op. If M=1, then End.
02	Transfer LIMIT to x,y position registers. LIMIT is used as a temporary storage for x,y position. If M=1, then End.
03	Save current character position as left and top margin position (or HOME position). If M=1, then End.
04	Set the current x,y position to the saved HOME position. If M=1, then End.
05	Save current character position as the carriage return position (CR position). If M=1, then End.
06	Save the current x,y position in LIMIT and then set the x,y position to the saved CR position. This instruction, with M=1, must only be used for a carriage return character. For other characters (i.e., the TAB and LINE FEED characters), then M=0). Also, if M=1, then End.
07	Return from subroutine (JMS). Note: subroutines may only be called one level deep. An attempt to JMS within a subroutine will cause the original return address to be destroyed.
10	For M=0: Test REPEAT counter and return from subroutine if REPEAT=-1; otherwise, increment REPEAT and proceed to the next instruction. For M=1: Test REPEAT counter and return from subroutine if REPEAT<-1; otherwise, increment REPEAT and proceed to the next instruction.
11	For M=0: Test REPEAT counter and skip next instruction if REPEAT=-1; otherwise, increment REPEAT and proceed to the next instruction. For M=1: Test REPEAT counter and skip next instruction if REPEAT<-1; otherwise, increment REPEAT and proceed to the next instruction.
12	Skip next instruction if PFORM CCHAR mode is not set.

13 Skip next instruction if the current x,y position does not exceed LIMIT.

14 Skip next instruction and increment REPEAT until REPEAT=-1. One instruction will be skipped for each incrementation of REPEAT.

15 Load REPEAT counter with the next character received by the Character Generator.

16-17 Load REPEAT counter with the contents of bits 3-0 of this instruction.

20 Push the contents of the next two character codes received by the Character Generator onto the Font Parameter Stack and zero the coefficient offset. If M=1, then End.

21 Skip the next instruction in the Character Memory, but push the contents of the skipped word onto the Font Parameter Stack and zero the coefficient offset. If M=1, then End.

22 Pop the top element of the Font Parameter Stack off the stack. (This instruction, with the M bit set to 1, must only appear in the set of instructions which implements the LINE FEED function. Specifically, it must follow the last RMOVE instruction).

23 Increment the Font Parameter Stack stack pointer. If M=1, then End.

24 Increment the Font Parameter Stack pointer, copy the Character Style and Coefficient Address fields from the previous entry in the stack, and zero the Coefficient Offset field. (This instruction must only appear in the set of instructions which implements the Line Feed function. Specifically, it must preceed the first RMOVE instruction).

25 Set blink mode so that all subsequent character strokes generated until the next RMOVE instruction will blink during display. If M=1, then End.

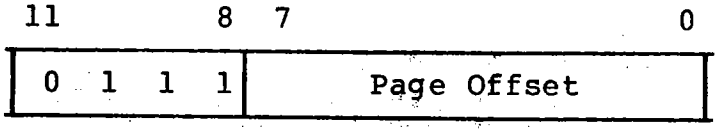
26 Undefined

27 For M=0: The Character Generator is set to a mode where all subsequent characters received will be taken as special data for the Character Generator. This mode is set to allow

the RAM Character Memory, coefficients, internal registers, etc. to be loaded. If PFORM CCHAR is not set, then End.
For M=1: Character Memory no-op.

30-37 Add the value specified by bits 5-0 of this instruction to the coefficient offset of the Font Parameter Stack.

e. JMP



The JMP (JuMp Page) instruction is used to transfer the control of the character generation instructions to the location offset specifed within the current page (Page Offset). Each Character Memory (PROM and RAM) is considered to be four 256-word pages (0-255, 256-511, 512-767, 768-1023). The Page Offset in the JMP instruction specifies the offset from the beginning of the current page. It is not possible to cross page boundaries using the JMP instruction.

2.4.4.4 The Character Multiplier

The Character Multiplier is the unit of the Character Generator that scales and/or rotates the RMOVE and CHDRAW/CHMOVE instruction data taken from the Character Memory. There are three basic units of the Character Multiplier:

1. PROM storage for 16 sets of coefficients.
2. RAM storage for 32 sets of coefficients.
3. A 16-level Font Parameter Stack

The coefficient PROM and RAM are used to contain values by which all character RMOVE and CHDRAW/CHMOVE instructions are multiplied before they are passed to the Line Generator for display. The values in these memories are called sets of coefficients because they represent the sets of coefficients in the following equations. The coefficients may be conveniently represented as elements of a 2x2 matrix as follows:

$$[x\ y] \begin{bmatrix} A & C \\ B & D \end{bmatrix} = [x'\ y']$$

By varying these coefficients, the characters may be scaled to be at any angle of orientation (i.e., horizontal, vertical, etc.), at any degree of slant (for variable italic emphasis) and at up to 64 different sizes (actually 4096 since there are 64 x 64 values available). The following examples illustrate use of the coefficients to produce a) horizontal character orientation at the largest size, b) vertical (counter-clockwise) orientation at the smallest size, and c) italicized characters at mid-range size:

a) A=63, B=0, C=0, D=63

$$[x\ y] \begin{bmatrix} 63 & 0 \\ 0 & 63 \end{bmatrix} = [63x\ 63y]$$

b) A=0, B=1, C=1, D=0

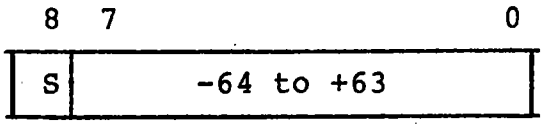
$$[x\ y] \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = [1y\ 1x]$$

c) A=31, B=15, C=0, D=31

$$[x\ y] \begin{bmatrix} 31 & 0 \\ 15 & 31 \end{bmatrix} = [31x+15y\ 31y]$$

The coefficient PROM and RAM are memories containing sets of four 9-bit values which are taken as the A, B, C and D coefficients. The PROM contains 16 sets of coefficients, addressed 0-17. The RAM contains 32 sets of coefficients, addressed 40-77.

Each 9-bit value is interpreted as 1 bit of character speed control, 2 bits of sign and 6 bits of data. It should be noted that the high order sign bit is not used. This provides for up to 64 different character sizes available.



S=Speed Control

The character speed control bits are used to control the actual speed (and therefore size) that character strokes are drawn. The speed control bit (S) must be the same for the A and C coefficients and the B and D coefficients. Thus, these speed control bits form a two-bit field which specifies the following:

B/D	A/C	Meaning
0	0	Normal character speed and size. All characters are drawn with strokes of constant speed at the fastest rate possible. Used for large-sized characters.
0	1	Half normal speed and size. All characters are drawn with strokes of constant speed at half the fastest rate possible. Used for "medium-" sized characters.
1	0	Quarter normal speed and size. All characters are drawn with strokes of constant speed at one-quarter the fastest rate possible. Used for small-sized characters.
1	1	Eighth normal speed and size. All characters are drawn with strokes of constant speed at one-eighth the fastest rate possible. Used for very small-sized characters.

The speed control bits provide for the finer resolution and size selection required to accurately represent small characters. It should be noted that even though characters are

drawn at a slower speed, the hardware compensates to retain the same relative intensity.

The Font Parameter Stack is a 16-level stack which holds the parameters defining the current character font. These parameters are:

1. Character Intensity
2. Character Style
3. Coefficient Address
4. Coefficient Offset

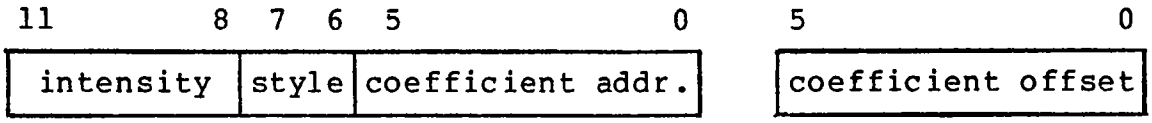
With the Z-Depth Cue feature, the character intensity is specified by 4 bits providing 16 distinct levels of intensity. Characters are normally at the intensity of the last Move or Draw command interpreted by the Line Generator. If the character intensity is to be selected independent of the Move or Draw commands to the Line Generator, the Line Generator status must be updated with the CHARI bit set indicating that the character intensity information should be taken from the Font Parameter Stack (see Section 2.4.3 for specific details of the Line Generator status format).

The character style is specified by 2 bits providing four selectable character styles in the PROM and RAM Character Memories (two styles in each). The most-significant style bit is used to indicate that the PROM (0) or RAM (1) is to be selected. The least-significant of the style bits is used, along with the eight bits of the actual character code, to specify the beginning location in the selected Character Memory (PROM or RAM) of the character stroke commands.

The coefficient address is specified by 6 bits providing 64 addressable coefficient locations--16 in the coefficient PROM and 32 in the coefficient RAM.

The coefficient offset is specified by 6 bits and is added to the coefficient address parameter to form the final coefficient address for this Font Parameter Stack entry. The coefficient offset allows indexing within a basic set of coefficients by adding to or subtracting from the coefficient offset value.

These parameters, described above, define the current character font. The top element of the Font Parameter Stack is always used to select the current character intensity, character style, and coefficients. The CONTROL instructions, previously defined in the Character Memory (Section 2.4.4.2), are used to load and manipulate the contents of the Font Parameter Stack. Each element of the Font Parameter Stack is of the following form:


11
8
7
6
5
0
intensity
style
coefficient addr.
5
0
coefficient offset

2.4.4.5 Detailed Programming of the Character Generator

The Character Generator has many capabilities which are not utilized when the standard PROM Character and Coefficient Memories are used. Among these capabilities are the ability to load the RAM Character and Coefficient Memories, load the inter-character and inter-line spacing registers, to embed comments (i.e., non-displayable character strings) with a string of characters, and to dispatch directly to a location within the PROM or RAM Character Memory (i.e., the ability to produce symbols for which there is no character associated for dispatching). This mode of operation is designated "transfer mode" and is available only when the Character Generator is selected to "Perform Control Character" mode (PFORM CCHAR, see Section 2.4.3b for details). The manner in which transfer mode is used to provide these capabilities is detailed for each of the functions below:

a. Load Character Memory

The RAM Character Memory is loaded by outputting a Line Generator Status Command with the PFORM CCHAR bit set, followed by a string of characters of the following form:

<KAaXxXx...Xx>

where:

< is the Start Transfer character code (SXF,026).

K is a lower or upper case K character code (113 or 153).

Aa are two character codes which are used to designate the beginning address in the RAM Character Memory into which the character generation instructions (XxXx...) are to be loaded. The RAM Character Memory address is formed from Aa in the following manner:

A<6:5:4>=1.
A<3-0>=the 4 most-significant bits of the 10-bit address.
a<6>=1.
a<5-0>=the 6 least-significant bits of the 10-bit address.

For example, to begin loading the RAM from location 0, the following character codes are used:

160,100.

The Character Memory address is incremented after each "Xx" value is loaded, until an End Transfer code is encountered.

Xx are pairs of character codes which specify the value to be loaded into the next location in the RAM Character Memory. The value for each 12-bit memory location is formed from Xx in the following manner:

X<6>=1.
X<5-0>=the 6 most-significant bits of the 12-bit value.
x<6>=1.
x<5-0>=the 6 least-significant bits of the 12-bit value.

For example, to load a "JMS 401" instruction into the RAM, Xx=164,101.

> is the End Transfer character code (EXF,022), or any other character code in the range 0-37.

For example, the seguence of character codes below loads a "JMS 401" instruction into RAM location 6042 and a "RMOV,Dx=5,Dy=0" instruction into location 6043:

26,113,160,142,164,101,101,120,27.

b. Load Coefficient Memory

The RAM portion of the Coefficient Memory is loaded by outputting a Line Generator Status Command with the PFORM CCHAR bit set, followed by a string of characters of the form:

<WapXxXx...Xx>

where:

< is the Start Transfer character code (SXF,026).

W is a lower or upper case W character code (127 or 167).

a is a character code which specifies the Coefficient Memory address where loading is to begin:

A<6>=1.
A<5-0>=Coefficient Memory address in the range 40-77.

The Coefficient Memory address is incremented once for each four "Xx" pairs (typically, once for each set of coefficients A, B, C and D).

p is a character code which specifies which of the coefficients A, B, C or D is to be loaded first. Here, the character to use matches the coefficient name: character A (code 101) for coefficient A, character B for coefficient B, etc. This coefficient pointer "increments" from A through D, then back to A as each "Xx" pair is encountered.

Xx are pairs of character codes which specify the 9-bit values to be loaded for each coefficient.

X<6>=1.
X<5:4:3>=0.
X<2>=speed bit
X<1-0>=the 2 most-significant bits (including sign) of value.
x<6>=1.
x<5-0>=the 6 least-significant bits of value.

> is the End Transfer character code (EXF,027) or any other character code in the range 0-37.

For example, the following sequence of character codes loads Coefficient Memory address 43 with coefficients A-D for a mid-range character at one-half speed:

26,127,143,101,104,177,100,100,104,100,100,177,27.

c. Load Inter-character and Inter-line Spacing Registers

The Displacement registers referred to in Section 2.4.4.3 may be loaded by outputting a Line Generator Status Command with the PFORM CCHAR bit set, followed by a string of characters of the form:

<PaXxXx...Xx>

where:

< is the Start Transfer character code (SXF,026).

P is a lower or upper case P character code (120 or 160).

a is a character code which specifies which register is to be written first:

a<6>=1.
a<5:4>=0.
a<3-0>=register address in range 10-13 for displacement registers.

The 12-bit integer part, and 12-bit fractional part of X Displacement are written to registers 10 and 12, respectively. The integer and fractional parts of Y Displacement are written to registers 11 and 13.

Xx are pairs of character codes which specify the 12-bit values to be loaded into the Displacement registers.

X<6>=1.
X<5-0>=the 6 most-significant bits of 12-bit value.
x<6>=1.
x<5-0>=the 6 least-significant bits of 12-bit value.

> is the End Transfer character code (EXF,027) or any other character code in the range 0-37.

The register address is incremented after each "Xx" pair, to load all four values in one transfer, the values must appear in the order: integer x, integer y, fractional x and fractional y. The following seguence of codes loads the X Displacement with 2.6421 (octal) and the Y Displacement with .37 (octal):

26,120,110,100,102,100,101,164,121,137,100,27.

d. Comments Within Character Strings

Comments may be embedded within strings of characters by including the text of the comments in the form:

<"TEXT...>

where:

< is the Start Transfer character code (SXF,026).

" is the quote symbol character code (42).

TEXT... is any string of character codes in the range 040-177 (i.e., all of the 95 displayable ASCII characters).

> is the End Transfer character code (EXF,027) or any other character code in the range 0-37.

If the comment is preceeded by a Line Generator Status command with the PFORM CCHAR bit set, none of the comment characters are displayed; if the PFORM CCHAR bit is not set, the comment characters are displayed.

e. Direct Dispatch to Character Memory Location

Control may be transferred to a specified Character Memory location by outputting a Line Generator Status Command with the PFORM CCHAR bit set, followed by a string of characters of the form:

<DAa>

where:

< is the Start Transfer character code (SXF,026).

D is a lower or upper case D character code (104 or 144).

Aa is a pair of character codes which specifies the starting Character Memory address:

A<6>=1.
A<5>=0.
A<4-0>=the 5 most-significant bits of 11-bit Character Memory address.
a<6>=1.

a<5-0>=the 6 least-significant bits of 11-bit Character Memory address.

> is the End Transfer character code (EXF,027) or any other character code in the range 0-37.

Once control has been transferred to location "Aa" in the Character Memory, character generator instructions are executed sequentially (or in the order determined by internal jump instructions) until some form of End is encountered (see Section 2.4.4.4). At this time, the control returns to a mode where the incoming character codes are interpreted for display.

The following sequence of character codes transfers control to Character Memory location 1623:

26,104,116,123,27.

2.4.5 Picture Generator Maintenance Registers

The maintenance registers of the Picture Generator provide access to the internal registers and data paths of the Picture Generator. These registers are provided for maintenance only and need not be accessed during normal operations.

The Picture Generator is controlled by three registers in the SCB, shown in Figure 2-13. These registers are described in detail in the following sections.

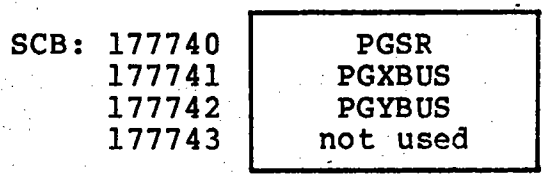
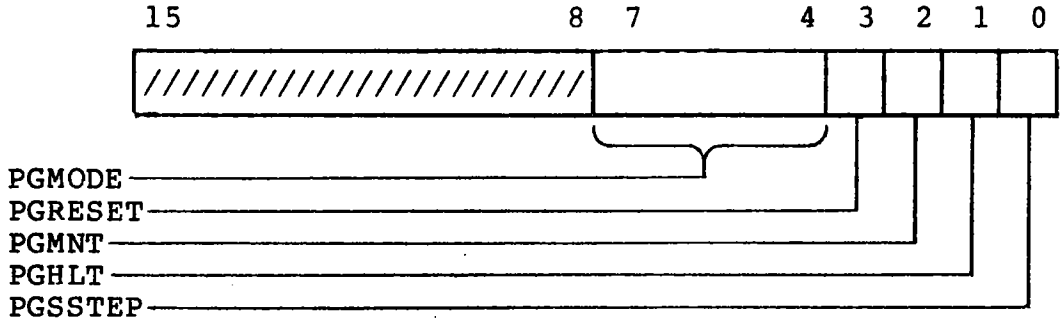


Figure 2-13
Picture Generator Maintenance Registers

a. Picture Generator Status Register (PGSR)-SCB:177740



The PGSR is used to provide maintenance control of the Picture Generator.

BIT	NAME	FUNCTION
7-4	PGMODE	PGMODE is used to select which internal Picture Generator registers can be read via the PGXBUS and PGYBUS registers. (See below). Cleared by RESET.
3	PGRESET	When this bit is set, it causes a RESET signal to be issued to the Line Generator, Character Generator and Refresh Controller. These will be reset to their initial power-up state. Write only. Always reads as a zero.
2	PGMNT	Setting PGMNT, enables the Picture Generator internal registers to be read by reading the PGXBUS and PGYBUS registers. The PGMODE field is used to select which registers are to be accessed. PGMNT must be set for the registers to be accessed. Cleared by RESET.
1	PGHLT	When PGHLT is set, it disables the Picture Generator system clock which normally issues a pulse each 100 nsec. This freezes the Picture Generator for diagnostic purposes. Cleared by RESET.

0	PGSSTEP	This bit is set to issue a single clock pulse to the Picture Generator. Write only. Always reads as a zero.

b. Picture Generator X-Bus (PGXBUS)-SCB:177741

This register is used to read the data on the main internal X-Bus of the Picture Generator. When the Picture Generator is in maintenance mode (PGMNT=1), the internal registers of the Picture Generator connected to the X-Bus may be read using this register and the PGMODE field of the PGSR register and the PGMODE field of the PGSR register. Read only.

The following table details the data that may be read from this register for each of the PGMODE values. If the contents of a bit field is not specified, then it should be considered undefined.

PGMODE values 0-7 apply to the Line Generator. PGMODE values 10-17 apply to the Character Generator.

PGMODE	PGXBUS Register Contents
0	PGXBUS<12-8>=output Control ROM address<4-0>.
1	unused.
2	PGXBUS<15-12>=X normalize code<3-0>. PGXBUS<11-0>=X ALU B input<11-0>.
3	PGXBUS<15>=X sign. PGXBUS<14>=X inverted carry out. PGXBUS<13>=X inverted carry. PGXBUS<12-0>=X ALU F output<12-0>.
4	PGXBUS<11>=carry. PGXBUS<10-8>=sum of squares ROM address<7-5>. PGXBUS<7-0>=Y squared ROM output<7-0>.
5	PGXBUS<11-8>=sum of squares ROM address<4-1>. PGXBUS<7-0>=X squared ROM output<7-0>.
6	PGXBUS<11-8>=integrator count<15-12> complement. PGXBUS<7-0>=multiplier<7-0>.
7	PGXBUS<11-0>=integrator count<11-0>.
10	unused.
11	unused.
12	PGXBUS<7-6>=Style field of Font Parameter Stack. PGXBUS<5-0>=Coefficient Address of Font Parameter Stack (Coefficient Address + Coefficient Offset).
13	PGXBUS<13-12>=Character Intensity<3-2>.

	PGXBUS<11-0>=register at multiplier output.
14	PGXBUS<11-0>=value register.
15	PGXBUS<10-0>=SAVE CHADR register (if EXT=1, nothing).
16	PGXBUS<11-0>=Character Memory output (if EXT=1, SAVE CHADR register).
17	PGXBUS<7-0>=current Microcontrol address.

c. Picture Generator Y-Bus (PGYBUS)-SCB:177742

This register, like the PGXBUS, is used to read the data on the main internal Y-Bus of the Picture Generator. When the Picture Generator is in maintenance mode (PGMNT=1), the internal registers of the Picture Generator connected to the Y-Bus may be read using this register and the PGMODE field of the PGSR register. Read only.

The following table details the data that may be read from this register for each of the PGMODE values. If the contents of a bit field is not specified, it should be considered undefined.

PGMODE values 0-7 apply to the Line Generator. PGMODE values 10-17 apply to the Character Generator.

PGMODE	PGYBUS Register Contents
0	PGYBUS<4-0>=Arithmetic Control ROM Address<4-0>.
1	PGYBUS<4-0>=Multiplier Control ROM Address<4-0>.
2	PGYBUS<15-12>=Y normalize code<3-0>. PGYBUS<11-0>=Y ALU B input<11-0>.
3	PGYBUS<15>=Y sign. PGYBUS<14>=Y inverted carry out. PGYBUS<13>=Y inverted carry. PGYBUS<12-0>=Y ALU F output.
4	PGYBUS<11>=XOUT<11> inverted. PGYBUS<10-0>=XOUT<10-0>.
5	PGYBUS<11>=YOUT<11> inverted. PGYBUS<10-0>=YOUT<10-0>.
6	PGYBUS<7-2>=Z value. PGYBUS<1-0>=unshifted delta Z<4-3>.
7	PGYBUS<7>=unshifted delta Z sign. PGYBUS<6-0>=unshifted delta Z<11-5>.
10	unused.
11	unused.
12	unused.
13	PGYBUS<15-14>=Character Intensity<1-0>. PGYBUS<11-0>=shifted output of multiplier.

14	unused.
15	unused.
16	unused.
17	unused.

The Character and Coefficient Memories of the Character Generator may be read using the maintenance registers by executing special transfer functions similar to those described in Section 2.4.4.5.

The procedure for reading the Character Memory is as follows:

1. Execute the Read Character Memory transfer function (see below), which specifies the first address to be read.
2. Place the Picture Generator into single-step mode by setting the PGHLT bit in the PGSR register.
3. Place the Picture Generator into maintenance mode by setting the PGMNT bit in the PGSR register.
4. Set the PGMODE bits to 16 in the PGSR register. This drives the output from the Character Memory to the Character Generator XCBUS.
5. Single-step the Picture Generator by setting the PGSSTEP bit in the PGSR register.
6. Read the Character Memory output from the PGXBUS register.
7. Repeat steps 5 and 6 to read successive locations in the Character Memory.
8. Return the Picture Generator to normal "run-free" mode by clearing the PGMNT, then the PGHLT bit in the PGSR register.

The Read Character Memory transfer function is executed by outputting a Line Generator Status command with the PFORM CCHAR bit set, followed by a string of characters of the form:

<MAa>

where:

< is the Start Transfer control code (SXF,026).

M is a lower or upper case M character code (115 or 155).

Aa is a pair of character codes which specifies the Character Memory address at which reading is to begin.

A<6>=1.
A<5>=0.
A<4-0>=the 5 most-significant bits of
the 11-bit Character Memory address.
a<6>=1.
a<5-0>=the 6 least-significant bits of
the 11-bit Character Memory address.

> is the End Transfer control code (EXF,027).

The procedure for reading the Coefficient Memories is as follows:

1. Execute the Read Coefficient Memory transfer function (see below) with the parameters set to read the A and C coefficients for the range of coefficient addresses desired.

2. Place the Picture Generator into single-step mode by setting the PGHLT bit in the PGSR register.

3. Place the Picture Generator into maintenance mode by setting the PGMNT bit in the PGSR register.

4. Set the PGMODE bits to 13 in the PGSR registers. This drives the output of the character multiplier logic (which, for this specialized case, is the output from the Coefficient Memory) to the YCBUS.

5. Single-step the Picture Generator twice by setting the PGSSTEP bit in the PGSR register two times.

6. Read the Coefficient Memory output A, B, C or D from the PGYBUS register.

7. Repeat steps 5 and 6 to read successive locations of the Coefficient Memory.

8. Return the Picture Generator to normal "free-run" mode by clearing the PGMNT bit, then clearing the PGHLT bit in the PGSR.

9. Repeat steps 1 through 8, except for substituting parameters for reading the B and D coefficients in step 1.

The Read Coefficient Memory transfer function is executed by outputting a Line Generator Status Command with the PFORM CCHAR bit set, followed by a string of characters of the form:

<]apMm>

where:

< is the Start Transfer character code (SXF,026).

] is the close bracket character code (135).

a is the character code which specifies the address in the Coefficient Memory at which reading is to begin:

a<6>=1.
a<5-0>=Coefficient Memory address.
0-17=Coefficient ROM.
20-37=nothing.
40-77=Coefficient RAM

p is a character code which specifies which Coefficient (A, B, C or D) will be read first. The character here is taken literally; i.e., to begin reading at coefficient A, the character code for upper case A is used (code 101). NOTE: Only half of the coefficients (either all A's and C's, or all B's and D's) may be accessed during the execution of one transfer function. If this p value is "A", then the coefficient values will appear in the following order as the clock is stepped: coefficient A of address a, e.g., A(a), A(a), C(a), C(a), A(a+1), A(a+1), C(a+1), etc. A second transfer function would be required to read the B and D coefficients for the same Coefficient Memory addresses.

Mm is a pair of character codes which specifies the value by which coefficients will be multiplied before appearing on the PGYBUS. This parameter is made necessary by the fact that the only read path to the Coefficient Memory is through the character multiplier. The multiplier value Mm is used in place of the Dx and Dy values which would multiply coefficients if a CHDRAW instruction were being executed (see instruction format in Section 2.4.4.3). Therefore, to read the A or C coefficients, we must have Dx=1, Dy=0 or multiplier Mm=20; to read the B or D coefficients we must have Dx=0, Dy=1 or multiplier Mm=1. The format for the multiplier characters is:

M<6>=1.
M<5-0>=the 6 most-significant bits of
the 12-bit multiplier value.
m<6>=1.
m<5-0>=the 6 least-significant bits of
the 12-bit multiplier value.

2.5 Interrupt Control

As described in Section 2.2, the PICTURE SYSTEM 2/PDP-11 Interface is the only PICTURE SYSTEM device interfaced directly to the UNIBUS of the PDP-11. As such, the DMA channel is under direct interrupt control of the PDP-11 with READY, Interrupt Enable (DMAIE) and Extended Bus Address (XBA) bits available in the IOST register as with all standard DEC NPR device controllers. However, there are many PICTURE SYSTEM 2 devices available for use under interrupt control from the PDP-11 other than the I/O interface. These devices are interfaced to the PICTURE SYSTEM 2 PSBUS and not directly to the UNIBUS of the PDP-11. Three PDP-11 interrupt vectors are used to initiate the appropriate interrupt handler routine for these PICTURE SYSTEM interrupt processes. These three interrupt vectors, in addition to the DMA interrupt vector, are assigned the following priority levels (in descending order) and standard interrupt vectors as follows:

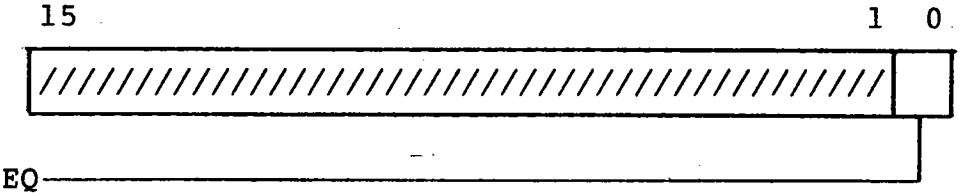
1. Real-Time Clock Interrupt Control: BR5 - 340
2. System Interrupt Control: BR5 - 344
3. Device Interrupt Control: BR5 - 350
4. DMA Interrupt Control: BR5 - 354

The DMA Interrupt Control is described in detail in Section 2.2. Control of the Real-Time Clock, System and Device interrupt processes is supervised by a PICTURE SYSTEM Interrupt Enable bit (PSIE) in the IOST register and individual interrupt enable bits for each of the various interrupt processes. These are described in detail in the following sections.

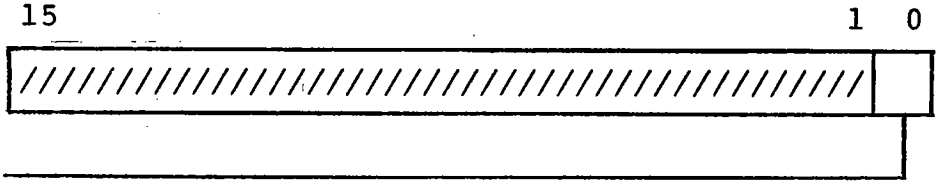
2.5.1 Real-Time Clock Interrupt Control

The Real-Time Clock is used to control refreshing of the PICTURE SYSTEM at line frequency intervals under interrupt control. The Real-Time Clock has two registers in the SCB assigned to regulate this interrupt--the Real-Time Clock Interrupt Request and the Real-Time Clock Interrupt Enable registers.

a. Real-Time Clock Interrupt Request (RTCREQ)-SCB:177760



b. Real-Time Clock Interrupt Enable (RTCIE)-SCB:177761



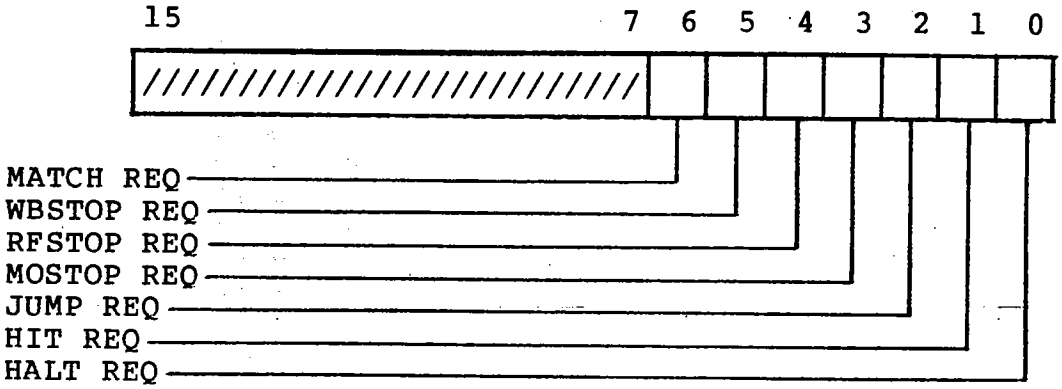
The Real-Time Clock Interrupt Request and Real-Time Clock Interrupt Enable registers have a corresponding bit assigned in each to control the interrupt process for the Real-Time Clock. The RTC IE bit is set in the RTCIE register to allow an interrupt to be issued to the host computer whenever the RTC REQ bit becomes set in the RTCREQ register. Both the RTCREQ and RTCIE registers are cleared by RESET. The conditions which cause a request to be issued are detailed below.

BIT	NAME	FUNCTION
15-1	not used	
0	RTC REQ	RTC REQ is set by the Real-Time Clock controller whenever the line frequency counter 1 (CNT1) has elapsed. If counter 1 (CNT1) is selected for external syncing to a source other than the line frequency, RTC REQ is set when the count elapses. (See Section 2.6.1 for specific details.) RTC REQ must be cleared to acknowledge a RTC interrupt. Read/Write. Cleared by RESET or loading with a 1. Cannot programmably be set to a 1.

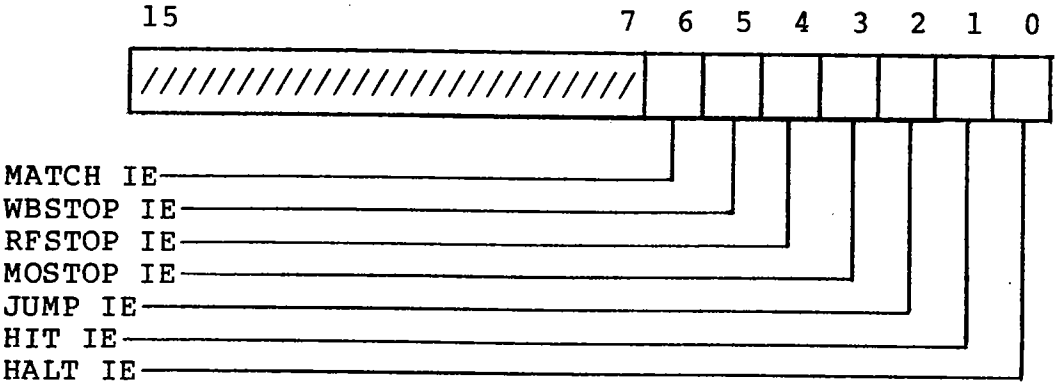
2.5.2 System Interrupt Control

There are several PICTURE SYSTEM functions which have been designated System Control functions and have the ability to cause the processing of data in the PICTURE SYSTEM to cease and an interrupt to be issued to the host computer. These System Control functions have two registers in the SCB assigned to regulate these interrupt processes--the System Interrupt Request and System Interrupt Enable registers.

a. System Interrupt Request (SYSREQ)-SCB:177762



b. System Interrupt Enable (SYSIE)-SCB:177763



The System Interrupt Request and System Interrupt Enable registers have corresponding bits assigned for each System Control function. A bit in the System Interrupt Enable register is set to allow an interrupt to be issued to the host computer whenever the corresponding REQ bit becomes set in the System Interrupt Request register. The bits in the System Interrupt Request register are set by the Picture Processor or Refresh Controller to indicate the presence of a condition which

may require the attention of the control program. Both the SYSREQ and SYSIE registers are cleared by RESET. The conditions which cause a request to be issued are detailed for each of the REQ bits below.

BIT	NAME	FUNCTION
15-7	not used	
6	MATCH REQ	MATCH REQ is set by the Refresh Controller whenever the RFCSN=RFSN and indicates that a segment name match has been encountered. Note that the registers need not be exactly equal depending on the current status of the SEARCH MODE bits of the RFSR. If the MATCH HOLD bit in the RFSR is set when the MATCH REQ is set, the MATCH REQ bit must be cleared before the current refresh cycle will continue. MATCH REQ must be cleared to acknowledge a MATCH interrupt. Cleared by RESET or loading with a 1. Cannot be programmably set to a 1.
5	WBSTOP REQ	WBSTOP REQ (Writeback Stopped) is set by the Refresh Controller to indicate that the RFAWA has reached the limit in PSMEM as determined by the RFAWL, during writeback processing. Writeback may be continued by resetting the RFAWA or the RFAWL and clearing the WBSTOP REQ bit. Writeback may be stopped by clearing the WRITEBACK bit of the RFSR. WBSTOP REQ must be cleared to acknowledge a WBSTOP interrupt. Read/Write. Cleared by RESET or loading with a 1. Cannot be programmably set to a 1.
4	RFSTOP REQ	RFSTOP REQ (Refresh Stopped) is set by the Refresh Controller whenever the RFAIA=RFAIL or a Refresh Con-

trol command with the HALT bit set is encountered by the Refresh Controller. RFSTOP REQ must be cleared to acknowledge a RFSTOP interrupt. Read/Write. Cleared by RESET or loading with a 1. Cannot be programmably set to a 1.

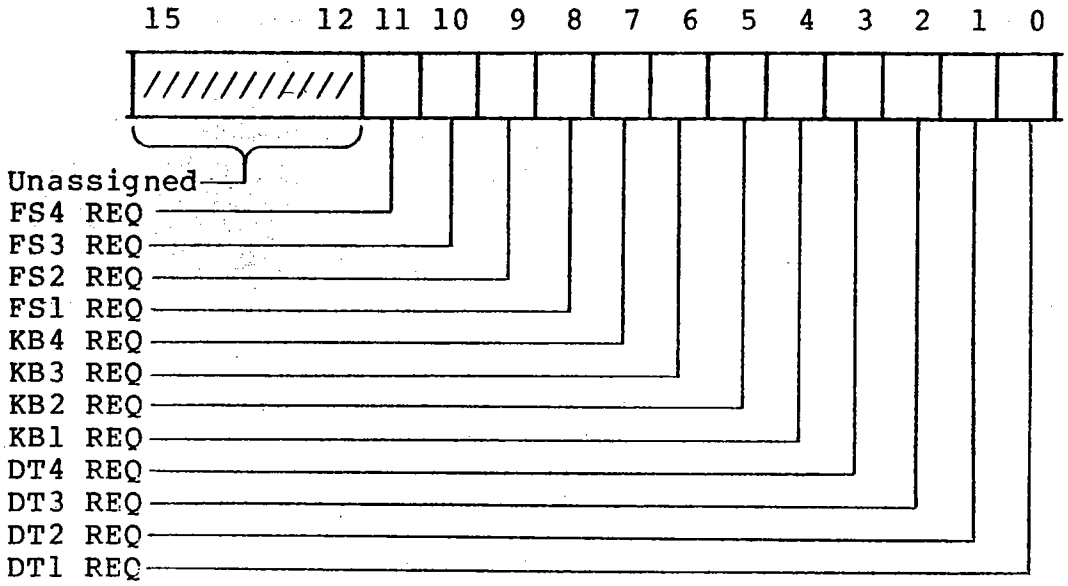
3	MOSTOP REQ	MOSTOP REQ (MAP Output Stopped) is set by the Picture Processor to indicate that all output from the MAP Output Formatter has ceased because the MAOA has reached the limit in PSMEM as determined by the MAOL. Output may be continued by resetting the MAOA or MAOL and clearing the MOSTOP REQ bit. MOSTOP REQ must be cleared to acknowledge a MOSTOP interrupt. Read/Write. Cleared by RESET or loading with a 1. Cannot be programmably set to a 1.
2	JUMP REQ	JUMP REQ is set by the Picture Processor whenever the MAP Input Controller is functioning as a Passive device and the MAP has just executed a JUMP, PUSHJ, or POPJ RSR command. The Picture Processor is in the HOLD state while JUMP REQ is set. When servicing the JUMP interrupt, the control program in the host computer can clear the MAP input FIFO by resetting the MAIA register and then start a new DMA transfer from the host computer memory corresponding to the location specified for the JUMP, PUSHJ or POPJ command. JUMP REQ must be cleared to acknowledge a JUMP interrupt. Read/Write. Cleared by RESET or loading with a 1. Cannot be programmably set to a 1.
1	HIT REQ	HIT REQ is set by the Picture processor whenever a point or line which is being processed

		by the MAP is not entirely clipped from the scene. This indicates that a point or line is within or passes through the current window. HIT REQ must be cleared to acknowledge a HIT interrupt. Read/Write. Cleared by RESET or loading with a 1. Cannot be programmably set to 1.
0	HALT REQ	HALT REQ is set by the Picture Processor whenever the MAP has just executed a HALT RSR command. The Picture Processor is in the HOLD state while HALT REQ is set. HALT REQ must be cleared to acknowledge a HALT interrupt and/or to restart the MAP after a HALT RSR command has been executed. Read/Write. Cleared by RESET or loading with a 1. Cannot be programmably set to a 1.

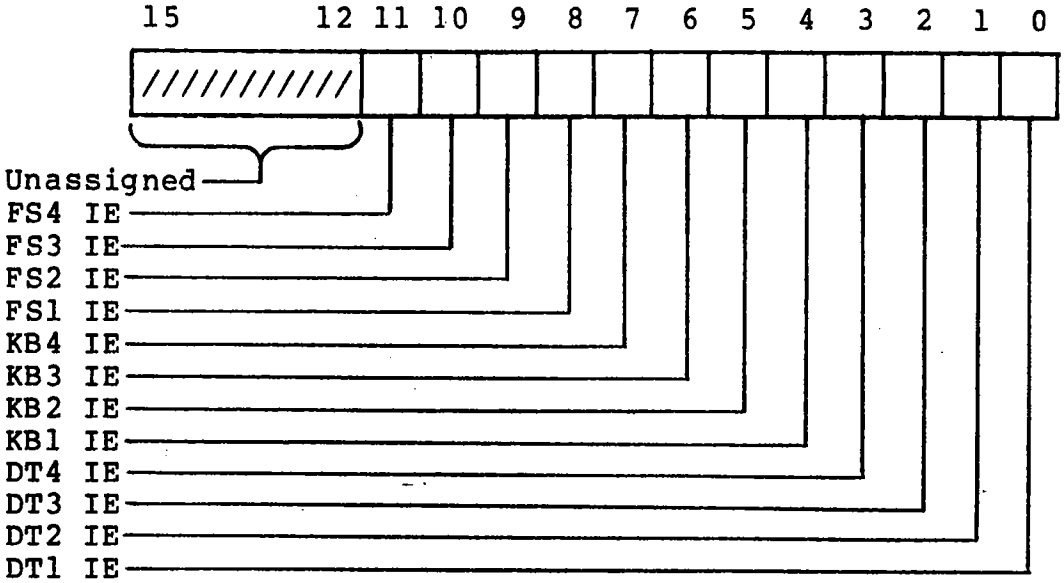
2.5.3 Device Interrupt Control

All other PICTURE SYSTEM devices (i.e., all those but the Real-Time Clock, Picture Processor and Refresh Controller) which may be used under interrupt control are assigned a set of Interrupt Request and Enable bits in the Device Interrupt Control registers. There are four registers in the SCB assigned to regulate device interrupts--two Device Interrupt Request and two Device Interrupt Enable registers.

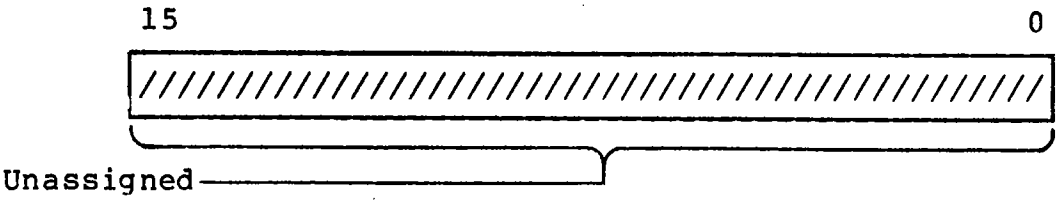
a. Device Interrupt Request 0-15 (DEVREQ 0-15)-SCB:177764



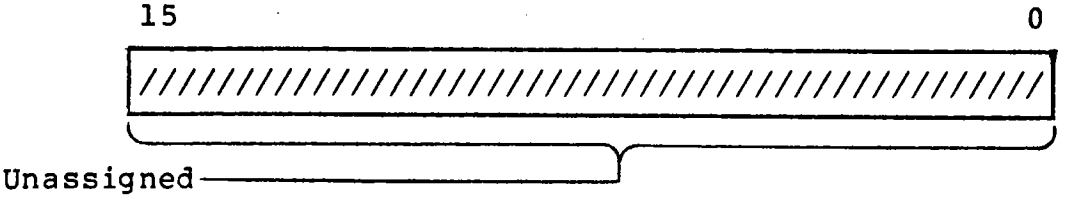
b. Device Interrupt Enable 0-15 (DEVIE 0-15)-SCB:177765



c. Device Interrupt Request 16-31 (DEVREQ 16-31)-SCB:177766



d. Device Interrupt Enable 16-31 (DEVIE 16-31)-SCB:177767



The Device Interrupt Request and Device Interrupt Enable registers have corresponding bits assigned for each PICTURE SYSTEM device which may be used under interrupt control. A bit in a Device Interrupt Enable register is set to allow an interrupt to be issued to the host computer whenever the corresponding REQ bit becomes set in the Device Interrupt Request register. The REQ bits are set by the individual device controllers to indicate the presense of a condition which may require the attention of the control program. The conditions which cause a request to be issued are detailed

for each of the devices in Section 2.6. The devices for which interrupt request and enable bits are currently assigned are the Data Tablet (DT), the Alphanumeric Keyboard (KB) and the Function Switches (FS). Both sets of DEVREQ and DEVIE registers are cleared by RESET.

2.6 PICTURE SYSTEM 2 Devices

All devices which interface to PICTURE SYSTEM 2 are controlled by a set of device registers in the System Control Block. These device registers may be read or written under program control from the host computer by using the Direct I/O interface to the PICTURE SYSTEM. There are five standard device interfaces available for use with PICTURE SYSTEM 2:

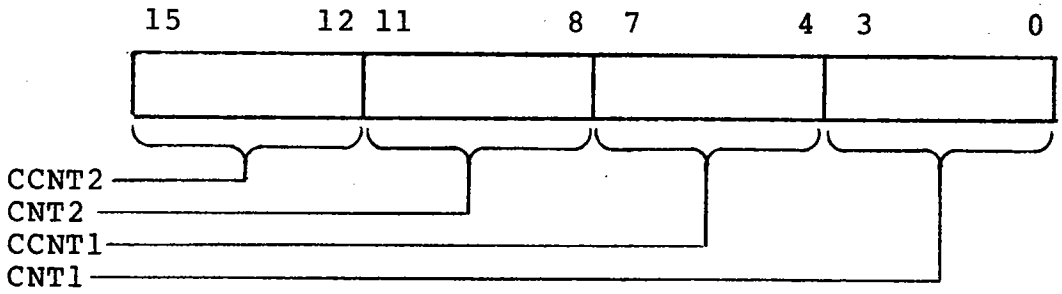
1. Real-Time Clock
2. Data Tablet
3. Alphanumeric Keyboard
4. Function Switches & Lights
5. Control Dials/Joystick

Detailed descriptions for each of these devices and their associated interfaces are contained in the following sections.

2.6.1 Real-Time Clock

The Real-Time Clock is used to provide a source of timing to the host computer allowing synchronization of the display of data with the frequency of the power source (i.e., 60 or 50 Hz.), or with an external source such as a camera synchronization signal. The Real-Time Clock actually provides two clocks (CLOCK1 and CLOCK2) for which twice the line frequency or a user-supplied external input can be selected as the basic timing. Associated with each clock is a 4-bit counter (CNT1 and CNT2) which is used to divide the basic timing providing the final clock frequency. CLOCK1 is used to interrupt the host computer while CLOCK2 is used to provide timing for the automatic refresh feature of the Refresh Controller (see Section 2.4.2 for automatic refresh details). The clocks are controlled by four registers--RTCCNT, RTCSR, RTCREQ and RTCIE. RTCREQ and RTCIE are located in the SCB at 177760 and 177761 as decribed in Section 2.5.1. The RTCCNT and RTCSR are described in detail below:

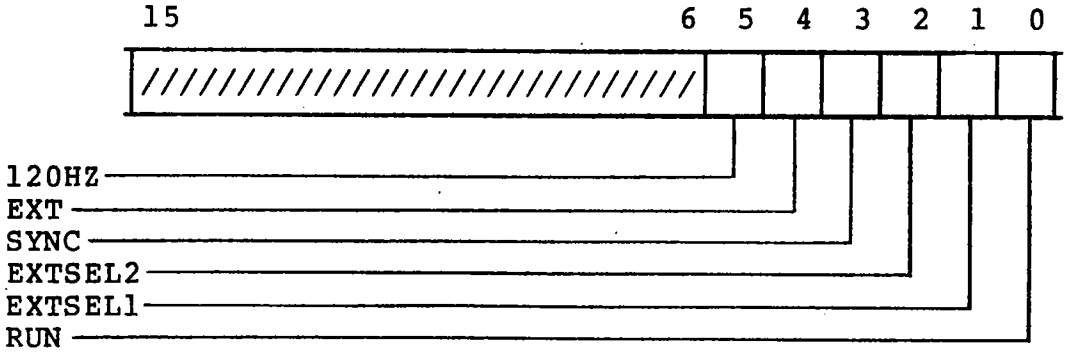
a. Real-Time Clock Counts (RTCCNT)-SCB:177744



The RTCCNT register is used to select the final clock frequencies by selecting the increments by which the basic timing is to be divided. The division is accomplished by incrementing the CCNT values (CCNT2 and CCNT1) at each clock pulse until the count is equal to all ones. The current value of the clock(s) may be read at any time by reading the current count fields (CCNT2 and CCNT1). When either CCNT2 or CCNT1 equals all ones, the field is automatically reloaded with the value in the associated CNT field allowing the clock to continue to run at the same rate without program supervision. When CCNT1 goes to all ones, it sets the RTC REQ bit in the RTCREQ register, issuing an interrupt to the host computer if the RTCIE bit is enabled. When CCNT2 goes to all ones, it initiates an automatic refresh cycle by the Refresh Controller if the AUTOREF bit is set in the RFSR (see Section 2.4.2 for automatic refresh details).

BIT	NAME	FUNCTION
15-12	CCNT2	CCNT2 is a read-only field that reflects the current count of CLOCK2. Incremented by one for each clock pulse and set to the value in CNT2 when CCNT2 equals all ones.
11-8	CNT2	CNT2 defines the value the basic timing of CLOCK2 is to be divided by. Read/Write. Set to 1110 (60 Hz.) by RESET.
7-4	CCNT1	CCNT1 is a read-only field that reflects the current count of CLOCK1. Incremented by one for each clock pulse and set to the value in CNT1 when CCNT1 equals all ones.
3-0	CNT1	CNT1 defines the value by which the basic timing of CLOCK1 is to be divided. Read/Write. Set to 1111 (120 Hz.) by RESET.

b. Real-Time Clock Status Register (RTCSR)-SCB:177745

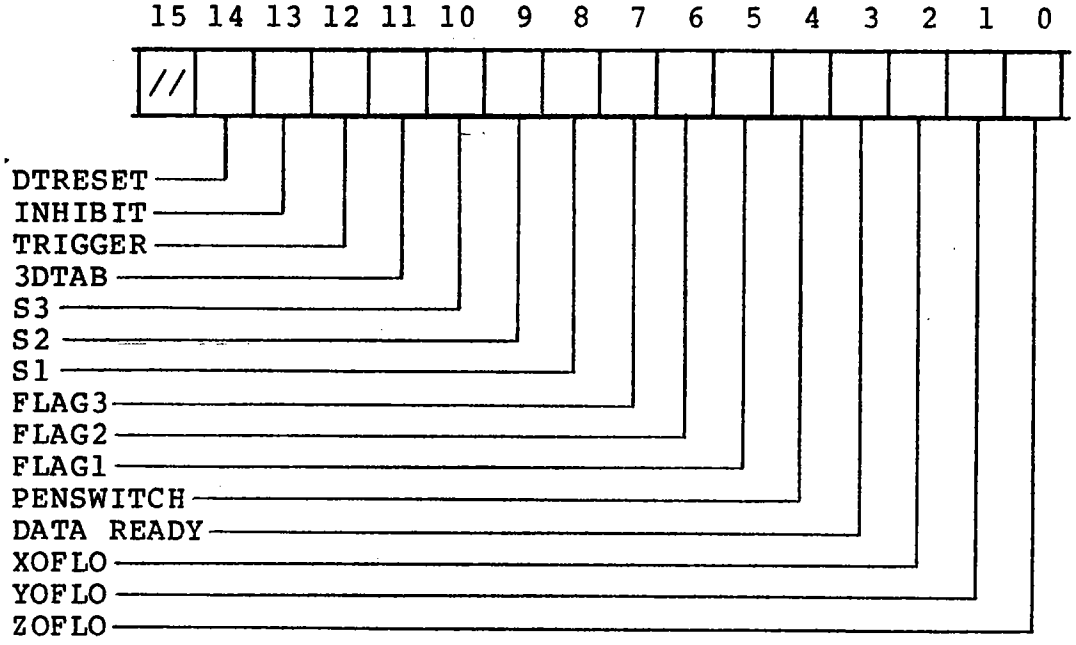


BIT	NAME	FUNCTION
15-6	not used	
5	120HZ	The basic line frequency can be determined from this bit. This bit is driven by a signal which is twice the line frequency. The duty cycle of the signal is such that the bit is clear for approximately 10% of the cycle. This bit stays clear for a short duration so that it will usually be read as a one. Used for diagnostic purposes. Read only.
4	EXT	The basic external timing source can be read by this bit. This is the basic PICTURE SYSTEM one-bit interface for such things as movie camera interfaces. Read only.
3	SYNC	SYNC causes automatic refresh timing to be taken from CLOCK1 so that interrupt and automatic refresh timing can be synchronized as well as run at the same rate. Read/Write. Cleared by RESET.
2	EXTSEL2	Selects external basic timing for CLOCK2. Read/Write. Cleared by RESET.
1	EXTSEL1	Selects external basic timing for CLOCK1. Read/Write. Cleared by RESET.
0	RUN	RUN is set to allow the individual clock pulses from CLOCK1 to increment the CLOCK1 counter CCNT1. Read/Write. Cleared by RESET.

2.6.2 Data Tablet

The Data Tablet is controlled by four data registers and two interrupt control registers in the SCB. The Interrupt Request and Interrupt Enable registers are described in Section 2.5.3. The data registers are the Data Tablet Control and Status Register and the X, Y and Z Data registers detailed below:

a. Data Tablet Status Register (DTSR)-SCB:177664



The Data Tablet interface has been flexibly designed so that interfacing with many different types of tablets (i.e., magnetic, electrostatic, two- or three-dimensional, etc.) is possible. PICTURE SYSTEM 2 typically is supplied with a Summagraphics Data Tablet. For this tablet, up to five milliseconds are required to convert pen positional information to coordinate data. This conversion can be initiated periodically by the tablet controller (stream mode) or by the user's program (remote mode). Failure of a conversion is a result of the setting of any overflow bit, indicating that the pen position relative to that axis could not be determined. A switch selectable option on the interface determines under what conditions the tablet can be made to request an interrupt (i.e., to set the Data Tablet Request bit, DT REQ, in the Device Interrupt Request register). The DTREQ bit is normally set at the successful or unsuccessful completion of a tablet

conversion. Optionally, the tablet interface is DIP switch-selectable to allow the setting of the DTREQ bit only at the successful completion of a tablet conversion.

BIT	NAME	FUNCTION
15	not used	
14	DTRESET	Reset the tablet interface and controller. Write only. Always reads as a zero.
13	INHIBIT	Suspend initiation of tablet data conversions from any source (remote trigger or stream mode).
12	TRIGGER	Initiate a data tablet controller conversion (used with tablet in remote mode). Write only. Always reads as a zero.
11	3DTAB	Switch settable status bit used to denote that the tablet interface is to be used for a 3D tablet. This enables the setting of the ZOFLO bit and causes Z DATA value to be updated in the Data Tablet Z Data (DTZDAT) register.
10-8	S3-S1	Switch settable status bits used to denote individual tablet characteristics. Read only.
7-5	FLAG3,2,1	Status bits associated with tablet options (cursor with multiple switches, etc.). Read only.
4	PENSWITCH	Reads 1 if the pen switch is closed (pen is down), 0 if open (pen is up). Read only.
3	DATA READY	Set when successful conversion is complete (no overflow). Causes data values to be updated and inhibits further data conversions until the DTREQ bit is acknowledged. This assures the values in the data

		registers are a result of the same conversion. Cleared by acknowledging the DTREQ bit or RESET.
2	XOFLO	Set if the pen was out of range in respect to the X-axis when a data conversion was completed. Read only. Cleared by initiation of a new conversion or RESET. Note: Completion of a data conversion attempt results in setting the DATA READY bit (3) or one of the X, Y or Z overflow bits (2-0).
1	YOFLO	Set if the pen was out of range with respect to the Y-axis when a data conversion was completed. Read only. Cleared by initiation of a new conversion or RESET.
0	ZOFLO	Set if the pen was out of range with respect to the Z-axis when a data conversion was completed. This bit will be set only if a 3D tablet is being used. Read only. Cleared by initiation of a new conversion or RESET.

b. Data Tablet X Register (DTXDAT) - SCB:177665

This register contains a number which represents the pen X position relative to the origin of the tablet.

c. Data Tablet Y Register (DTYDAT) - SCB:177666

This register contains a number which represents the pen Y position relative to the origin of the tablet.

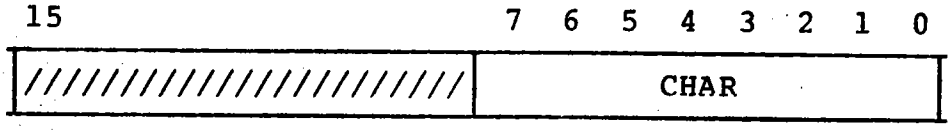
d. Data Tablet Z Register (DTZDAT) - SCB:177667

This register contains a number which represents the pen Z position relative to the origin of the 3D tablet. This register is used only when a 3D data tablet is interfaced to the system.

2.6.3 Alphanumeric Keyboard

The standard PICTURE SYSTEM 2 Alphanumeric Keyboard is a 61-key, 128-character keyboard. Each time a key is depressed, the associated KB REQ bit in the DEVREQ 0-15 register is set, generating an interrupt request to the Picture Controller if the associated KB IE bit is set in the DEVIE 0-15 register. (See Section 2.5 for specific interrupt control details.)

a. Keyboard Data Register (KBDR) - SCB:177620



BIT	NAME	FUNCTION
7-0	CHAR	The ASCII code for the character which was most recently typed may be read from this register. If interrupts are enabled and a keystroke occurs, a device interrupt will be generated.

2.6.4 Function Switches & Lights

The standard PICTURE SYSTEM 2 Function Switches & Lights option provides a box with 16 switches whose setting (set/not set) which may be read under program control. The box also provides 16 lights which may be individually set or cleared under program control. The transition of any switch (0 to 1 or vice versa) will cause the associated FS REQ bit to be set in the DEVREQ 0-15 register, generating an interrupt request to the Picture Controller if the associated FS IE bit is set in the DEVIE 0-15 register. (See Section 2.5 for specific interrupt control details.) There are two data registers, FSWR and FSLR, which provide control and status information.

a. Function Switch Register (FSWR)-SCB:177610

   Bits 15-0 correspond to the setting of switches 15-0 on the function switch The corresponding bits are 1 if the switch is set, 0 if not set. Read only.

b. Function Switch Light Register (FSLR)-SCB:177611

   Bits 15-0 correspond to the status of lights 15-0 on the function switch box. The corresponding bits are 1 if the light is to be on and 0 if off. Read/Write.

2.6.5 Control Dials/Joystick

Both the Control Dial and Joystick options are supported by an analog-to-digital converter which is controlled by registers comprising a control and status register (ADSR) and data registers.

2.6.6 User Devices

All devices previously detailed throughout this chapter are for the standard PICTURE SYSTEM 2. There are, however, certain instances in which special devices are required for a given application. The architecture of PICTURE SYSTEM 2 facilitates the interfacing of these "user devices" by providing 18 device slots in each PICTURE SYSTEM 2 backpanel into which standard and user devices may be inserted. Each device slot is pre-wired to provide all interfacing signals (i.e., interrupt control, address selection, etc.) necessary to interface a device.

There are three types of devices:

1. Passive Devices

   (Real-time Clock, Data Tablet, Alphanumeric Keyboard, Function Switches & Lights, Control Dials/Joystick)

2. Passive Devices with a Passive Input or Output Port

   (Picture Controller Interface, Line Generator Input Controller, MAP Output Formatter, MAP Input Controller)

3. Active Devices

   (Picture Controller Interface, Refresh Controller, MAP Output Formatter, MAP Input Controller)

These types of devices are listed in increasing order of complexity. Type 1 typically requires only 1 device slot; types 2 and 3 usually require 2 device slots. Types 2 and 3 are more complex as they require interfacing to one of the eight PICTURE SYSTEM data channels of the PSBUS. Each of these data channels are capable of supporting a Passive Device with a Passive Input or Output Port and an Active Device. Thus, there may be a maximum of 8 Type 2 devices and 8 Type 3 devices. Other than device slot allocation in the PICTURE SYSTEM 2 backpanel, there is no maximum number of Type 1 devices.

PICTURE SYSTEM 2 has the following standard device assignments:

Device Name	Device Type	Number of Device Slots	PSBUS Data Channel Assignment
Real-Time Clock	1	1	none
Data Tablet	1	1	none
Alphanumeric Keyboard	1	1	none
Function Switches & Lights	1	1	none
Control Dials/Joystick	1	1	none
Picture Controller Interface	2,3	2	DIO-0 (Type 3) DMA-4 (Type 2,3)
LG Input Controller	2	0*	5
Refresh Controller	3	0*	5
MAP Output Formatter	2,3	0*	6
MAP Input Controller	2,3	0*	7

Thus, the standard PICTURE SYSTEM 2 utilizes 7 of the 18 available device slots and 5 of the 8 available PSBUS data channels. This provides 11 device slots and 3 PSBUS data channels available for user devices.

*These devices are interfaced to special device slots in the PICTURE SYSTEM 2 backpanel and therefore do not require one of the 18 standard device slots.

2.7 Map of the System Control Block (SCB)

The System Control Block contains all the PICTURE SYSTEM 2 device control and maintenance registers as detailed throughout this chapter. Table 2.7-1 is a map of all locations in the SCB that are assigned or reserved for PICTURE SYSTEM use.

Table 2.7-1
System Control Block

From	To	Device	Register Name	Remarks
177400-177477		unused		
177500-177607		reserved for Control Dials/Joysticks		
177610-177625		FSL2-8		Function Switches & Lights
177626		FSL1	FSWR	
177627		FSL1	FSLR	
177620-177626		KB2-8	KBDATA	Keyboards
177627		KB1		
177630-177663		DT2-8		Data Tablets
177664		DT1	DTSR	
177665		DT1	DTXDAT	
177666		DT1	DTYDAT	
177667		DT1	DTZDAT	
177670-177677		RF2	same as RF1	Refresh Controller
177700-177703		PG2	same as PG1	Picture Generator
177704-177707		unused		
177710-177717		PP2	same as PP1	Picture Processor
177720-177723		reserved for Light Pen		
177724-177727		unused		
177730		RF1	RFCN	Refresh Controller
177731		RFC1	RFSN	
177732		RF1	RFAWA	
177733		RF1	RFAWL	
177734		RF1	RFAIA	
177735		RF1	RFASA	
177736		RF1	RFAIL	
177737		RF1	RFSR	
177740		PG1	PGSR	Picture Generator
177741		PG1	PGXBUS	
177742		PG1	PGYBUS	
177743		unused		
177744		RTC	RTCCNT	Real-Time Clock
177745		RTC	RTCSR	
177746		DMA2	DMAPSA	DMA2 PS Address
177747		DMA1	DMAPSA	DMA1 PS Address
177750		PP1	MAOL	Picture Processor
177751		PP1	MAOA	
177752		PP1	MAIA	
177753		PP1	MSR	
177754		PP1	MMSR	
177755		PP1	MMRSR	
177756		PP1	MMPAR	
177757		PP1	MMBUS	
177760		RTC	RTCREQ	Real-Time Clock Interrupt
177761		RTC	RTCIE	
177762		System	SYSREQ	System Interrupts
177763		System	SYSIE	
177764		Devices	DEVREQ 0-15	Device Interrupts

177765	Devices	DEVIE 0-15	
177766	Devices	DEVREQ 16-31	
177767	Devices	DEVIE 16-31	
177770	DMA1	DMAPIP	DMA Passive Input Port
177771-177774	reserved for options using		Passive Ports
177775	PG1	LGPIP	LG Passive Input Port
177776	PP1	MPOP	MAP Passive Output Port
177777	PP1	MPIP	MAP Passive Input Port

CHAPTER THREE

3. PICTURE SYSTEM 2 SOFTWARE DETAILS

PICTURE SYSTEM 2 is a highly modular system which is used in a variety of configurations and applications. This chapter presents an overview of how the system may be utilized by the Graphics Software provided with PICTURE SYSTEM 2 for the PDP-11 minicomputer.

3.1 System Concepts

As presented in Chapter 1 and detailed in Chapter 2, PICTURE SYSTEM 2 consists of the following major components and subunits:

1. Picture Controller Interface
   a. Direct I/O (DIO)
   b. Direct Memory Access (DMA)
2. Picture Processor
   a. MAP Input Controller
   b. Matrix Arithmetic Processor
   c. MAP Output Formatter
3. PICTURE SYSTEM Memory
4. Picture Generator
   a. Line Generator
   b. Character Generator
   c. Refresh Controller
5. Picture Display
6. PICTURE SYSTEM Devices
   a. Real-Time Clock
   b. Data Tablet
   c. Alphanumeric Keyboard
   d. Function Switches and Lights
   e. Control Dials/Joystick
   f. User Devices

PICTURE SYSTEM 2, as a high-performance computer graphics system, comes standard with items 1-5, 6a and 6b as listed above. However, for certain applications, it may be desirable to utilize less than the total system available. This section describes the conventions and assumptions made by the system software to support this capability.

3.1.1 Initialization

During PICTURE SYSTEM 2 initialization, it is the responsibility of the software to either assume or determine the particular configuration of hardware components available and to connect the various Active and Passive devices together for the current configuration. For the standard PICTURE SYSTEM, this entails configuring in the manner shown in Figure 3-1.

As this figure indicates, the DMA (as an Active device) is set to channel all data directly to the MAP Input Controller (as a Passive device) for processing by the MAP. The MAP Output Formatter is set to function as an Active device, outputting data from the MAP to the area of PICTURE SYSTEM Memory designated as a refresh buffer. The refresh buffer area in PSMEM is read by the Refresh Controller (an Active device) and the data is then channeled to the Line Generator Input Controller for display. During the remainder of the PICTURE SYSTEM session, these hardware component connections typically remain the same, with the DIO path used for all miscellaneous data transfers to and from the PICTURE SYSTEM.

If the software determines there is no PICTURE SYSTEM Memory available to be used as a refresh buffer, or there is no Refresh Controller available to be used to read the data from the refresh buffer, the MAP Output Formatter is set to channel the data directly to the Line Generator Input Controller for display. This configuration does not perform digital picture buffering and the entire frame must be recreated for each refresh. This configuration is shown in Figure 3-2.

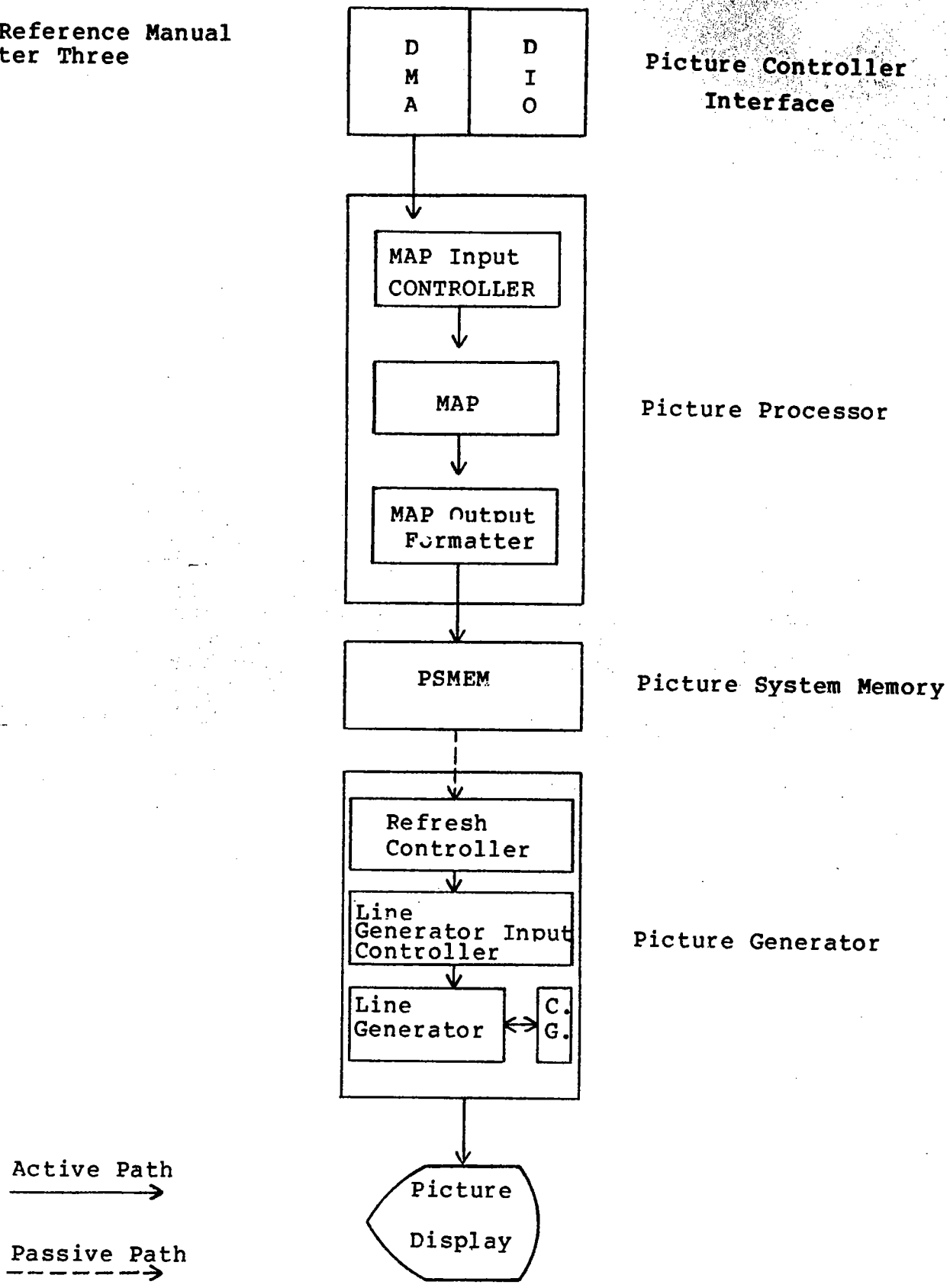


Figure 3-1

Standard PICTURE SYSTEM 2 Usage Configuration

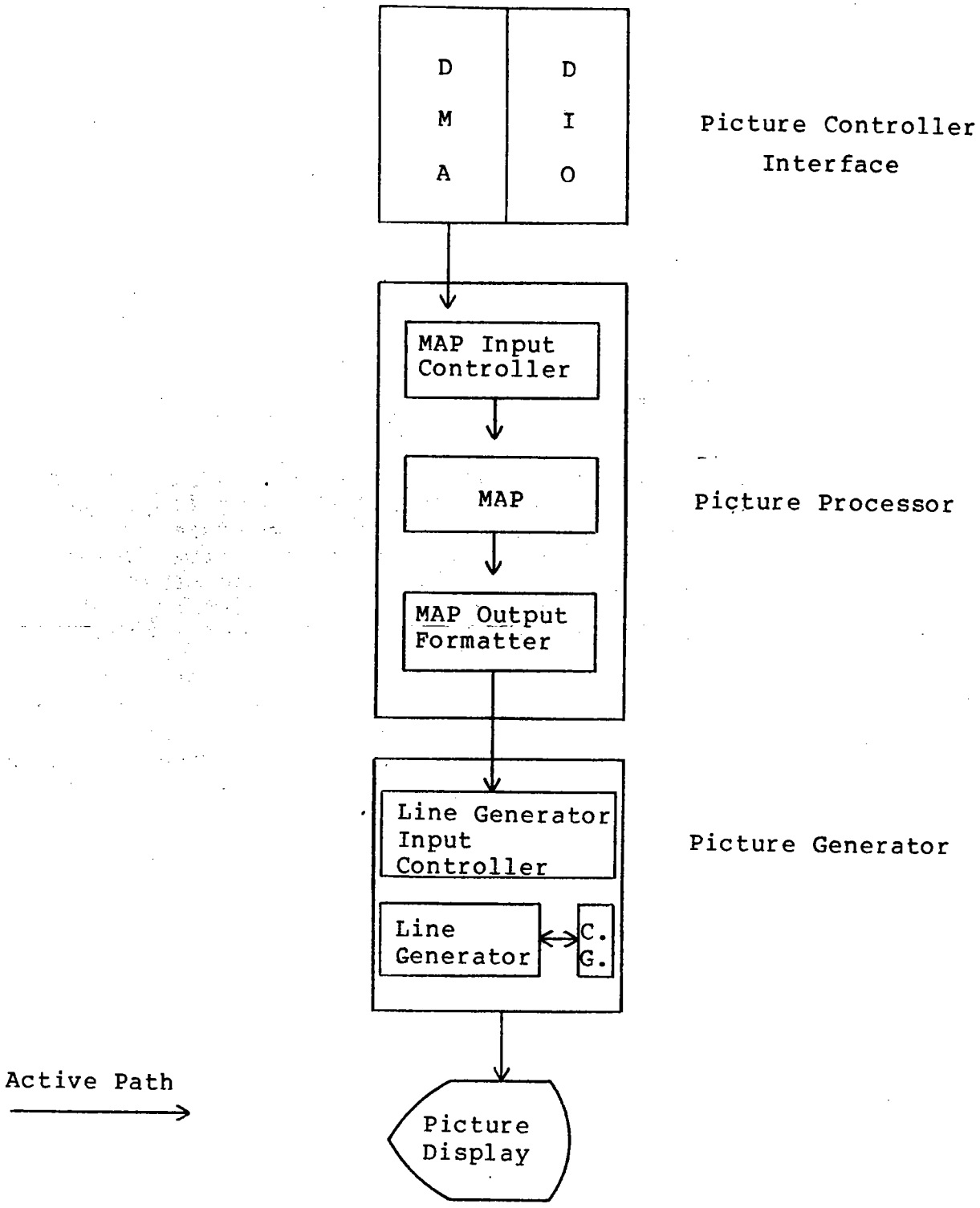


Figure 3-2
PICTURE SYSTEM 2 Usage Configuration
for systems with no PICTURE SYSTEM Memory
and/or no Refresh Controller

Initialization of the system takes advantage of the following PICTURE SYSTEM 2 hardware features:

1. Data which is directed to the Passive Input Port of a device which does not exist is simply "thrown away". This allows, for instance, the initialization of the Line Generator or Picture Processor even though one is not present in the system.

2. Character data which is directed to the Line Generator is simply "thrown away" if the Character Generator is not present. This allows Character Generator initialization (i.e., loading alternate character fonts, etc.) to be performed regardless of whether or not the Character Generator is present in the system.

3. Certain devices have read-only bits available in their status register which indicate the setting of a DIP switch on an interface card. These switches may be set to indicate the type of device which is installed. This may be used to dynamically determine system parameters, such as data tablet size (11" x 11", 36" x 48"), etc.

Thus, the software initialization philosophy is to assume a maximum PICTURE SYSTEM 2 configuration and then dynamically alter this configuration assumption based upon what hardware components are available. This adds very little overhead to the initialization routine but provides the capability of modifying the PICTURE SYSTEM configuration (adding more PICTURE SYSTEM Memory, for example) without relinking existing PICTURE SYSTEM 2 programs.

3.1.1.1 Picture Processor Initialization

Initialization of the Picture Processor consists of loading the initial contents of the 256 MAP Internal Registers (see Section 2.3.2.1) and selecting the MAP Output Formatter as an Active device to output data formatted for display (Display mode).

The structure of the MAP Internal Registers is shown in Figure 3-3. As shown in this figure, there are only 32 (40 octal) registers dedicated for use by the MAP during the processing of data. The remaining registers (40-377), may be used in the manner required by the software control program. This area may be partitioned into 16 register blocks and used as Matrix Stack area, thus providing the capability of storing up to 14 matrices within these registers. These registers can, however, be partitioned in other ways, depending upon the application in which the Picture Processor is to be used.

For general use by the PICTURE SYSTEM 2 software, the remaining registers (40-377) are partitioned as shown in Figure 3-4. As this figure shows, Registers 0-37 are dedicated for use by the MAP. Registers 40-117 are established to allow rapid transformation of a cursor with minimal software overhead. The way in which this is used is as follows: When a cursor is to be displayed, Registers 0-37 and 40-77 are "swapped" using the XFER RSR control command. This saves the current Viewport and Save registers and loads the cursor values into the dedicated registers. The cursor is then drawn and Registers 0-37 and 40-77 "swapped" again restoring the system to its previous state.

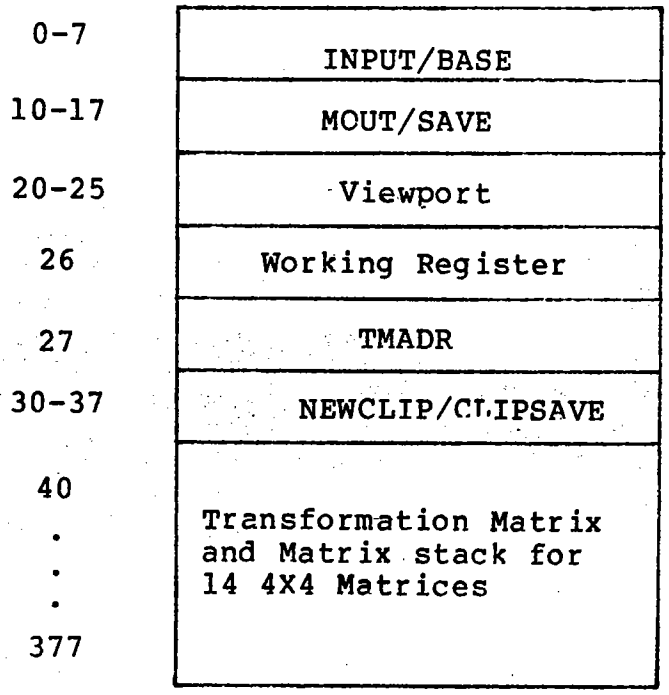


Figure 3-3

MAP Internal Register Structure

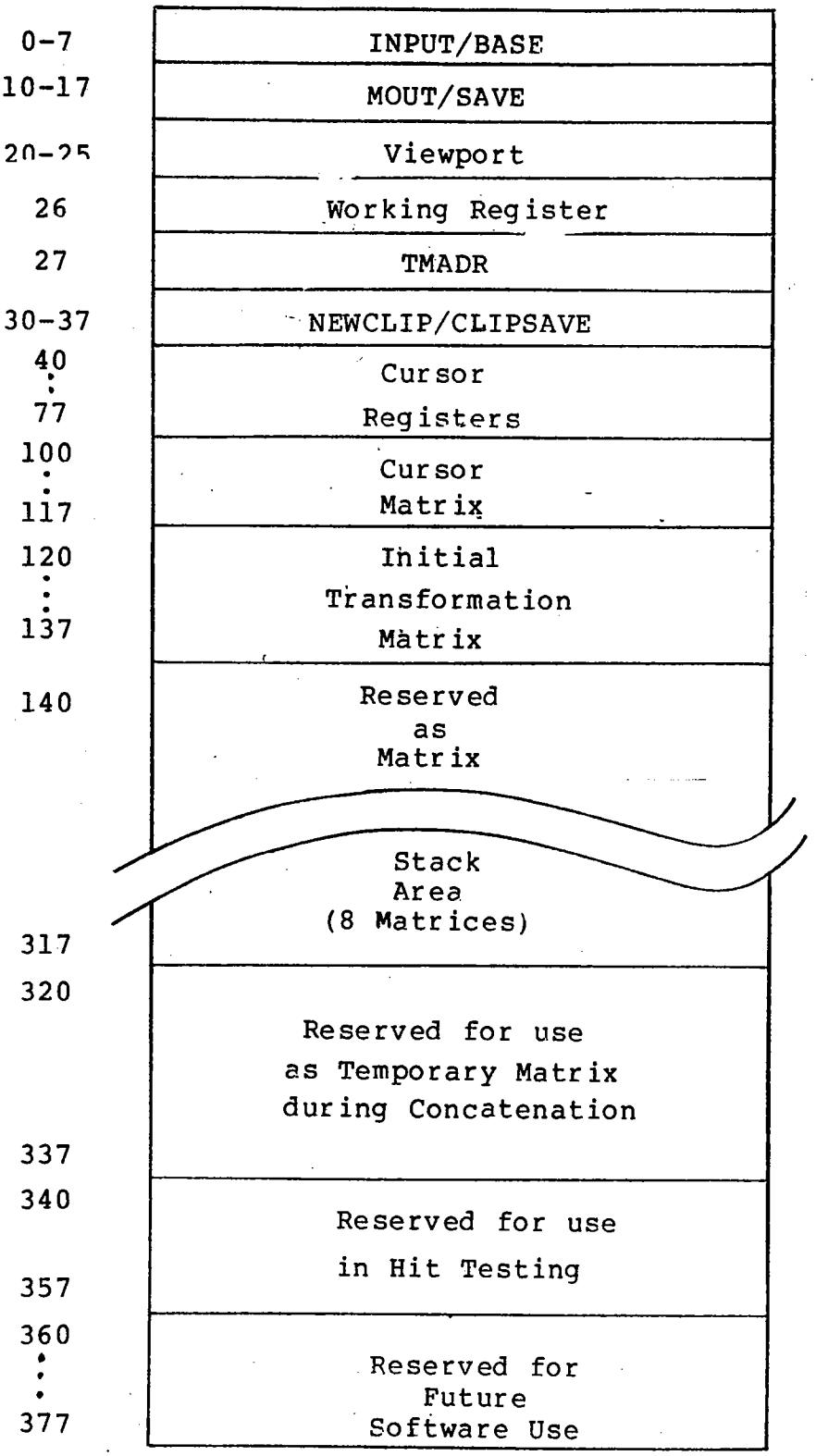


Figure 3-4

Software Partitioning of
the MAP Internal Registers

Registers 100-117 are reserved as a cursor matrix for use in cursor transformation. This area is initialized to an identity matrix. The TMADR Register (27) is loaded with 137, thereby selecting Registers 120-137 as the initial Transformation Matrix. Registers 120-137 are used to hold the Transformation Matrix which is initialized to an identity matrix. Registers 140-317 are reserved for use as Matrix Stack area, providing room for up to 8 matrices to be stacked. It should be noted that this area may be used in a random access manner by modifying the TMADR register to select one of the 9 matrices which may be contained in Registers 120-317. Registers 320-337 are reserved as working registers for a Temporary Matrix during matrix concatenation when the matrix stack is at level 8. Registers 340-357 are reserved for saving the current Transformation Matrix during the Hit Testing operation when the Matrix Stack is at level 8. Registers 360-377 are reserved for future software use to facilitate the utilization of PICTURE SYSTEM Memory to contain structured display lists.

3.1.1.2 Line Generator Initialization

Initialization of the Line Generator consists of outputting to the Line Generator Passive Input Port a Status Command (see Section 2.4.3) which sets the Line Generator to:

1. Non-blink mode
2. Solid line texture
3. Non-continuous texture mode
4. Non-relative line mode
5. Color and intensity selected for a monochrome Picture Display
6. All Picture Displays (Scopes 5-0) are selected for output

This status is then stored in a global variable and updated as modification to the Line Generator status is made.

3.1.1.3 Character Generator Initialization

As previously described in Section 2.4.4, the Character Generator is microcoded to provide interpretation of the 128 ASCII character set and to display standard or italicized characters in any of 8 programmable sizes. Initialization of the Character Generator consists of loading the RAM coefficient memory with the scaling coefficients to produce standard and italicized characters of sizes equal to the ROM coefficient memory but at a 90 degree counter-clockwise orientation as shown in Figure 3-5. The Font Parameter Stack is then loaded with a value which selects standard horizontal characters of mid-range size. The RAM Character Memory is not loaded and is available for loading and utilization by the user program.

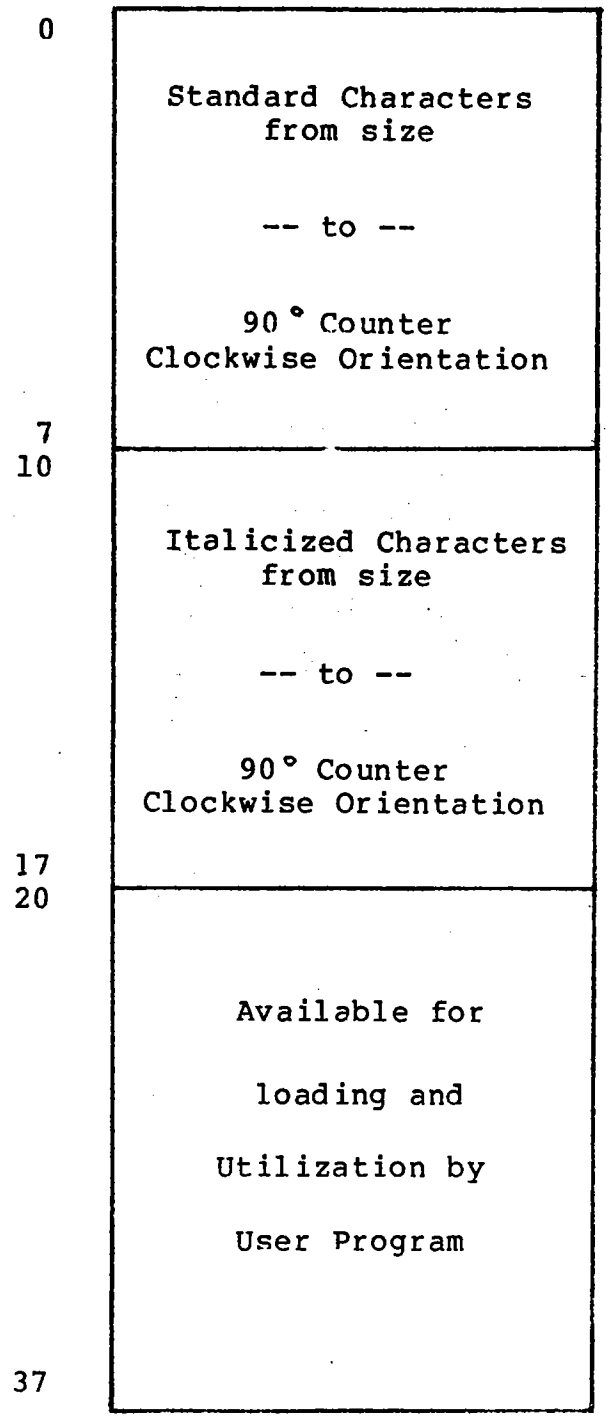


Figure 3-5

Character RAM Coefficient Memory Utilization

3.1.2 System Conventions

The following conventions have been adopted to simplify the development and debugging of software for PICTURE SYSTEM 2. These conventions should be followed unless just cause can be found for violating the convention. Such instances must be clearly documented.

1. Interrupt processing uses the DIO path only. Interrupt service routines are not allowed to perform DMA data transfers.

2. The DIOPSA register should be read only by interrupt service routines, unless all PICTURE SYSTEM 2 interrupt levels are masked.

3. Passive Output Ports (i.e., MPOP) must not be addressed using the DIO path unless all PICTURE SYSTEM 2 interrupt levels are masked. This is necessary since data from a Passive Output Port can be read only once; therefore, data could be lost if an interrupt occurred before all data was read.

4. The Picture Processor should not be used during interrupt processing. There is no guarantee that all of the user data will have been transferred to the MAP Input Controller when the interrupt occurs.

3.2 System Subroutines

This section describes system subroutines utilized to implement the Graphics Software Package provided with PICTURE SYSTEM 2. These subroutines are available to the programmer who wishes to interface to the system software directly.

The subroutines provided are:

P$AVE

R$TORE

P$WAIT

P$DMA

I$MATX

P$DIV

P$MUL

E$RROR

P$AVE

The P$AVE subroutine is called to save registers R0-R5 on the program stack.

Assembly Calling Sequence:

```
JSR     PC,P$AVE
```

R$TORE

The R$TORE subroutine is called to restore registers R0-R5 from the program stack.

Assembly Calling Sequence:

```
JSR     PC,R$TORE
```

P$WAIT

The P$WAIT subroutine is called to ensure that the Direct I/O interface has completed its previous operation. This routine performs a time-out on the operation and declares an error if Direct I/O operation has not completed at the end of the time-out.

Assembly Calling Sequence:

```
JSR     PC,P$WAIT
```

ERRORS:

1,3: Direct I/O error. This indicates the DIO interface was not ready. Check to ensure the PICTURE SYSTEM is powered-up properly.

P$DMA

The P$DMA subroutine is called to initiate a Direct Memory Access (DMA) transfer and check for the correct completion of the operation.

Assembly Calling Sequence:

R0 = Repeat Status Register (RSR) value
R1 = DMA word count value
R2 = DMA base address for transfer

```
JSR     PC,P$DMA
```

ERRORS:

1,2: DMA error. This indicates that an error occurred in the last Direct Memory Access operation.

I$MATX

The I$MATX subroutine is called to initialize a 16-word array in memory (P$MATX) to a 4x4 identity matrix.

Assembly Calling Sequence:

```
JSR     PC,I$MATX
```

ERRORS:

None

The following two-function subroutines are optimized for the particular PDP-11 hardware configuration:

P$DIV

The P$DIV function subroutine divides the signed dividend in R0 and R1 by the signed divisor in R2, leaving the quotient in R0 and the remainder in R1, with R2 undisturbed. The quotient bears the algebraic sign of the division, while the remainder retains the sign of the dividend.

Assembly Calling Sequence:

```
R0,R1 = Dividend
R2    = Divisor

JSR     PC,P$DIV
```

ERRORS:

v=1 (overflow condition code set) if the magnitude of the dividend is not less than half that of the divisor, or if the divisor is zero.

P$MUL

The P$MUL function subroutine multiplies the signed multiplicand in R0 by the signed multiplier in R2, leaving a signed product in R0 and R1, with R2 undisturbed.

Assembly Calling Sequence:

R0 = Multiplicand
R2 = Multiplier

```
JSR     PC,P$MUL
```

ERRORS:

None

E$RROR

The E$RROR subroutine is called by all PICTURE SYSTEM 2 subroutines that encounter an error condition during the course of execution. This subroutine, in turn, calls the user error subroutine specified in the call to PSINIT, or the default system error routine.

Assembly Calling Sequence:

```
JSR     PC,E$RROR
.BYTE   ICODE,IERR
```

where:

ICODE is the error code used to indicate the origin of the error detected.

IERR is the error type used to indicate the error condition encountered.

ERRORS:

None

Error detection by the Graphics Software Package is performed to ensure program integrity and to facilitate program debugging. A user may make four types of programming errors that will be detected by the Graphics Software Package:

1. The call of a graphics subroutine with an invalid number of parameters specified.
2. The call of a graphics subroutine with an invalid parameter value.
3. The attempt by the user to PUSH the matrix stack to a depth greater than that specified by the user in the call to PSINIT.
4. The attempt by the user to POP a transformation from the matrix stack which had not been previously PUSHed.

When an error is detected by a graphics subroutine, the system subroutine E$RROR is called with an argument that specifies the origin of the error detected and the error condition encountered. The system subroutine E$RROR then calls the user error subroutine, specified in the call to PSINIT. When called, the user error subroutine will be passed a parameter which specifies the origin and type of error detected. The error parameter is of the following form:

```
.BYTE       ICODE,IERR
```

where:

ICODE — is the error code used to indicate the origin of the error detected.

IERR — is the error type used to indicate the error condition encountered.

Return from the user error subroutine will result in the termination of the program. If, in the call to PSINIT, the user does not specify an error subroutine, the graphics error subroutine PSERRS will be called. PSERRS, when called, will output the following message to the console terminal:

ERROR "IERR" DETECTED IN GRAPHICS SUBROUTINE "ICODE".

and terminate the execution of the program.

NOTE: Unless the user's error subroutine is named PSERRS, the resultant core image will also include the graphics error subroutine PSERRS.

3.3 PICTURE SYSTEM 2 Graphics Software Implementation

The PICTURE SYSTEM 2/PDP-11 Graphics Software is written using MACRO-11 assembly language to support a PICTURE SYSTEM on any PDP-11 family computer. The software support package consists basically of a set of FORTRAN-callable subroutines which may be conditionally assembled to execute under any of the following DEC operating environments:

DOS/BATCH Operating System
RT-11 Real-Time Operating System
RSX-11M Real-Time System

The following sections describe the implementation characteristics under each of these operating environments and the macros used to facilitate implementation.

3.3.1 PICTURE SYSTEM 2 Macros

Each release of the Graphics Software contains the source files for each of the subroutines provided and a file of macros with which each subroutine must be assembled if the assembly is to perform properly. This file of macros is named PSMACS.MAC and contains the following macros:

PSMACS - This macro is invoked to perform four basic functions:

1. All conditional assembly variables are defined. See Section 3.3.2 for a definition of all conditional assembly variables.
2. All other macros which are in general referenced in every PICTURE SYSTEM 2 graphics subroutine are called.
3. A macro is invoked to check for conflicting definitions of conditional assembly variables (CKCDX).
4. All general-purpose registers are defined (R0-PC).

CKCDX - This macro is invoked to check as to whether conflicting conditional assembly variables have been defined.

PSREGS - This macro is invoked to define all PICTURE SYSTEM 2 device registers.

GLOBLS - This macro is invoked to define all PICTURE SYSTEM 2 service routines and variables as globals.

.CALL - This macro is invoked to produce the FORTRAN-calling sequence "JSR PC,XXX". This has been included to facilitate the use of alternate calling sequence conventions if required by the user.

.RETURN - This macro is invoked to produce the FORTRAN return from subroutine sequence "RTS PC". Only FORTRAN-callable subroutines return via this macro.

.SAVE - This macro is invoked to cause the registers R0-R5 to be saved on the stack if the user requires that all registers be saved upon entry to a graphics subroutine.

.RSTORE - This macro is invoked to cause the contents of registers R0-R5 to be restored from the stack if the user requires that all registers be saved upon entry to a graphics subroutine.

WFBFSW - This macro is invoked to test whether data may be sent to the refresh buffer of the PICTURE SYSTEM Memory. For RT-11 Foreground programs and RSX tasks, this causes control to be passed to the executive for background processing if data may not be output.

.DEVO - This macro is invoked whenever a PICTURE SYSTEM 2 UNIBUS device register is to be referenced for an output operation. The code produced is dependent upon the operating system under which the program is to run (as defined by conditional assembly variables).

.DEVI - This macro is invoked whenever a PICTURE SYSTEM 2 UNIBUS device register is to be referenced for an input operation. The code produced is dependent upon the operating system under which the program is to run (as defined by conditional assembly variables).

3.3.2 Conditional Assemblies

The source files supplied with each release of the PICTURE SYSTEM 2 Graphics Software are identical, regardless of what operating system they are to be used on. These source files are conditionally assembled for the particular operating system that is to be used. This conditional assembly process is controlled by the definition (or non-definition) of several variables within the PSMACS macro. Those variables are as follows:

RT11, RSX -
If either of these variables is defined, (e.g., RT11=0), the source files will be assembled for that operating system. If neither of these variables is defined, the sources are assembled for the DOS/BATCH operating system.

SVREGS - If this variable is defined (e.g., SVREGS=0), the Graphics Subroutines will be assembled to save and restore all registers (R0-R5) upon entry and exit. The software is distributed to not save registers across subroutine calls.

HRDEAU - If this variable is defined (e.g., HRDEAU=0), PSINIT is assembled to utilize a MUL and DIV instruction. (This requires a PDP-11/35 or PDP-11/40 with a KE11-E or a PDP-11/34, PDP-11/45, PDP-11/55 or PDP-11/70).

UNIEAU - If this variable is defined (e.g., UNIEAU=0), PSINIT is assembled to utilize a KE11-A-extended arithmetic unit of a PDP-11/05 or PDP-11/10 for multiplication and division.

If neither HRDEAU or UNIEAU is defined, PSINIT is assembled to provide a software routine for multiplication and division.

RSXF - If this variable is defined (e.g., RSXF=0) in addition to the RSX conditional variable, the source files will be assembled to interface directly to the five UNIBUS device registers providing a reduction in system overhead.

3.3.3 DOS/BATCH and RT-11 Implementation

The Graphics Software for the DOS/BATCH and RT-11 operating systems is similar in the manner in which it is implemented. All interaction with the PICTURE SYSTEM 2 hardware is handled directly by the Graphics Software which directly manipulates the UNIBUS device registers of the PICTURE SYSTEM Picture Controller Interface. Under both operating systems, the PSINIT subroutine contains an interrupt handler which manages the refresh process of PICTURE SYSTEM 2 and all graphics functions are performed by subroutines of the Graphics Software which are linked with the user's program. It should be noted that the DOS/BATCH or RT-11 overlay facility may be utilized, with the exception of PSINIT, to include any graphics subroutine in an overlay segment. Additionally, if the TABLET or CURSOR subroutines are used in "automatic mode", those subroutines should also be included in the root segment rather than in an overlay segment as these routines are entered for interrupt servicing.

3.3.3.1 DOS/BATCH Specifics

To assemble the Graphics Software to be used under the DOS/BATCH operating system, the conditional variables RT11 and RSX must not be defined in the macro PSMACS.

When using the DOS/BATCH operating system, the general memory layout is that shown in Figure 3.3-1. As this figure illustrates, a user program is linked to reside in "high core" and expands downward toward the DOS/BATCH monitor.

When a program is aborted by the user under DOS/BATCH by means of a C and KILL command, a system initialization signal is produced by the DOS/BATCH monitor by the execution of a RESET instruction. This causes the refreshing of the display to stop by resetting the PICTURE SYSTEM 2 interrupt enable bits of the IOST device register. Due to this instruction, there are no special requirements of the Graphics Software.

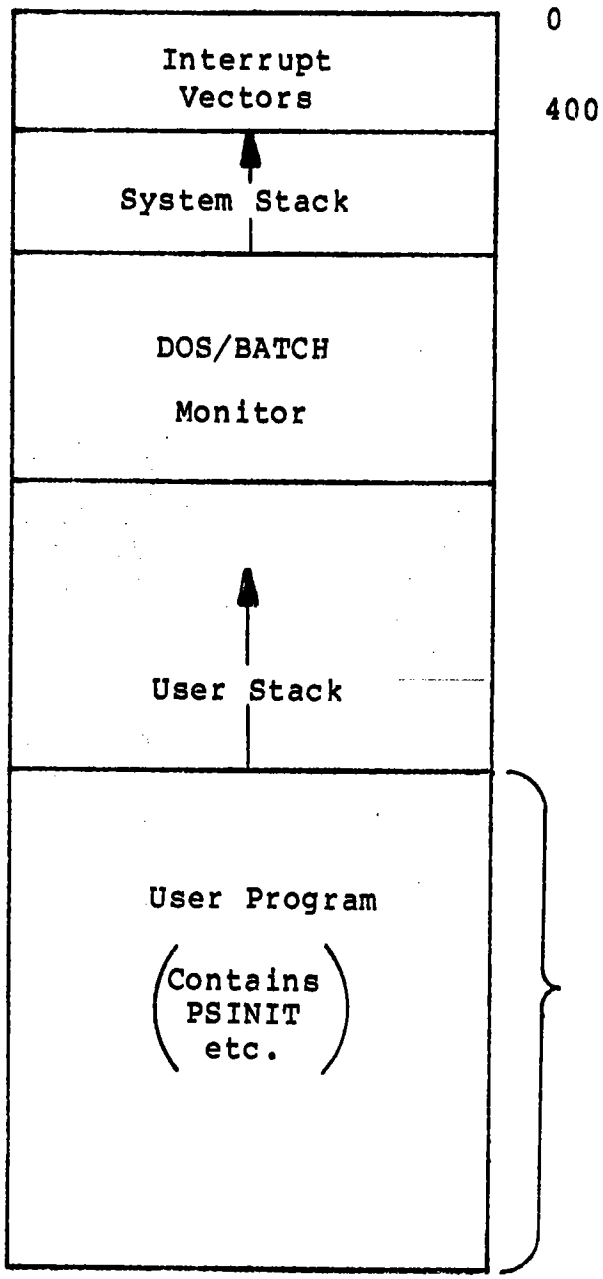


Figure 3.3-1

Typical DOS/BATCH Memory Configuration

3.3.3.2 RT-11 Specifics

To assemble all Graphics Software to be used under the RT-11 operating system, the symbol RT11 should be defined within the macro PSMACS as RT11=0. The conditional variables RSX must not be defined since a conditional assembly error will be produced, indicating invalid conditional flags were defined.

When using the RT-11 operating system (RT-11SJ or RT-11FB), the general memory layout is that shown in Figure 3.3-2. As this figure illustrates, the RT-11 resident monitor always resides in "high core" with background jobs (or RT-11SJ programs) residing in low core and is linked upwards toward the monitor. When a program is aborted by the user under the RT-11SJ monitor by typing two Control C's (^C^C), a system initialization signal is produced by the RT-11SJ monitor by the execution of a RESET instruction. This causes the refreshing of the display to stop by resetting the interrupt enable bits of the IOST device register. However, when a program is aborted by the user under the RT-11FB monitor by typing two Control C's (^C^C), no RESET instruction is executed to reset the PICTURE SYSTEM. Because of this, the Graphics Software (specifically, PSINIT when called) must provide the RT-11FB monitor with a list of addresses to be loaded with specific values when a program is terminated by a Control C (^C) or .EXIT. This is accomplished via the RT-11 .DEVICE macro. To facilitate the execution of PICTURE SYSTEM 2 graphics programs as Foreground jobs, the Graphics Software has been implemented to suspend execution of the Foreground job whenever the Graphics Software must wait for a new frame to be initiated before output may proceed to the refresh buffer in PICTURE SYSTEM Memory. The PICTURE SYSTEM 2 refresh interrupt handler then requests that the Foreground job be resumed when output to the refresh buffer may proceed. If the user program is executing as a Background job, the program then loops rather than suspending execution. In this way, any user program may be successfully executed and terminated under either the RT-11SJ or RT-11FB monitor without relinking or modification.

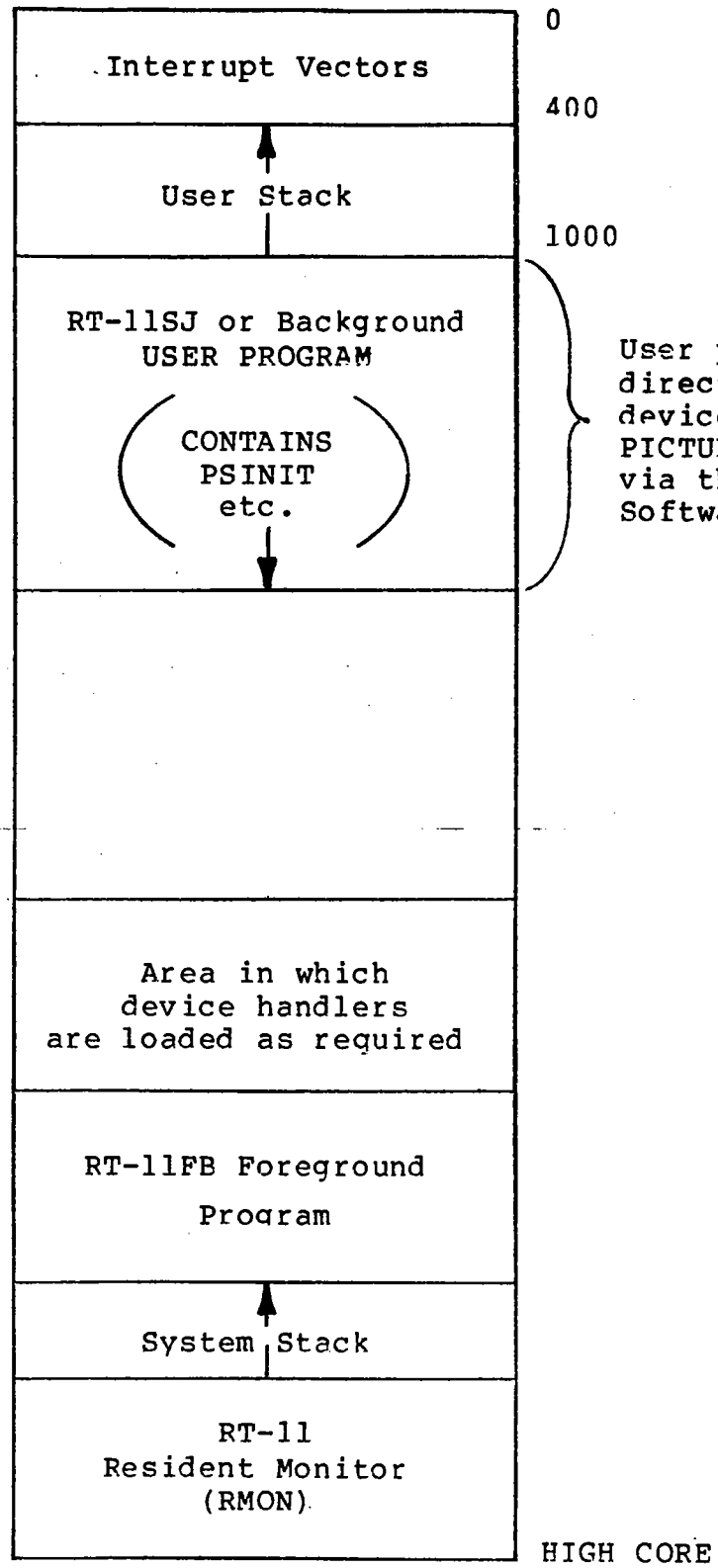


Figure 3.3-2

Typical RT-11 Memory Configuration

3.3.4 RSX-11M Implementation

The Graphics Software for the RSX-11M operating system differs from that produced for the DOS/BATCH and RT-11 operating systems in that all interaction with the PICTURE SYSTEM 2 hardware is not handled directly by the Graphics Software. Rather, the hardware is handled by a PICTURE SYSTEM Device Driver which must be installed as a part of the resident executive. This Device Driver (PSDRV) contains an interrupt handler which manages the refresh process of PICTURE SYSTEM 2 rather than the PSINIT subroutine. All graphics functions which are performed directly by the Graphics Software under other operating systems are now set up by the Graphics Subroutines and the Device Driver is "called" to perform the function by directly manipulating the device registers in PICTURE SYSTEM 2. In this manner, the user's graphics program does not require direct access to the peripheral I/O page and therefore may be a non-priviledged task which has a full 32K words of address space available for the maximum length of the task. The general memory layout for an RSX-11M operating system generated to support a PICTURE SYSTEM 2 is that shown in Figure 3.3-3. As this figure illustrates, a user program is built by the Task Builder to reside in a partition of memory and communicates to the PICTURE SYSTEM via the Graphics Software. Upon execution of a user graphics program, the PSINIT subroutine is called to initialize the PICTURE SYSTEM 2 hardware and software. Upon entry, PSINIT attempts to assign logical unit 4 to PS0:. Once a successful attach operation has been completed, the user task is free to perform graphics operations using the graphics subroutines.

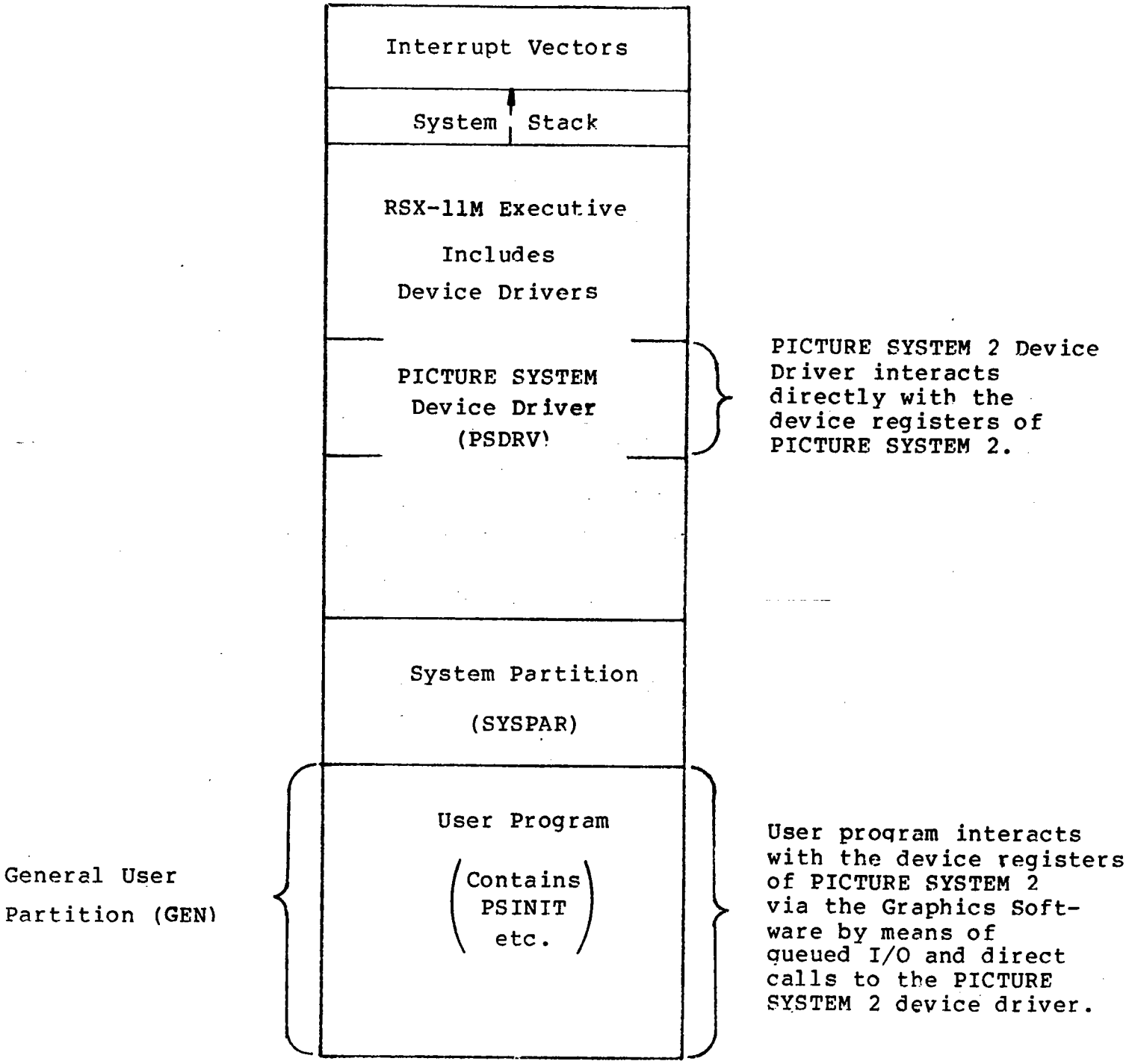


Figure 3.3-3

Typical RSX-11M Memory Configuration

All graphics subroutines communicate with the PICTURE SYSTEM 2 Device Driver by macros which expand into illegal instructions. These instructions provide a way to directly call "PSDRV", thus avoiding the overhead inherent in the queued I/O (QIO$) method. This direct call is set up by the driver during the attach function and restored by a detach function. When an illegal instruction trap occurs (trap through vector 10), the violating instruction is checked first by PSDRV. If it is not one of the PICTURE SYSTEM "codes", it is passed to the usual illegal instruction handler of the RSX-11M executive. If it is one of the PICTURE SYSTEM "codes" (instructions 7000-7004), the function specified by the code is performed. User graphics output proceeds until a segment is complete. At that time, a queued I/O (QIO$) request to PSDRV is initiated to cause the data to be appended to the refresh buffer. To allow other programs to execute concurrently with graphics programs, when output to the refresh buffer must wait for a previous I/O request to complete, the graphics task is suspended waiting for Event Flag 49 to be set. When Event Flag 49 becomes set, indicating that output to the refresh buffer may proceed, execution of the graphics task is resumed.

To avoid the overhead involved in fielding and interpreting illegal instruction traps, the graphics software may also be assembled to interface directly with the five PICTURE SYSTEM 2 UNIBUS device registers. This is done by installing a five-word Global Common Block (PSDEV) with the DEV attribute and Task Building the graphics program to the Global Common Block. This allows direct access to the PSDATA, DIOPSA, DMAWC and IOST registers. The DMABA register is not referenced directly to facilitate the mapping of the 16-bit virtual addresses to the UNIBUS physical address for the DMA transfer. This method, however, limits the maximum size of any graphics task to 28K.

To assemble all Graphics Software to be used under the RSX-11M operating system, the symbol RSX should be defined within the macro PSMACS as RSX=0. If the Graphics Software is to interface directly to the five PICTURE SYSTEM 2 device registers, the symbol RSXF should also be defined within the macro PSMACS as RSXF=0. As distributed, the Graphics Libraries do not interface directly to the PICTURE SYSTEM 2 device registers, but performs all I/O operations by the PICTURE SYSTEM Device Driver, PSDRV. The conditional variable RT11 must not be defined or a conditional assembly error will be produced, indicating that invalid conditional flags were defined.

CHAPTER FOUR

4. THE 4x4 MATRIX TRANSFORMATIONS USED BY PICTURE SYSTEM 2

All transformations used by PICTURE SYSTEM 2 consist of simple or compound linear transformations and are implemented using a 4x4 matrix. This chapter details the matrices created by the Graphics Software Package provided with PICTURE SYSTEM 2. Each subroutine calling sequence is shown, and the manner in which the parameters passed to the subroutines are used to create the transformations is indicated. In most of the subroutines (all except where the transformation is purely rotational), the inclusion of the homogeneous coordinate (W) is optional and, if not specified, is assumed to be 32767 (the largest expressable integer in a 16-bit word length computer).

WINDOW

CALL WINDOW(WL,WR,WB,WT[,WH,WY[,WE[,W]]])

Two Matrices are produced: $\begin{bmatrix} \text{WINDOW} \end{bmatrix} = \begin{bmatrix} \text{W2} \end{bmatrix} \begin{bmatrix} \text{W1} \end{bmatrix}$

$$W1 = \begin{bmatrix} 1/A & 0 & 0 & 0 \\ 0 & 1/C & 0 & 0 \\ 0 & 0 & 1/E+1/F & 1/F \\ 0 & 0 & 0 & 1/W \end{bmatrix}$$

$$W2 = \begin{bmatrix} W & 0 & 0 & 0 \\ 0 & W & 0 & 0 \\ 0 & 0 & W & 0 \\ -B & -D & -WH & W \end{bmatrix}$$

Where:

A = (WR-WL)/2
B = (WR+WL)/2
C = (WT-WB)/2
D = (WT+WB)/2
E = WY-WH
F = WH-WE
WH = 0 default
WY = W default
WE = minus infinity default

ROT

CALL ROT(ANGLE,AXIS)

One of three Matrices is created, dependent upon the axis of rotation:

X axis =

$$\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & A & B & 0 \\ 0 & -B & A & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

Y axis =

$$\begin{bmatrix} A & 0 & -B & 0 \\ 0 & I & 0 & 0 \\ B & 0 & A & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

Z axis =

$$\begin{bmatrix} A & B & 0 & 0 \\ -B & A & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

Where:

```
I = 2**14
A = I*cos(ANGLE)
B = I*sin(ANGLE)
```

TRAN

CALL TRAN(X,Y,Z[,W])

A single Matrix is produced:

$$\begin{bmatrix} W & 0 & 0 & 0 \\ 0 & W & 0 & 0 \\ 0 & 0 & W & 0 \\ X & Y & Z & W \end{bmatrix}$$

SCALE

CALL SCALE(X,Y,Z[,W])

A single Matrix is produced:

$$\begin{bmatrix} X & 0 & 0 & 0 \\ 0 & Y & 0 & 0 \\ 0 & 0 & Z & 0 \\ 0 & 0 & 0 & W \end{bmatrix}$$

MASTER

CALL MASTER(ML,MR,MB,MT[,MH,MY[,W]])

Two Matrices are produced: $\begin{bmatrix} \text{MASTER} \end{bmatrix} = \begin{bmatrix} \text{M2} \end{bmatrix} \quad \begin{bmatrix} \text{M1} \end{bmatrix}$

$$\begin{bmatrix} 1/A & 0 & 0 & 0 \\ 0 & 1/C & 0 & 0 \\ 0 & 0 & 1/E & 0 \\ 0 & 0 & 0 & 1/W \end{bmatrix}$$

$$\begin{bmatrix} W & 0 & 0 & 0 \\ 0 & W & 0 & 0 \\ 0 & 0 & W & 0 \\ -B & -D & -MH & W \end{bmatrix}$$

Where:

A = (MR-ML)/2
B = (MR+ML)/2
C = (MT-MB)/2
D = (MT+MB)/2
E = MY-MH
MH = 0 default
MY = W default

INST

CALL INST(IL,IR,IB,IT[IH,IY[,W]])

A single Matrix is produced:

$$\begin{bmatrix} A & 0 & 0 & 0 \\ 0 & C & 0 & 0 \\ 0 & 0 & E & 0 \\ B & D & IH & W \end{bmatrix}$$

Where:

A = (IR-IL)/2
B = (IR+IL)/2
C = (IT-IB)/2
D = (IT+IB)/2
E = IY-IH
IH = 0 default
IY = W default

HITWIN

CALL HITWIN(X,Y,S[,W])

A single Matrix is produced:

$$\begin{bmatrix} W & 0 & 0 & -X \\ 0 & W & 0 & -Y \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & S \end{bmatrix}$$

NOTE: This Matrix is actually the transpose of the hit window matrix. This matrix is concatenated with the current Transformation Matrix in a manner such that both are treated as transposed matrices. This performs a matrix pre-concatenation (rather than post-concatenation as is typically the case). See Section 2.3.2.2, "MATCON" for specific details.

APPENDIX A

PROGRAMMING PICTURE SYSTEM 2

A.1 Introduction

Chapter 2 of this manual described in detail the PICTURE SYSTEM 2/PDP-11 Interface and PICTURE SYSTEM 2 registers used in programming the system. The purpose of this appendix is to show how these registers may be used in a program which interfaces with the hardware at the assembly language level.

A.2 Program Description

To illustrate how to program PICTURE SYSTEM 2, a simple program will be detailed which displays a cube and allows it to be translated in the x, y and z directions according to console switch settings. During this program, the characters "CUBE" will be displayed blinking continuously.

In addition to the program details, the following points should be emphasized:

1. During initialization, the various devices must be set to the configuration as it is intended to be used. For this program, the system is configured in the following manner:

   a. DMA (Active Output) => MAP Input Controller (Passive Input).

   b. MAP Output Formatter (Active Output) => PS Memory (Passive Input).

   c. PS Memory (Passive Output) => Refresh Controller (Active).

   d. Refresh Controller (Active) => Line Generator (Passive Input).

2. The number of data words transferred must correspond for the RSR Command that is being executed by the Picture Processor in order to ensure the operation continues to completion. Exactly what the correspondence is

depends upon the RSR command specified. For the 2DDRAW, 3DDRAW and 4DDRAW RSR commands, the CNT field of the RSR will contain the two's complement (sign implied) of the number of executions to be performed and the DMAWC will contain the two's complement of the number of PDP-11 words to be transferred. The following illustrates this relationship:

```
2DDRAW:   CNT <= -(number of 16-bit word pairs)
          DMAWC <= 2*CNT

3DDRAW:   CNT <= -(number of 16-bit word triples)
          DMAWC <= 3*CNT

4DDRAW:   CNT <= -(number of 16-bit 4-word sets)
          DMAWC <= 4*CNT
```

3. Before a 2DDRAW or 3DDRAW RSR command is performed (except for PASS commands, FSM2=5-7), the BASE register should be loaded with the constant Z and W coordinates for a 2DDRAW or the constant W coordinate for a 3DDRAW. The BASE register is loaded by a 4DDRAW Set Base (FSM2=0) RSR command. This is done since the BASE register supplies the Z and W coordinates for 2DDRAW commands and the W coordinate for the 3DDRAW command.

4. It should be noted that the Status Command (see Section 2.4.3) is deposited directly into the refresh buffer. Once a Status Command is encountered by the Picture Generator, the status specified will remain in effect (through subsequent frames when no Status Command is encountered) until an overriding Status Command is encountered.

The structure of this sample program is shown in Figure A-1. The following section contains the MACRO-11 assembly language listing of the program. Careful study of the listing should clarify many of the topics covered within this manual.

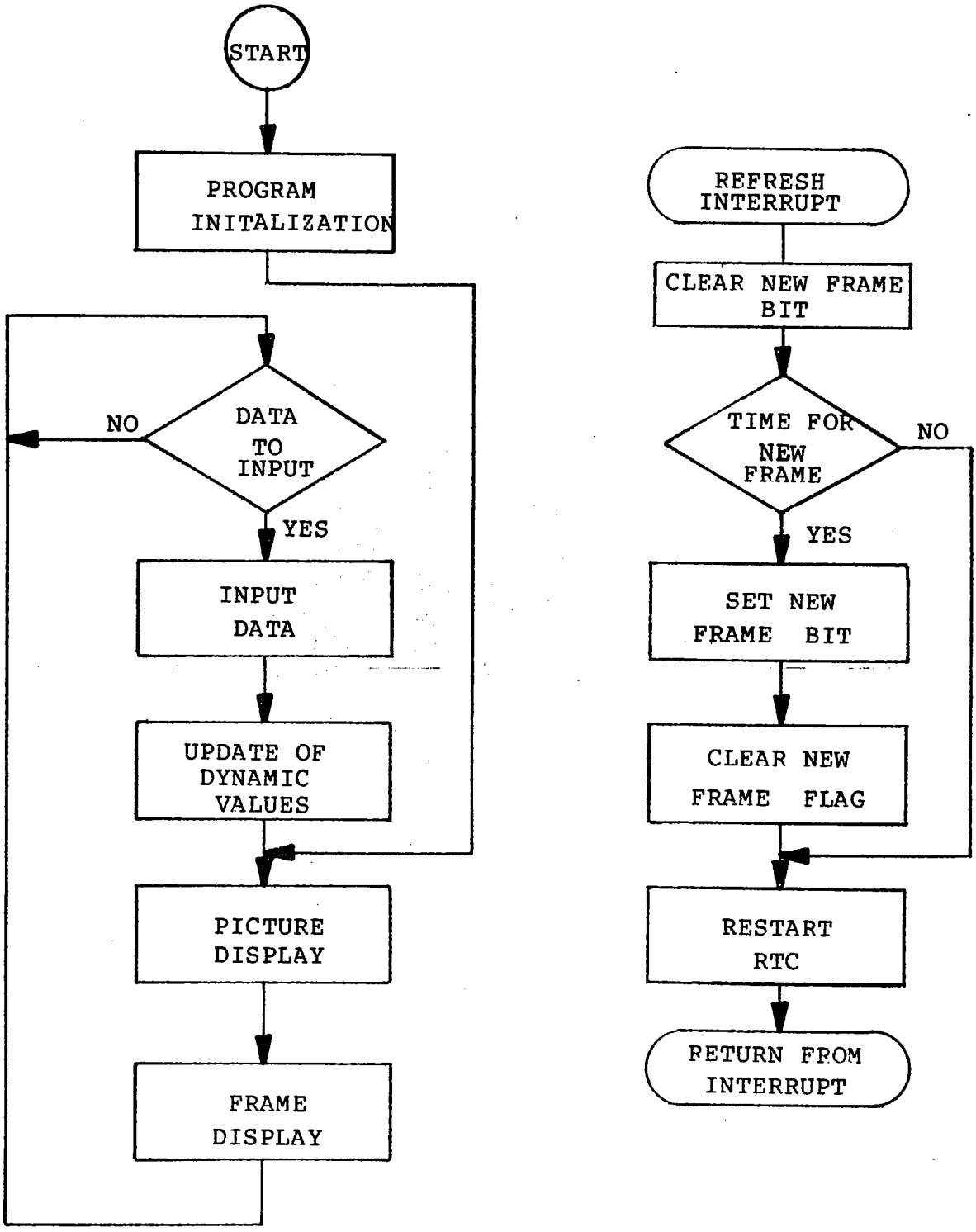


Figure A-1

Example Program Structure

A.3 MACRO-11 Program Example

```
                        .SBTTL  MACROS

                        ; THESE MACROS ARE IMPLEMENTED TO ALLOW THE MANIPULATION OF
                        ; PICTURE SYSTEM 2 MEMORY AND SCB LOCATIONS AS IF THEY WERE
                        ; DIRECTLY ADDRESSABLE FROM THE PDP-11.

                        .MACRO  RDPS    SRC,DST         ;READ PS2 "SRC" TO PDP-11 "DST"

                                TST     IOST
                                BPL     .-4
                                MOV     SRC,DIOPSA
                                TST     IOST
                                BPL     .-4
                                MOV     PSDATA,DST

                        .ENDM   RDPS


                        .MACRO  WTPS    SRC,DST         ;WRITE PDP-11 "SRC" TO PS2 "DST"

                                TST     IOST
                                BPL     .-4
                                MOV     DST,DIOPSA
                                TST     IOST
                                BPL     .-4
                                MOV     SRC,PSDATA

                        .ENDM


                        .MACRO  RWPS    INST,SRC,DST    ;PERFORM PDP-11 "INST"
                                                        ;FROM PDP-11 "SRC" TO PS2 "DST"
                                                        ;OR ON PS2 "SRC" IF ONE OPERAND

                                TST     IOST
                                BPL     .-4

                         .IF    B       DST
                                MOV     SRC,DIOPSA
                                TST     IOST
                                BPL     .-4
                                INST    PSDATA
                         .IFF
                                MOV     DST,DIOPSA
                                TST     IOST
                                BPL     .-4
                                INST    SRC,PSDATA
                         .ENDC

                        .ENDM   RWPS
```

TABLE OF CONTENTS

```
.TITLE  APPENDIX A:  SAMPLE PROGRAM

; EVANS & SUTHERLAND COMPUTER CORPORATION    COPYRIGHT (C) 1976 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; THIS SAMPLE PROGRAM DISPLAYS A CUBE IN PERSPECTIVE WITHIN A
; VIEWPORT WHICH IS SPECIFIED AS THE ENTIRE SCREEN USING PDP-11
; CODE WHICH INTERFACES DIRECTLY WITH PICTURE SYSTEM 2.  THE
; CUBE MAY BE TRANSLATED IN X, Y AND Z DIRECTIONS DEPENDENT UPON
; THE CONSOLE DATA SWITCH SETTINGS, ALL THE WHILE DISPLAYING THE
; CHARACTERS "THIS IS A CUBE" WHICH BLINK CONTINUALLY.
;
; CONSOLE SWITCH SETTINGS:
;
;       SWITCH 15:  PROGRAM RESTART
;       SWITCH 14:  TRANSLATE -X
;       SWITCH 13:  TRANSLATE -Y
;       SWITCH 12:  TRANSLATE -Z
;       SWITCH 11:  TRANSLATE +X
;       SWITCH 10:  TRANSLATE +Y
;       SWITCH  9:  TRANSLATE +Z
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; M. MANTLE  11/22/76                                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
                        .SBTTL  MACROS

                        ; THESE MACROS ARE IMPLEMENTED TO ALLOW THE MANIPULATION OF
                        ; PICTURE SYSTEM 2 MEMORY AND SCB LOCATIONS AS IF THEY WERE
                        ; DIRECTLY ADDRESSABLE FROM THE PDP-11.

                        .MACRO  RDPS    SRC,DST         ;READ PS2 "SRC" TO PDP-11 "DST"

                                TST     IOST
                                BPL     .-4
                                MOV     SRC,DIOPSA
                                TST     IOST
                                BPL     .-4
                                MOV     PSDATA,DST

                        .ENDM   RDPS


                        .MACRO  WTPS    SRC,DST         ;WRITE PDP-11 "SRC" TO PS2 "DST

                                TST     IOST
                                BPL     .-4
                                MOV     DST,DIOPSA
                                TST     IOST
                                BPL     .-4
                                MOV     SRC,PSDATA

                        .ENDM


                        .MACRO  RWPS    INST,SRC,DST    ;PERFORM PDP-11 "INST"
                                                        ;FROM PDP-11 "SRC" TO PS2 "DST"
                                                        ;OR ON PS2 "SRC" IF ONE OPERAND

                                TST     IOST
                                BPL     .-4

                         .IF    B       DST
                                MOV     SRC,DIOPSA
                                TST     IOST
                                BPL     .-4
                                INST    PSDATA
                         .IFF
                                MOV     DST,DIOPSA
                                TST     IOST
                                BPL     .-4
                                INST    SRC,PSDATA
                         .ENDC

                        .ENDM   RWPS
```

```
                                .SBTTL  DEFINITION OF REGISTERS AND COMMANDS

                        R0      =%0
                        R1      =%1
                        R2      =%2
                        R3      =%3
                        R4      =%4
                        R5      =%5
                        SP      =%6
                        PC      =%7

                        LOWVEC  =000340         ;LOWEST PS2 INTERRUPT VECTOR

                        PSDATA  =167660         ;PS2 PSDATA UNIBUS REGISTER
                        DIOPSA  =PSDATA+2       ;PS2 DIOPSA UNIBUS REGISTER
                        DMAWC   =PSDATA+4       ;PS2 DMAWC UNIBUS REGISTER
                        DMABA   =PSDATA+6       ;PS2 DMABA UNIBUS REGISTER
                        IOST    =PSDATA+10      ;PS2 IOST UNIBUS REGISTER
                        SWITCH  =177570         ;CONSOLE SWITCH REGISTER

                        RFASA   =177735         ;PS2 RFASA SCB REGISTER
                        RFAIL   =RFASA+1        ;PS2 RFAIL SCB REGISTER
                        RFSR    =RFASA+2        ;PS2 RFSR SCB REGISTER

                        RTCCNT  =177724         ;PS2 RTCCNT SCB REGISTER
                        RTCSR   =RTCCNT+1       ;PS2 RTCSR SCB REGISTER

                        DMAPSA  =177727         ;PS2 DMAPSA SCB REGISTER

                        MAOL    =177750         ;PS2 MAOL SCB REGISTER
                        MAOA    =MAOL+1         ;PS2 MAOA SCB REGISTER
                        MAIA    =MAOL+2         ;PS2 MAIA SCB REGISTER
                        MSR     =MAOL+3         ;PS2 MSR SCB REGISTER
                        MMSR    =MAOL+4         ;PS2 MMSR SCB REGISTER

                        RTCREQ  =177760         ;PS2 RTCREQ SCB REGISTER
                        RTCIE   =RTCREQ+1       ;PS2 RTCIE SCB REGISTER

                        LGPIP   =177775         ;PS2 LGPIP SCB REGISTER
                        MPIP    =177777         ;PS2 MPIP SCB REGISTER

                        LOAD    =010000         ;LOAD RSR COMMAND
                        PUSH    =022360         ;PUSH RSR COMMAND
                        POP     =024360         ;POP RSR COMMAND
                        MATCON  =026000         ;MATCON RSR COMMAND
                        DRAW2D  =040000         ;BASIC 2DDRAW RSR COMMAND
                        DRAW3D  =100000         ;BASIC 3DDRAW RSR COMMAND
                        SETBAS  =140377         ;4DDRAW (SETBASE) RSR COMMAND
```

```
                                                .SBTTL  DEFINITION OF CONSTANTS AND DATA

000000                                          BEGIN:
000000  000000                                  MAPREG: .WORD   0                       ;BEGIN LOADING MAP REGISTER 0
000002  000000  000000  000000                          .WORD   0,0,0,77777             ;0-3: INPUT/BASE
000010  077777
000012  000000  000000  000000                          .WORD   0,0,0,77777             ;4-7: BASE/INPUT
000020  077777
000022  000000  000000  000000                          .WORD   0,0,0,0                 ;10-13: MOUT/SAVE
000030  000000
000032  000000  000000  000000                          .WORD   0,0,0,0                 ;14-17: SAVE/MOUT
000040  000000
000042  003777                                         .WORD   2047.                   ;20: VIEWXH
000044  000000                                         .WORD   0                       ;21: VIEWXC
000046  003777                                         .WORD   2047.                   ;22: VIEWYH
000050  000000                                         .WORD   0                       ;23: VIEWYC
000052  177701                                         .WORD   -63.                    ;24: VIEWZS (MAX DEPTH CUEING)
000054  000077                                         .WORD   63.                     ;25: VIEWZF
000056  000000                                         .WORD   0                       ;26: NOT USED
000060  000057                                         .WORD   57                      ;27: IMADR
000062  000000  000000  000000                          .WORD   0,0,0,0                 ;30-33: NEWCLIP/CLIPSAVE
000070  000000
000072  000000  000000  000000                          .WORD   0,0,0,0                 ;34-37: CLIPSAVE/NEWCLIP
000100  000000
                                                                                        ;40-57: INDENTITY MATRIX
000102  040000  000000  000000                          .WORD   40000,0,0,0             ;(DIAGONAL=40000)
000110  000000
000112  000000  040000  000000                          .WORD   0,40000,0,0
000120  000000
000122  000000  000000  040000                          .WORD   0,0,40000,0
000130  000000
000132  000000  000000  000000                          .WORD   0,0,0,40000
000140  040000

                                                ; INITTIAL PICTURE GENERATOR STATUS

000142  000200  176000                          PGINIT: .WORD   200,176000              ;INITIAL LG STATUS (ALL SCOPES)
000146     003     000     005                          .BYTE   3,0,5,0                 ;LOAD CG FONT PARAMETER STACK
000151     000

                                                ; BLINK STATUS

000152  020200  176000                          BLNKON: .WORD   20200,176000            ;SELECT BLINK AND ALL SCOPES
000156  000200  176000                          BLNKOF: .WORD   200,176000              ;SELECT ONLY ALL SCOPES

                                                ; PERSPECTIVE MATRIX

000162  040000  000000  000000                  PERMAT: .WORD   40000,0,0,0             ;40000,00000,00000,00000
000170  000000
000172  000000  040000  000000                          .WORD   0,40000,0,0             ;00000,40000,00000,00000
000200  000000
000202  000000  000000  040000                          .WORD   0,0,40000,20000         ;00000,00000,40000,20000
000210  020000
000212  000000  000000  000000                          .WORD   0,0,0,40000             ;00000,00000,00000,40000
000220  040000
```

```
                                        ; TRANSLATION MATRIX
                                        TRNMAT: .WORD   40000,0,0,0      ;40000,00000,00000,00000
                                                .WORD   0,40000,0,0      ;00000,40000,00000,00000
                                                .WORD   0,0,40000,0      ;00000,00000,40000,00000
                                        TX:     .WORD   0                ;   TX,   TY,   TZ,40000
                                        TY:     .WORD   0
                                        TZ:     .WORD   0,40000

                                        ; BASE REGISTER VALUES

                                        ZWBASE: .WORD   0,0,0,77777      ;Z=0, W=77777 (W=32767.)

                                        ; VARIABLES AND DATA

                                        SETPT:  .WORD   -10000.,-10000.
                                        TEXT:   .ASCII  'THIS IS A CUBE'

                                                .BYTE   0,0              ;(CHARACTERS MUST BE OUTPUT
                                                                         ; IN MULTIPLES OF 4)

                                        CDAT1:  .WORD   2048.,2048.,-2048., -2048.,2048.,-2048.
                                                .WORD   -2048.,-2048.,-2048., 2048.,-2048.,-2048.
                                                .WORD   2048.,2048.,-2048., 2048.,2048.,2048.
                                                .WORD   -2048.,2048.,2048., -2048.,-2048.,2048.
                                                .WORD   2048.,-2048.,2048., 2048.,2048.,2048.
                                        CDAT2:  .WORD   -2048.,2048.,-2048., -2048.,2048.,2048.
                                                .WORD   -2048.,-2048.,-2048., -2048.,-2048.,2048.
                                                .WORD   2048.,-2048.,-2048., 2048.,-2048.,2048.
                                        NUFRAM: .WORD   0
                                        INCVAL: .WORD   200.
                                        BUFPTR: .WORD   -2                       ;RBTB INDEX
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                                .WORD   0                        ;REFRESH BUFFER POINTER TABLE
                                        RBTB:   .WORD   0,0,0                    ;DO NO SEPARATE
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
                                        .SBTTL  PROGRAM INITIALIZATION

000474  012706  000000'      START:  MOV     #BEGIN,SP       ;SET THE STACK POINTER

                                ; RESET PICTURE SYSTEM 2

000500  052767  020000  167670'          BIS     #20000,IOST     ;ISSUE A PSRESET
                                                                ;THIS DOES THE FOLLOWING:
                                                                ;  DIOPSA <= 0
                                                                ;  DMAWC  <= 0
                                                                ;  DMABA  <= 0
                                                                ;  IOST   <= 100200
                                                                ;  DMAPSA <= 0

                                ; INITIALIZE THE DMA

000506                            WTPS    #MPIP,#DMAPSA   ;SET THE DMA TO ADDRESS THE MPIP

                                ; INITIALIZE THE PICTURE PROCESSOR

                                                                ;MAP INPUT CONTROLLER WAS SET
                                                                ;PASSIVE BY PSRESET

000536                  RWPS    BIC     #6,#MMSR        ;ENABLE MAP-CLEAR MAPMNT & MAPHL
                                                        ;MAP INTERNAL REGISTERS (0-57)
000566  012700  010320          MOV     #LOAD+320,R0    ;SET LOAD RSR
000572  012701  177717          MOV     #-61,R1         ;SET TO TRANSFER 61 (OCTAL) WORDS
000576  012702  000000'         MOV     #MAPREG,R2      ;SET BASE ADDRESS FOR TRANSFER
000602  004767  001554          JSR     PC,DMAOUT       ;AND DMA OUT THE DATA

                                                        ;MAP OUTPUT FORMATTER
000606                  RWPS    CLR     #MAOA           ;SET PS2 REFRESH BUFFER LIMITS
000634                          WTPS    #8190.,#MAOL    ;ASSUME 16K MEMORY (DOUBLE BUFFER
000664                  RWPS    BIS     #40,#MSR        ;SET MAP OUTPUT FORMATTER ACTIVE

                                ; INITIALIZE PSMEM LIMITS FOR 16K REFRESH BUFFER

000714  005067  177544          CLR     RBTB-2          ;SET BUFFER 1 LOW START ADDRESS
000720  012767  017776  177540  MOV     #8190.,RBTB     ;SET BUFFER 1 HIGH LIMIT ADDRESS
000726  012767  020000  177534  MOV     #8192.,RBTB+2   ;SET BUFFER 2 LOW START ADDRESS
000734  012767  037776  177530  MOV     #16382.,RBTB+4  ;SET BUFFER 2 HIGH LIMIT ADDRESS
000742  012767  000002  177512  MOV     #2,BUFPTR       ;INITIALIZE THE BUFFER POINTER
                                                        ;BUFPTR & RBTB USED BY RTC ISR
000750                          WTPS    #8192.,#RFASA   ;SET INITIAL REFRESH LIMITS
001000                          WTPS    #8192.,#RFAIL

                                ; INITIALIZE PICTURE GENERATOR STATUS

001030  012701  000004          MOV     #4,R1           ;SET # OF WORDS TO TRANSFER
001034  012702  000142'         MOV     #PGINIT,R2      ;SET BASE ADDRESS
001040                  1$:     WTPS    (R2)+,#LGPIP    ;WRITE DIRECTLY TO LGPIP
001066  005301                  DEC     R1              ;DONE?
001070  003363                  BGT     1$              ;IF GT NO

                                ; INITIALIZE INTERRUPT VECTORS, INTERRUPT ENABLES & DEVICE STATUS

001072  012700  000340          MOV     #LOWVEC,R0      ;SET LOW VECTOR ADDRESS
```

PROGRAM INITIALIZATION

```
                                                        MOV     #RTCINT,(R0)+   ;SET RTC ISR ADDRESS
                                                        MOV     #240,(R0)+      ;AND PSW
                                                        MOV     #NOINT,(R0)+    ;SET SYSTEM ISR ADDRESS
                                                        MOV     #240,(R0)+      ;AND PSW
                                                        MOV     #NOINT,(R0)+    ;SET DEVICE ISR ADDRESS
                                                        MOV     #240,(R0)+      ;AND PSW
                                                        MOV     #NOINT,(R0)+    ;SET DMA ISR ADDRESS
                                                        MOV     #240,(R0)+      ;AND PSW

                                                        WTPS    #1,#RTCIE       ;ENABLE THE RTC INTERRUPT
                                                        WTPS    #16,#RTCCNT     ;SET CLOCK COUNT (60 HZ.)
                                                RWPS    BIS     #1,#RTCSR       ;AND START THE RTC

                                                        BIS     #400,IOST       ;SET PSIE AND DMAIE

                                                ; INITIALIZE VARIABLES

                                                        CLR     NUFRAM
                                                        CLR     TX
                                                        CLR     TY
                                                        CLR     TZ

                                                ; AND OUTPUT AT LEAST THE FIRST FRAME

                                                        JMP     DSPLAY          ;BRANCH AROUND THE DATA INPUT
```

```
                                        .SBTTL  RTC INTERRUPT SERVICE ROUTINE
                                ;
001300  010546                  RTCINT: MOV     R5,-(SP)        ;SAVE R5
001302  010446                          MOV     R4,-(SP)        ;SAVE R4
001304  005767  167670'          1$:     TST     IOST            ;DIO READY?
001310  100375                          BPL     1$              ;IF PL NO...WAIT FOR IT
001312  005767  167662'                  TST     DIOPSA          ;READ THE CURRENT DIOPSA
001316  005767  167670'          2$:     TST     IOST            ;DIO READY?
001322  100375                          BPL     2$              ;IF PL NO...WAIT FOR IT
001324  016746  167660'                  MOV     PSDATA,-(SP)    ;SAVE THE CURRENT DIOPSA
                                ;
001330                          RWPS    TST     #RFSR           ;RFSTOPPED?
001356  100140                          BPL     5$              ;IF PL NO...JUST EXIT
                                ;
001360  005767  177072                  TST     NUFRAM          ;IS A NEW FRAME READY?
001364  001521                          BEQ     4$              ;IF EQ NO...JUST RESTART RFFRESH
                                ;
001366  032767  000200  167670'          BIT     #200,IOST       ;DMA DONE?
001374  001515                          BEQ     4$              ;IF EQ NO...JUST RESTART REFRESH
                                ;
001376                          3$:
001376                          RWPS    TST     #MSR            ;PPDONE?
001424  100364                          BPL     3$              ;IF PL NO...LOOP
001426                                  RDPS    #MAOA,R4        ;GET CURRENT MAOA
001454  016705  177002                  MOV     BUFPTR,R5       ;GET CURRENT BUFFER POINTER
001460                                  WTPS    RBTB(R5),#MAOA  ;SET RB LIMITS FOR NEXT FRAME
001510                                  WTPS    RBTB+2(R5),#MAOL
001540  005405                          NEG     R5              ;TOGGLE BUFFER POINTER
001542  010567  176714                  MOV     R5,BUFPTR       ;SAVE BUFFER POINTER
001546                                  WTPS    RBTB(R5),#RFASA ;SET REFRESH START ADDRESS
001576                                  WTPS    R4,#RFAIL       ;AND REFRESH LIMIT ADDRESS
001624  005067  176626                  CLR     NUFRAM          ;RESET NEW FRAME FLAG
                                ;
                                ; RESTART REFRESH CYCLE
                                ;
001630                          4$:
001630                          RWPS    BIS     #20000,#RFSR    ;RESTART REFRESH
                                ;
                                ; ACKNOWLEDGE RIC INTERRUPT
                                ;
001660                          5$:
001660                          RWPS    BIS     #1,#RTCREQ      ;ACKNOWEDGE INTERRUPT
                                ;
                                ; RESTORE SYSTEM TO STATE PRIOR TO INTERRUPT
                                ;
001710  005767  167670'          NDONE: TST     IOST            ;DIO READY?
001714  100375                          BPL     NDONE           ;IF PL NO...WAIT FOR IT
001716  012667  167662'                  MOV     (SP)+,DIOPSA    ;RESTORE DIOPSA
001722  012604                          MOV     (SP)+,R4        ;RESTORE R4
001724  012605                          MOV     (SP)+,R5        ;RESTORE R5
                                ;
001726  000002                  NOINT:  RTI                     ;RETURN FROM INTERRUPT
                                ;
```

```
                                        .SBTTL  DATA TO INPUT?

001730  016700  177570'               PLOOP:  MOV     SWITCH,R0       ;GET SWITCH VALUE
001734  001775                               BEQ     PLOOP           ;AND LOOP IF NOTHING IS SET
                                        ;
                                        ; CHECK SWITCH 15
                                        ;
001736  032700  100000                       BIT     #100000,R0      ;SW 15 SET?
001742  001402                               BEQ     SW14            ;BRANCH IF NOT
001744  000167  176524                       JMP     START           ;RESTART
                                        ;
                                        ; CHECK SWITCH 14
                                        ;
001750  032700  040000               SW14:   BIT     #40000,R0       ;SW 14 SET?
001754  001403                               BEQ     SW13            ;BRANCH IF NOT
001756  166767  176476  176266               SUB     INCVAL,TX       ;TRANSLATE -X
                                        ;
                                        ; CHECK SWITCH 13
                                        ;
001764  032700  020000               SW13:   BIT     #20000,R0       ;SW 13 SET?
001770  001403                               BEQ     SW12            ;BRANCH IF NOT
001772  166767  176462  176254               SUB     INCVAL,TY       ;TRANSLATE -Y
                                        ;
                                        ; CHECK SWITCH 12
                                        ;
002000  032700  010000               SW12:   BIT     #10000,R0       ;SW 12 SET?
002004  001403                               BEQ     SW11            ;BRANCH IF NOT
002006  166767  176446  176242               SUB     INCVAL,TZ       ;TRANSLATE -Z
                                        ;
                                        ; CHECK SWITCH 11
                                        ;
002014  032700  004000               SW11:   BIT     #4000,R0        ;SW 11 SET?
002020  001403                               BEQ     SW10            ;BRANCH IF NOT
002022  066767  176432  176222               ADD     INCVAL,TX       ;TRANSLATE +X
                                        ;
                                        ; CHECK SWITCH 10
                                        ;
002030  032700  002000               SW10:   BIT     #2000,R0        ;SW 10 SET?
002034  001403                               BEQ     SW09            ;BRANCH IF NOT
002036  066767  176416  176210               ADD     INCVAL,TY       ;TRANSLATE +Y
                                        ;
                                        ; CHECK SWITCH 9
                                        ;
002044  032700  001000               SW09:   BIT     #1000,R0        ;SW 9 SET?
002050  001403                               BEQ     DSPLAY          ;BRANCH IF NOT
002052  066767  176402  176176               ADD     INCVAL,TZ       ;TRANSLATE +Z
```

```
                                        .SBTTL  PICTURE DISPLAY
                                        DSPLAY:
                                        ;
                                        ; SET BLINK MODE ON
                                        ;
002060  012700  074377                          MOV     #DRAW2D+34377,R0 ;SET THE 2DDRAW COMMAND
                                                                         ;FSM1=0, FSM2=7
002064  012701  177776                          MOV     #-2.,R1          ;2 WORDS TO BE OUTPUT
002070  012702  000152'                        MOV     #BLNKON,R2       ;SET THE BLINK BASE ADDRESS
                                        ;
                                        ; WAIT FOR THE NEW FRAME FLAG TO BE CLEARED BEFORE OUTPUT
                                        ;
002074  005767  176356                  DSP010: TST     NUFRAM           ;IS IT =0 YET?
002100  001375                                  BNE     DSP010           ;BRANCH IF NOT
                                        ;
002102  004767  000254                          JSR     PC,DMAOUT        ;NEW FRAME... DMA IT OUT
                                        ;
                                        ; RESET THE BASE REGISTER
                                        ;
002106  012700  140377                          MOV     #SETBAS,R0       ;SET THE SETBASE COMMAND
002112  012701  177774                          MOV     #-4.,R1          ;4 WORDS TO BE LOADED
002116  012702  000262'                        MOV     #ZWBASE,R2       ;SET THE BASE ADDRESS
002122  004767  000234                          JSR     PC,DMAOUT        ;AND DMA IT OUT
                                        ;
                                        ; DO THE SET-POINT FOR THE TEXT
                                        ;
002126  012700  050377                          MOV     #DRAW2D+10377,R0 ;SET THE 2DDRAW COMMAND
                                                                         ;FSM1=0, FSM2=2
002132  012701  177776                          MOV     #-2.,R1          ;2 WORDS TO BE OUTPUT
002136  012702  000272'                        MOV     #SETPT,R2        ;SET THE DATA ADDRESS
002142  004767  000214                          JSR     PC,DMAOUT        ;AND DMA IT OUT
                                        ;
                                        ; NOW OUTPUT THE ASCII CHARACTERS
                                        ;
002146  012700  074374                          MOV     #DRAW2D+34374,R0 ;SET THE 2DDRAW COMMAND
                                                                         ;FSM1=0, FSM2=7
002152  012701  177770                          MOV     #-8.,R1          ;8 WORDS OF CHARACTERS (2/WORD)
002156  012702  000276'                        MOV     #TEXT,R2         ;SET THE TEXT ADDRESS
002162  004767  000174                          JSR     PC,DMAOUT        ;AND DMA IT OUT
                                        ;
                                        ; SET BLINK MODE OFF
                                        ;
002166  012700  074377                          MOV     #DRAW2D+34377,R0 ;SET THE 2DDRAW COMMAND
                                                                         ;FSM1=0, FSM2=7
002172  012701  177776                          MOV     #-2.,R1          ;2 WORDS TO BE OUTPUT
002176  012702  000156'                        MOV     #BLNKOF,R2       ;SET THE BLINK OFF BASE ADDRESS
002202  004767  000154                          JSR     PC,DMAOUT        ;AND DMA IT OUT
                                        ;
                                        ; PUSH THE CURRENT TRANSFORMATION MATRIX
                                        ;
002206  012700  022360                          MOV     #PUSH,R0         ;SET THE PUSH COMMAND
002212  005001                                  CLR     R1               ;A DATALESS COMMAND
                                                                         ;R2 CAN BE ANYTHING
002214  004767  000142                          JSR     PC,DMAOUT        ;AND DMA IT OUT
                                        ;
                                        ; CONCATENATE THE PERSPECTIVE MATRIX
```

```
                                ;
002220  012700  026000          ;       MOV     #MATCON,R0      ;SET THE MATCON COMMAND
002224  012701  177760          ;       MOV     #-16.,R1        ;16 WORDS WILL BE OUTPUT
002230  012702  000162'         ;       MOV     #PERMAT,R2      ;SET THE MATRIX BASE ADDRESS
002234  004767  000122          ;       JSR     PC,DMAOUT       ;AND DMA IT OUT
                                ;
                                ; CONCATENATE THE TRANSLATION MATRIX
                                ;
002240  012700  026000          ;       MOV     #MATCON,R0      ;SET THE MATCON COMMAND
002244  012701  177760          ;       MOV     #-16.,R1        ;16 WORDS WILL BE OUTPUT
002250  012702  000222'         ;       MOV     #TRNMAT,R2      ;SET THE MATRIX BASE ADDRESS
002254  004767  000102          ;       JSR     PC,DMAOUT       ;AND DMA IT OUT
                                ;
                                ; RESET THE BASE REGISTER
                                ;
002260  012700  140377          ;       MOV     #SETBAS,R0      ;SET THE SETBASE COMMAND
002264  012701  177774          ;       MOV     #-4,R1          ;4 WORDS TO BE LOADED
002270  012702  000262'         ;       MOV     #ZWBASE,R2      ;SET THE BASE ADDRESS
002274  004767  000062          ;       JSR     PC,DMAOUT       ;AND DMA IT OUT
                                ;
                                ; NOW DRAW THE CUBE (IN 2 INSTALLMENTS)
                                ;
002300  012700  121366          ;       MOV     #DRAW3D+21366,R0 ;SET THE BASIC 3D DRAW COMMAND
                                ;                               ;FSM1=2, FSM2=4
002304  012701  177742          ;       MOV     #-30.,R1        ;30 WORDS TO BE OUTPUT
002310  012702  000316'         ;       MOV     #CDAT1,R2       ;SET THE DATA BASE ADDRESS
002314  004767  000042          ;       JSR     PC,DMAOUT       ;AND DMA IT OUT
                                ;
002320  012700  120372          ;       MOV     #DRAW3D+20372,R0 ;SET THE BASIC 3D DRAW COMMAND
                                ;                               ;FSM1=0, FSM2=4
002324  012701  177756          ;       MOV     #-18.,R1        ;18 WORDS TO BE OUTPUT
002330  012702  000412'         ;       MOV     #CDAT2,R2       ;SET THE DATA BSE ADDRESS
002334  004767  000022          ;       JSR     PC,DMAOUT       ;AND DMA IT OUT
                                ;
                                ; INITIATE THE DISPLAY OF THE DATA (SET THE NEW FRAME FLAG)
                                ;
002340  005267  176112          ;       INC     NUFRAM
                                ;
                                ; POP THE ORIGINAL TRANSFORMATION MATRIX BACK
                                ;
002344  012700  024360          ;       MOV     #POP,R0         ;SET THE POP COMMAND
002350  005001                  ;       CLR     R1              ;A DATALESS COMMAND
002352  004767  000004          ;       JSR     PC,DMAOUT       ;AND DMA IT OUT
                                ;
                                ; RESTART THE DISPLAY LOOP
                                ;
002356  000167  177346          ;       JMP     PLOOP           ;AND CONTINUE
```

```
                                        .SBTTL  DMA OUTPUT ROUTINE
                                        ;
                                        ; THIS ROUTINE DOES ALL OF THE DMA OUTPUT FOR THIS PROGRAM.
                                        ;       EXPECTED PARAMETERS ARE:
                                        ;               R0=RSR COMMAND
                                        ;               R1=DMA WORD COUNT FOR DRWC REGISTER
                                        ;               R2=DMA BASE ADDRESS FOR DRBA REGISTER
                                        ;
002362  032767  000200  167670' DMAOUT: BIT     #200,IOST       ;IS THE DMA READY?
002370  001774                          BEQ     DMAOUT          ;IF EQ NO...WAIT FOR IT

002372                          1$:
002372                  RWPS    TST     #MSR            ;PPDONE?
002420  100364                          BPL     1$              ;IF PL NO...WAIT FOR IT

002422                                  WTPS    R0,#MPIP        ;OUTPUT THE RSR
002450  010167  167664'                MOV     R1,DMAWC        ;SET THE DMA WORD COUNT
002454  002005                          BGE     2$              ;IF GE NOTHING TO DO
002456  010267  167666'                MOV     R2,DMABA        ;SET THE DMA BASE ADDRESS
002462  052767  000001  167670'        BIS     #1,IOST         ;INITIATE THE TRANSFER
002470  000207                  2$:     RTS     PC              ;AND RETURN

        000474'                         .END    START
```

```
APPENDIX A:  SAMPLE PROGRAM     RT-11 MACRO VM02-10   22-NOV-76 08:00:12 PAGE 10+
SYMBOL TABLE

BEGIN   000000R          BLNKOF  000156R          BLNKON  000152R
BUFPTR  000462R          CDAT1   000316R          CDAT2   000412R
DIOPSA= 167662           DMABA  = 167666          DMAOUT  002362R
DMAPSA= 177727           DMAWC  = 167664          DRAW2D= 040000
DRAW3D= 100000           DSPLAY  002060R          DSP010  002074R
INCVAL  000460R          IOST   = 167670          LGPIP  = 177775
LOAD   = 010000          LOWVEC= 000340           MAIA   = 177752
MAOA   = 177751          MAOL   = 177750          MAPREG  000000R
MATCON= 026000           MMSR   = 177754          MPIP   = 177777
MSR    = 177753          NDONE   001710R          NOINT   001726R
NUFRAM  000456R          PC     =%000007          PERMAT  000162R
PGINIT  000142R          PLOOP   001730R          POP    = 024360
PSDATA= 167660           PUSH   = 022360          RBTB    000466R
RFAIL  = 177736          RFASA  = 177735          RFSR   = 177737
RTCCNT= 177724           RTCIF  = 177761          RTCINT  001300R
RTCREQ= 177760           RTCSR  = 177725          R0     =%000000
R1     =%000001          R2     =%000002          R3     =%000003
R4     =%000004          R5     =%000005          SETBAS= 140377
SETPT   000272R          SP     =%000006          START   000474R
SWITCH= 177570           SW09    002044R          SW10    002030R
SW11    002014R          SW12    002000R          SW13    001764R
SW14    001750R          TEXT    000276R          TRNMAT  000222R
TX      000252R          TY      000254R          TZ      000256R
ZWBASE  000262R
. ABS.  000000      000
        002472      001
ERRORS DETECTED: 0
FREE CORE: 17515. WORDS

SAMPLE,SAMPLE/N:REX/C=SAMPLE
```

CROSS REFERENCE TABLE (CREF V01-03)

```
        5-68*   5-69*   6-10    6-12    6-22    6-24    6-26*
        6-27*   6-30*   6-31*   6-37*   6-42*  10-14   10-17*
PUSH    3-43#   9-52
RBTB    4-76#   5-38*   5-39*   5-40*   5-41*   6-26    6-27
        6-30
RFAIL   3-22#   5-45    6-31
RFASA   3-21#   3-22    3-23    5-44    6-30
RFSR    3-23#   6-12    6-37
RTCCNT  3-25#   3-26    5-68
RTCIE   3-37#   5-67
RTCINT  5-58    6-3#
RTCREQ  3-36#   3-37    6-42
RTCSR   3-26#   5-69
SETBAS  3-48#   9-21    9-73
SETPT   4-58#   9-31
START   5-3#    7-10   10-24
SW09    7-39    7-44#
SW10    7-33    7-38#
SW11    7-27    7-32#
SW12    7-21    7-26#
SW13    7-15    7-20#
SW14    7-9     7-14#
SWITCH  3-19#   7-3
TEXT    4-59#   9-39
TRNMAT  4-45#   9-68
TX      4-48#   5-76*   7-16*   7-34*
TY      4-49#   5-77*   7-22*   7-40*
TZ      4-50#   5-78*   7-28*   7-46*
ZWBASE  4-54#   9-23    9-75
```

APPENDIX A: SAMPLE PROGRAM RT-11 MACRO VM02-10 22-NOV-76 08:00:21
TABLE OF CONTENTS

APPENDIX B

PDP-11 FORTRAN CALLING SEQUENCE CONVENTION

B.1 Introduction

This calling sequence convention is compatible with all PDP-11 processor options (including use of distint Instruction and Data Space capabilties of the KT-11 Memory Management Option), provides both reentrant and non-reentrant forms and is as fast and short as possible, consistent with these requirements. This description is oriented toward the programmeer who wishes to write assembly language routines which can be called by or which call FORTRAN-compiled routines.

B.2 Subroutine Linkage

All instances of subprogram linkage are performed in the same manner, including linkage of user written FORTRAN subprograms, and assembly following instruction:

```
JSR     PC,routine
```

Register five (R5) contains the address of an argument list having the following format:

```
.BYTE   N,0             ;N=NUMBER OF ARGUMENTS
                        ;BYTE 1 IS UNDEFINED
.WORD   ARG1            ;ADDRESS OF ARGUMENT 1
.WORD   ARG2            ;ADDRESS OF ARGUMENT 2
  .       .
  .       .
  .       .
.WORD   ARGN            ;ADDRESS OF ARGUMENT N
```

Control is return to the calling program by use of the instruction:

```
RTS     PC
```

An assembly language subroutine to find the sum of any number of integers using the following call:

```
CALL ADD(NUM1,NUM2,...NUMN,ISUM)
```

might look like the following:

```
        .TITLE  ADD
        .GLOBL  ADD
ADD:    MOV     (R5)+,R0        ;GET # OF ARGUMENTS
        CLR     R1              ;PREPARE WORKING REG.
        DECB    R0              ;FIND # OF TERMS TO ADD
1$:     ADD     @(R5)+,R1       ;ADD NEXT TERM
        DECB    R0              ;DECREMENT COUNTER
        BNE     1$              ;LOOP IF NOT DONE
        MOV     R1,@(R5)+       ;RETURN RESULT
        RTS     PC              ;RETURN CONTROL
```

B.3 Subroutine Register Usage

A subroutine that is called by a FORTRAN program need not preserve any register. Howerver, the stack must be kept in sync; that is each "push" onto the stack must be matched by a "pop" from the stack prior to leaving the routine.

User-written assembly language programs that call FORTRAN subroutines must preserve any pertinent registers before calling the FORTRAN routine and restore the registers, if necessary, upon return.

Function subroutines return a single result in the registers. The register assignments for returning the different variable types are listed below:

Integer and Logical functions - result in R0.

Real functions - high-order result in R0, low-order result in R1.

Double Precision functions - result in R0-R3, lowest-order result in R3.

Complex functions - high-order real result in R0, low-order real result in R1, high-order imaginary result in R2, low-order imaginary result in R3.

APPENDIX C

USE OF THE GRAPHICS SOFTWARE WITH THE DOS/BATCH DISK OPERATING SYSTEM

C.1 Use of the Graphics Software Package

The Graphics Software Package is available to the DOS/BATCH user as a library of catalogued object modules which may be linked with the user's FORTRAN or MACRO-11 Program to form a graphics application program. The PICTURE SYSTEM Graphics Library (PICLIB), which contains all of the subroutines described in Chapter 4, is searched by the linker (LINK) to load those subroutines called by the user program. The resulting program forms a load module (LDA format) which may be executed upon user demand.

C.2 Use of PDP-11 FORTRAN IV with PICTURE SYSTEM 2

DOS/BATCH FORTRAN conforms to the specifications for American National Standard FORTRAN and is also highly compatable with IBM 1130 FORTRAN. DOS/BATCH FORTRAN programs can be compiled and run on any PICTURE SYSTEM 2 configuration that supports the DOS/BATCH Operating System, and which has a minimum of 16k of memory. DOS/BATCH FORTRAN supports all standard hardware options supported by the operating system.

Graphics applications programs written using FORTRAN interface to PICTURE SYSTEM 2 by means of the subroutines contained in the Graphics Library (PICLIB). All FORTRAN statements and functions are available to the user of PICTURE SYSTEM 2; however, the following should be stressed:

1. All parameters passed to the subroutines of the Graphics Library are integers. Should a REAL constant or variable be passed as a parameter to a graphics subroutine, the sign, binary excess 128 exponent and high-order mantissa will be treated as an integer.

2. The "one word integers" switch (/ON) must be specified to the FORTRAN compiler to ensure that the elements of integer arrays are contiguous in memory as required by the graphics software.

Figure C-1 outlines the steps required to prepare a FORTRAN source program for execution under the DOS/BATCH monitor: (1) Compilation, (2) Linking and (3) Execution.

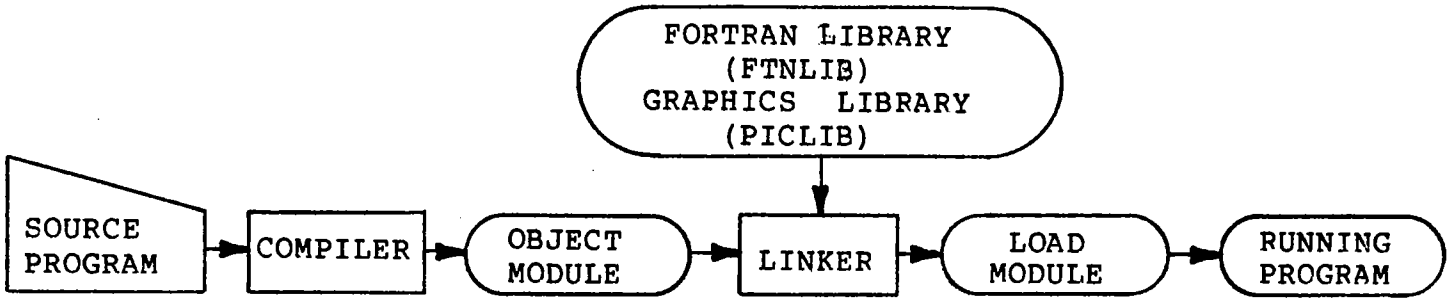


Figure C-1

Steps in Compiling and Executing a FORTRAN Graphics Program

Step 1 in Figure C-1 is initiated by a call to the FORTRAN Compiler, accompanied by a command string that describes input and output files, and switch options to be used by the Compiler. Step 2 is initiated by a call to the Linker, accompanied by a similar command string. Step 3 is initiated upon user keyboard request or a user programmed request.

Step 1: The DOS/BATCH FORTRAN compiler accepts a standard DOS command string of the form:

#object module,listing<source/options

A typical FORTRAN command string is of the form:

#SY:PROG1.OBJ,SY:PROG1.LST<SY:PROG1.FTN/ON

or

#PROG1,PROG1<PROG1/ON

(device SY: is assumed to be the default device just as the filename extensions .OBJ, .LST and .FTN are the default filename extensions when not specified.)

In the above example, the user should note the use of the "one word integers" switch (/ON) in the source file specification:

<PROG1.FTN/ON

Step 2: The DOS/BATCH Linker accepts a standard DOS command string of the form:

#load module,load map,symbol table<object modules/E

A typical LINK command string is of the form:

#SY:PROG1.LDA,SY:PROG1.MAP,SY:PROG1.STB<SY:PROG1.OBJ
#SY:PICLIB.OBJ,SY:FTNLIB.OBJ/E

or

#PROG1,PROG1,PROG1<PROG1,PICLIB,FTNLIB/E

(device SY: is assumed to be the default device, just as the filename extensions .LDA, .MAP, .STB and .OBJ are the default filename extensions when not specified.)

In the above example, the user should note the specification of the PICTURE SYSTEM Graphics Library (PICLIB) and the FORTRAN OTS Library (FTNLIB). These libraries are searched to resolve all global references for the load module. These libraries (PICLIB) and (FTNLIB) reside in the system area [1,1] and are therefore available to all users. Note: The Linker searches the user's [UIC] area for all object files specified. If an object file is not found the system area [1,1] is searched regardless of the user UIC.

Step 3: To run a load module which has been created by the Linker a user need only request the monitor to run the program. This is accomplished by the monitor command:

$RUN SY:PROG1.LDA

or

$RUN PROG1

(device SY: is assumed the default device, just as the filename extension .LDA is the default filename extension when not specified.)

The following is a typical listing which illustrates the process described by Figure C-1 and Steps 1, 2 and 3 above.

```
$LOG 202,112
DATE:-22-NOV-76
TIME:-10:30:20
$RUN FORTRN
FORTRAN  V06.13
#PROG1,KB:<PROG1/ON

FORTRAN V06.13                      10:30:52     22-NOV-76    PAGE     1

        C FORTRAN DEMONSTRATION PROGRAM
        C
0001            DIMENSION IHOUSE(14)
        C
0002            DATA IHOUSE/-10000,10000, -10000,-10000, 10000,-10000,
                1 10000,10000, -10000,10000, 0,20000, 10000,10000/
        C
        C INITIALIZE PICTURE SYSTEM 2
        C
0003            CALL PSINIT(2,0)
        C
        C DRAW THE DATA
        C
0004            CALL DRAW2D(IHOUSE,7,2,2,0)
        C
        C AND DISPLAY THE "NEW FRAME"
        C
0005            CALL NUFRAM
        C
0006            PAUSE
        C
0007            STOP
0008            END


        ROUTINES CALLED:
        PSINIT, DRAW2D, NUFRAM

        OPTIONS =/ON,/OP:1

        BLOCK        LENGTH
        MAIN.   62      (000174)*

        **COMPILER ----- CORE**
           PHASE       USED  FREE
        DECLARATIVES 00622 01983
        EXECUTABLES  00702 01903
        ASSEMBLY     00880 06365

#^C
,KI

$
```

```
$RUN LINK

LINK V01-04
#PROG1,PROG1<PROG1,PICLIB,FTNLIB/E
%DIAG*-1 UNDEFINED SYMBOLS

SPACE USED 004630, SPACE FREE 024274
#^C
.KILL

$RUN PROG1
```

APPENDIX D

USE OF THE GRAPHICS SOFTWARE WITH THE RT-11 OPERATING SYSTEM

D.1 Use of the Graphics Software Package

The Graphics Software Package is available to the RT-11 user as a library of catalogued object modules which may be linked with the user's FORTRAN or MACRO-11 program to form graphics application programs. PICTURE SYSTEM Graphics Library (PICLIB), which contains all of the subroutines described in Chapter 4, is searched by the Linker (LINK) to load those subroutines called by the user program. The resulting program forms a load module (SAV format) which may be executed upon user demand.

D.2 Use of PDP-11 FORTRAN IV with PICTURE SYSTEM 2

RT-11 FORTRAN conforms to the specifications for American National Standard FORTRAN and is also highly compatible with IBM 1130 FORTRAN. RT-11 FORTRAN programs can be compiled and run on any PICTURE SYSTEM 2 configuration that supports the RT-11 Operating System, and which has a minimum of 8K of memory RT-11 FORTRAN supports all standard hardware options supported by the operating system.

Graphics applications programs written using FORTRAN interface to PICTURE SYSTEM 2 by the subroutines contained in the Graphics Library (PICLIB). All FORTRAN statements and functions are available to the user of PICTURE SYSTEM 2. However, the following should be stressed to the PICTURE SYSTEM FORTRAN user:

> All parameters passed to the subroutines of the Graphics Library are integers. Should a REAL parameter be passed as a parameter to a graphics subroutine, the sign, binary excess 128 exponent and high-order mantissa will be treated as an integer.

Figure D-1 outlines the steps required to prepare a FORTRAN source program for execution under the RT-11 Monitor: (1) Compilation, (2) Linking, and (3) Execution.

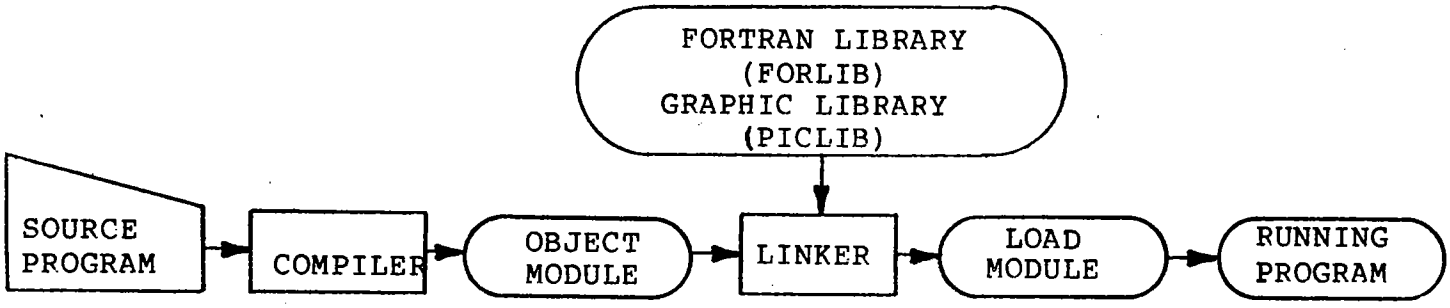


Figure D-1

Steps in Compiling and Executing
a FORTRAN Graphics Program

Step 1 in Figure D-1 is initiated by a call to the FORTRAN compiler, accompanied by a command string that describes the input and output files, and switch options to be used by the Compiler. Step 2 is initiated by a call to the LINKER, accompanied by a similar command string. Step 3 is initiated upon user keyboard request or a user programmed request.

Step 1: The RT-11 FORTRAN compiler accepts a command string of the form:

*object module,listing=source/options

A typical FORTRAN command string is of the form:

*SY:PROG1.OBJ,SY:PROG1.LST=SY:PROG1.FOR
or
*PROG1,PROG1=PROG1

(device SY: is assumed to be the default device, just as the filename extensions .OBJ, .LST and .FOR are the default filename extensions when not specified.)

Step 2: The RT-11 Linker accepts a command string of the form:

*load module,load map=object modules/switches

A typical LINK command string is of the form:

```
*SY:PROG1.SAV,SY:PROG1.MAP=SY:PROG1.OBJ,PICLIB.OBJ/F
     or
*PROG1,PROG1=PROG1,PICLIB/F
```

(device SY: is assumed to be the default device, just as the filename extensions .SAV .MAP and .OBJ are the default filename extensions when not specified.)

In the above example, the user should note the explicit specification of PICTURE SYSTEM Graphics Library (PICLIB) and the FORTRAN OTS Library (FORLIB) by the /F switch. These libraries are searched to resolve all global references for the load module.

Step 3: To run a load module which has been created by the Linker, a user need only request the monitor to run the program. This is accomplished by the monitor command:

```
.RUN SY:PROG1.SAV
     or
.RUN PROG1
```

(device SY: is assumed the default device, just as the filename extension .SAV is the default filename extension when not specified).

The following is a typical listing which illustrates the process described by Figure D-1 and steps 1, 2 and 3 above.

Note:

Since the user's program is linked with PSINIT which contains an interrupt handler for PICTURE SYSTEM 2, care must be taken that the RT-11 USR does not swap-in over that interrupt handler. This may be done by issuing the following RT-11 keyboard command before running the graphics program:

```
.SET USR NOSWAP
```

If this is not done and the USR does swap over the interrupt handler, a fatal system error may occur.

```
RT-11FB    V02B-05

.DATE 22-NOV-76

.TIME 8:30

.RUN FORTRAN
*PROG1,TT:=PROG1


RT-11 FORTRAN IV          V01B-08     MON 22-NOV-76 08:30:19          PAGE 001

      C FORTRAN DEMONSTRATION PROGRAM
      C
0001        DIMENSION IHOUSE(14)
      C
0002        DATA IHOUSE/-10000,10000, -10000,-10000, 10000,-10000,
     1 10000,10000, -10000,10000, 0,20000, 10000,10000/
      C
      C INITIALIZE PICTURE SYSTEM 2
      C
0003        CALL PSINIT(2,0)
      C
      C DRAW THE DATA
      C
0004        CALL DRAW2D(IHOUSE,7,2,2,0)
      C
      C AND DISPLAY THE "NEW FRAME"
      C
0005        CALL NUFRAM
      C
0006        PAUSE
      C
0007        STOP
0008        END


RT-11 FORTRAN IV          STORAGE MAP

NAME     OFFSET   ATTRIBUTES

IHOUSE   000006   INTEGER*2 ARRAY (14)
PSINIT   000000   REAL*4     PROCEDURE
DRAW2D   000000   REAL*4     PROCEDURE
NUFRAM   000000   INTEGER*2 PROCEDURE
*^C

.RUN LINK
*PROG1,PROG1=PROG1,PICLIB/F

*^C

.RUN PROG1

PAUSE --

STOP --
```

D.3 RT-11FB and the PICTURE SYSTEM Graphics Software

Graphics applications programs which successfully execute under the RT-11SJ monitor or as a RT-11FB Background program may be re-linked for execution as a Foreground program without modification of the source program to provide time for Background execution. Control should be relinguished to the Background program whenever possible to provide for successful execution of the Background program. The Graphics Software provides time for Background execution by relinguishing control whenever the graphics program is a Foreground job (determined dynamically by the Software) and has to wait before data can be output to the refresh buffer. In this way, the user may develop applications which may then be run as either a Foreground or Background job without special programming required. In addition, programs are more easily transported from operating system to operating system (i.e., RT-11 to RSX-11M) without regard to special changes which would be required otherwise.

Section D.2 describes the process required to link a program for execution under the RT-11SJ monitor or as a Background job. The following details the steps required to link and execute a graphics program as a Foreground job.

Step 1: Step 1 (the compilation or assembly of the source program) is the same as required to compile or assemble a program for execution as a Background job.

Step 2: The RT-11 Linker accepts a command string of the form:

*load module,load map=object modules/switches

A typical LINK command string for a Foreground program is of the form:

*SY:PROG1.REL,SY:PROG1.MAP=SY:PROG1.OBJ,PICLIB.OBJ/F/R
or
*PROG1,PROG1=PROG1,PICLIB/F/R

(device SY: is assumed to be the default device, just as the filename extensions .REL, .MAP and .OBJ are the default filename extensions when not specified.)

In the above, the user should note the explicit specification of PICTURE SYSTEM Graphics Library (PICLIB), the FORTRAN OTS Library (FORLIB) by the /F switch and the specification that a Foreground program is to be produced by the /R switch.

Step 3: To execute a Foreground load module (.REL format) which has been created by the Linker, a user need only request the monitor to run the program in the Foreground. This is accomplished by the monitor command:

```
.FRUN SY:PROG1.REL
     or
.FRUN PROG1
```

(device SY: is assumed the default device, just as the filename extension .REL is the default filename extension when not specified).

If the RT-11FB monitor prints the message

"? F ACTIVE?"

then this is an indication that a Foreground job already exists and is active. To abort the execution of an active Foreground job, the user should input the following:

```
^F (Control F)
^F>^C (Control C)
^C (Control C)
.
```

To recover that area of memory in which the Foreground job resided, the user may then enter the following command:

```
.UNLOAD FG
```

Since the FORTRAN Object Time System needs work areas to perform some of its functions, the FRUN/N:X monitor command must be used to allocate the needed space when running a FORTRAN program as a Foreground job. The following formula can be used to calculate the number (X) which must be designated to the /N option:

$$x = y + 21 \text{ octal } (N) + (R-210 \text{ octal}) + A*400 \text{ octal}$$

where:

y = 264 octal.
n = the value of the /N FORTRAN compiler switch option (the compiler defaults this to 6 octal).
R = the value of the /R FORTRAN compiler switch option (the compiler defaults this to 210 octal, 136 decimal).
A = the number of I/O buffers in simultaneous use during execution (Console terminal I/O does not require any

buffers. Each open logical unit typically requires one buffer).

For example, to calculate X for a program (PROG1.REL) which uses only terminal I/O and allows the compiler to assign the standard defaults, set the formula as follows (all values are octal):

X = 264 + 21 (6) + (210-210) + 0*400

Thus, 432 octal words are required. The FORTRAN program is executed using the FRUN command as shown:

.FRUN PROG1/N:432

The following is a typical listing which illustrates the process described by Figure D-1 and steps 1,2, and 3 above:

```
RT-11FB    V02B-05

.DATE 22-NOV-76

.TIME 9:00

.RUN FORTRAN
*PROG1,TT:=PROG1

RT-11 FORTRAN IV          V01B-08     MON 22-NOV-76 09:00:17        PAGE 001

      C FORTRAN DEMONSTRATION PROGRAM
      C
0001        DIMENSION IHOUSE(14)
      C
0002        DATA IHOUSE/-10000,10000, -10000,-10000, 10000,-10000,
           1 10000,10000, -10000,10000, 0,20000, 10000,10000/
      C
      C INITIALIZE PICTURE SYSTEM 2
      C
0003        CALL PSINIT(2,0)
      C
      C DRAW THE DATA
      C
0004        CALL DRAW2D(IHOUSE,7,2,2,0)
      C
      C AND DISPLAY THE "NEW FRAME"
      C
0005        CALL NUFRAM
      C
0006        PAUSE
      C
0007        STOP
0008        END


RT-11 FORTRAN IV          STORAGE MAP

NAME    OFFSET   ATTRIBUTES

IHOUSE  000006   INTEGER*2 ARRAY (14)
PSINIT  000000   REAL*4    PROCEDURE
DRAW2D  000000   REAL*4    PROCEDURE
NUFRAM  000000   INTEGER*2 PROCEDURE
*^C

.RUN LINK
*PROG1,PROG1=PROG1,PICLIB/F/R

*^C

.
```

```
.FRUN PROG1/N:1532


F>

PAUSE --
B>
.^C

.
F>
RES

STOP --

B>
```

APPENDIX E

USE OF THE GRAPHICS SOFTWARE WITH THE RSX-11M REAL-TIME OPERATING SYSTEM

E.1 Use of the Graphics Software Package

The Graphics Software Package is available to the RSX-11M user as a library of catalogued object modules which may be linked with the user's FORTRAN or MACRO-11 program to form graphics application programs. The PICTURE SYSTEM 2 Graphics Library (PICLIB), which contains all the subroutines described in the PICTURE SYSTEM 2 User's Manual, is searched by the Task Builder (TKB) to load those subroutines called by the user program. The resulting program forms a task image (TSK format) which may be executed upon user demand.

E.2 Use of PDP-11 FORTRAN IV with the PICTURE SYSTEM

RSX-11M FORTRAN conforms to the specifications for American National Standard FORTRAN and is also highly compatible with IBM 1130 FORTRAN. RSX-11M FORTRAN programs can be compiled and run on any PICTURE SYSTEM configuration that supports the RSX-11M Operating System, and which has had SYSGEN'ed into it a RSX-11M Device Driver for PICTURE SYSTEM 2. RSX-11M FORTRAN supports all standard hardware options supported by the operating system.

Graphics application tasks written using FORTRAN interface to PICTURE SYSTEM 2 by the subroutines contained in the Graphics Library (PICLIB). All FORTRAN statements and functions are available to the user of PICTURE SYSTEM 2; however, the following should be stressed to the PICTURE SYSTEM FORTRAN user:

> All parameters passed to the subroutines of the Graphics Library are integers. Should a REAL parameter be passed as a parameter to a graphics subroutine, the sign, binary excess 128 exponent and high-order mantissa will be treated as an integer.

Figure E-1 outlines the steps required to prepare a FORTRAN source program for execution under RSX-11M: (1) Compilation, (2) Task Building and (3) Execution.

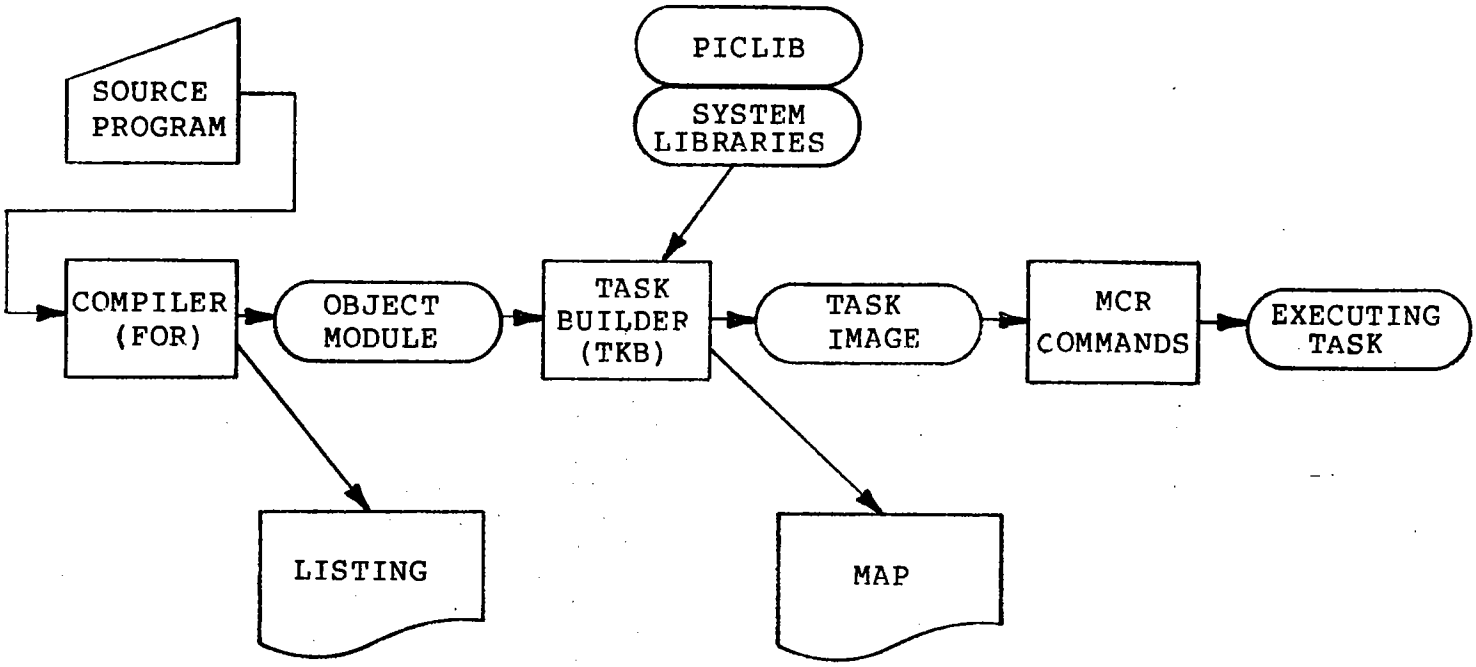


Figure E-1

Steps in Compiling and Executing a FORTRAN Graphics Program

Step 1 in Figure E-1 is initiated by a call to the FORTRAN compiler, accompanied by a command string that describes input and output files, and switch options to be used by the computer. Step 2 is initiated by a call to the Task Builder, accompanied by a similar command string. Step 3 is initiated upon an MCR keyboard request or can be requested to execute by another Task.

Step 1: The RSX-11M FORTRAN compiler accepts a command string of the form:

FOR>object module,listing=source/options

A typical FORTRAN command string is of the form:

FOR>SY:PROG1.OBJ,SY:PROG1.LST=SY:PROG1.FTN
or
FOR>PROG1,PROG1=PROG1

(device SY: is assumed the default device just as the filename extensions .OBJ, .LST, and .FTN are the default filename extensions when not specified.)

Step 2: The RSX-11M Task Builder accepts a standard RSX-11M command string of the form:

```
TKB>task-image,map,symbol-table=object modules,...
```

A typical TKB command string is of the form:

```
TKB>SY:PROG1.TSK,SY:PROG1.MAP,SY:PROG1.STB=SY:PROG1.OBJ
TKB>SY:[1,1]PICLIB.OLB
TKB>//
     or
TKB>PROG1,PROG1=PROG1,[1,1]PICLIB/LB
TKB>//
```

(device SY: is assumed to be the default device, just as the filename extensions .TSK, .MAP, .STB, .OBJ and .OLB are the default filename extensions when not specified.)

In the above examples, the user should note that device PS0: is assigned to logical unit 4 by PSINIT. When PSINIT is called if logical unit 4 or PS0: is not available, the following graphics error will be printed at the user's terminal indicating a fatal error:

```
ERROR 4 DETECTED IN GRAPHICS SUBROUTINE 1.
```

See Section E-3 for a more detailed discussion of the use of the Graphics Software under RSX-11M.

Step 3: To execute a task which has been created by the Task Builder, the user need only request RSX-11M to run the program. This is accomplished by the following MCR command:

```
MCR>RUN SY:PROG1.TSK
      or
MCR>RUN PROG1
```

(device SY: is assumed the default device, just as the filename extension .TSK is the default filename extension when not specified.)

PROG1 will then execute as soon as the partition (or sub-partition) into which the task was built, by the Task Builder, is free. If during Step 2 above, no explicit partition was specified in which the task was to run, the Task Builder binds the task to the default GEN partition.

The following is a typical listing which illustrates the process described by Figure E-1 and Steps 1, 2 and 3 above:

```
  RSX-11M V02 BL12    48K     MAPPED
>RED DK0:=SY0:
>MOU DK0:
>@[1,2]STARTUP
>! PLEASE ENTER DATE AND TIME
>@ <EOF>
>TIME 11/22/76 9:30
>SET /UIC=[202,112]
>FOR PROG1,TI:=PROG1

FORTRAN IV      V01B-02          MON 22-NOV-76 09:30:34          PAGE 001
CORE=08K, UIC=[202,112]                                  PROG1,TI:=PROG1

      C FORTRAN DEMONSTRATION PROGRAM
      C
0001        DIMENSION IHOUSE(14)
      C
0002        DATA IHOUSE/-10000,10000, -10000,-10000, 10000,-10000,
           1 10000,10000, -10000,10000, 0,20000, 10000,10000/
      C
      C INITIALIZE PICTURE SYSTEM 2
      C
0003        CALL PSINIT(2,0)
      C
      C DRAW THE DATA
      C
0004        CALL DRAW2D(IHOUSE,7,2,2,0)
      C
      C AND DISPLAY THE "NEW FRAME"
      C
0005        CALL NUFRAM
      C
0006        PAUSE
      C
0007        STOP
0008        END


FORTRAN IV      STORAGE MAP

NAME    OFFSET  ATTRIBUTES

IHOUSE  000006  INTEGER*2 ARRAY (14)
PSINIT  000000  REAL*4    PROCEDURE
DRAW2D  000000  REAL*4    PROCEDURE
NUFRAM  000000  INTEGER*2 PROCEDURE

>TKB PROG1,PROG1=PROG1,[1,1]PICLIB/LB
>RUN PROG1
TT0  --  PAUSE

>RES TT0
TT0  --  STOP
>
>
```

E.3 RSX-11M and the PICTURE SYSTEM 2 Graphics Software

RSX-11M application programs transfer data to/from PICTURE SYSTEM 2 by a non-conventional RSX-11M Device Driver written by E&S for PICTURE SYSTEM 2. Before graphics programs may successfully be run on an RSX-11M system, this Device Driver must be installed into the RSX-11M executive using the System Generation Process. Once this driver has been installed into the executive and the new system booted as the new RSX-11M system, the graphics software is available for use from the Graphics Library (PICLIB).