X-650-73-286

NASA TM X- 70473

SMALL INTERACTIVE IMAGE PROCESSING SYSTEM (SMIPS)

SYSTEM DESCRIPTION

JOHANNES G. MOIK

(NASA-TM-X-70473) SMALL INTERACTIVE IMAGE PROCESSING SYSTEM (SMIPS) SYSTEM DESCRIPTION (NASA) 48 p HC

N73-32085

CSCL 09B G3/08

Unclas 18196

SEPTEMBER 1973

GSFC

GODDARD SPACE FLIGHT CENTER

GREENBELT, MARYLAND

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

X-650-73-286

SMALL INTERACTIVE IMAGE PROCESSING SYSTEM (SMIPS)

SYSTEM DESCRIPTION

Johannes G. Moik*
Code 650.1, Goddard Space Flight Center
Greenbelt, Maryland 20771

September, 1973

*National Academy of Sciences
Research Associate
NASA-Goddard Space Flight Center
On Leave of absence from Institut fuer Angewandte
Mathematik und Informationsverarbeitung
Technische Hochschule Graz, Austria

SMALL INTERACTIVE IMAGE PROCESSING SYSTEM (SMIPS)

SYSTEM DESCRIPTION

Johannes G. Moik
Code 650.1, Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

The SMIP system description gives detail of the executive portion of the Small Interactive Image Processing System (SMIPS). The system operates under control of the IBM-OS/MVT operating system and uses an IBM-2250 model 1 display unit as interactive graphic device. The input language in the form of character strings or attentions from keys and light pen is interpreted and causes processing of built-in image processing functions as well as execution of a variable number of application programs kept on a private disk file. Major design goals were minimal impact on overall computer system performance, convenient communication between user and system, modularity and fast input/output routines.

This document contains a description of design considerations and summarizes characteristics, structure and logic flow of SMIPS. It discusses also data management and graphic programming techniques used for the interactive manipulation and display of digital pictures.

Preceding page blank

CONTENTS

Preceding page blank

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

SMALL INTERACTIVE IMAGE PROCESSING SYSTEM

SYSTEM DESCRIPTION

1. INTRODUCTION

Digital image processing involves the pictoral or numerical display of raw image data, the restoration or enhancement of images, the display of results as maps or photographs and the detection of objects. This requires examination of the image data from many different viewpoints.

The Small Interactive Image Processing System (SMIPS) is a tool with which the human experimenter can interact with the computer system. It is intended to gain experience in interactive image processing and to provide better communication between user and computer. The system has been developed to give the user flexible and convenient control of a variety of image processing methods. It provides a dialogue environment in which image processing functions are immediately executed and an input mode where requests are queued for later execution. Upon completion of processing steps the experimenter may investigate the intermediate results and use the information obtained for the specification of further analysis steps. SMIPS serves for a number of purposes:

1. Fast display of parts of pictorial data on the screen of a display device either numerically or as a character representation. Sampling rate and quantization levels can be changed by the user.

2. Computation and display of histograms.

3. Convenient specification of a variety of image processing tasks for restoration, enhancement and detection. The inclusion of classification procedures is planned.

4. Output of numerical results and pictures as graphs, maps and photographs.

In a batch processing mode it is necessary to specify the entire analysis to be performed prior to running the computer. This is inconvenient and may stretch the period of performing a careful analysis over a long time. Often the experimenter is not familiar with the computer system and the special analysis programs available which further complicates a successful analysis.

An interactive system differs from a conventional one in its interface with the user. By taking full advantage of the man-machine combination the efficiency

of a picture analysis can be increased considerably. By means of a display device which allows also to control the computer, the user specifies his input and the system displays the results on the screen. Thus the short time between specification of a problem and the return of intermediate results makes possible a more intelligent choice and sequencing of the analysis steps that are applied to the pictures.

This document describes the structure of the SMIP system, the system modules and its various routines and the conventions for writing application programs. It should enable a programmer to maintain, expand or modify the system. A users manual for SMIPS is available [1].

2. DESIGN CONSIDERATIONS

The design of an interactive image processing system is greatly influenced by data management problems because the large matrices representing digital pictures cannot be contained entirely in core storage. An image processing system needs, therefore, external mass storage, usually magnetic disk storage.

The raster-scan operation of image digitizers and scanning instruments on spacecrafts imposes a sequential-row access structure on the sampled image data. Therefore, the fundamental unit of the data structure is one row of the image matrix and a row is stored as one physical record on a sequential-access storage medium, magnetic tape. Magnetic tapes are used as permanent storage for large quantities of image data. It is natural to use the same data structure in an image processing system and consequently to store images as sequential row access files on magnetic disk during the execution of image processing programs.

Most of the image processing operations are performed on lines or columns of a picture and random access is not required. Each access retrieves one row of the image matrix which is processed and placed into another sequential file. This structure, however, makes operations on columns difficult, e.g. the column operations in a fast Fourier transform.

Storage of pictures on disk and accessing individual rows results in an I/O-bound program. The elapsed time to transfer a picture line to core storage is usually much longer than the time required for numerical computation on the data. The situation can be improved by blocking, i.e. grouping of several logical records into a physical. Thus, several logical records are transferred with one access into core storage. This requires, however, a larger area in core storage. In

addition the time lost in waiting for I/O operations can be reduced by a good I/O buffering system which attempts to supply new records as fast as old ones are processed. The extent to which these I/O operations exceed the actual time required for numerical computation determines how much an image processing program becomes I/O-bound. Generally, the increase of main storage areas for blocking and buffering will reduce the I/O-boundedness of a program.

The optimal design decision is dependent on the computer system used to operate the image processing system. In a multiprogramming operating system an I/O-bound program is no liability. In a multiprogramming system several different programs reside in main memory simultaneously with only one program executing at any given time. When the program in execution requests an I/O operation, its execution is suspended and another program begins execution. Concurrently a channel (peripheral processor) is assigned to perform the I/O operation requested by the suspended program. Thus, an I/O-bound program does not waste any of the total computing time of the central processor. As long as the image processing system does not overburden the main memory of the computer, there is room for other programs to share the memory. Thus, the chief impact of the multiprogramming environment on the design of an image processing system is to minimize the utilization of central processor and main memory.

Minimal main memory utilization is even more important in the design of an interactive image processing system. In such a system short computations are followed by relatively long periods of waiting for input from the user. These periods of thinking and decision making may last several minutes. An adequate environment for an interactive image processing system is a timesharing system with paging, where parts of the program are transferred to external storage during wait periods in order to free main storage for other users. In such a system the design goal is to keep the page traffic between main storage and backing storage to a minimum.

In a multiprogramming system with no dynamic allocation and deallocation of resources (main storage, tape units, disk storage) like IBM-OS/360, all resources needed by the interactive image processing system remain allocated from start to end of operation and are not available to other users.

The SMIP system operates under OS-MVT, its major design goals are, therefore, minimal utilization of main storage and tape units at the expense of disk space on scratch disk units and some overhead for unloading tapes if several tapes are required in a sequence.

If a multiprogramming system is not available the design goal for an image processing system is to make it as little I/O-bound as possible. These efforts include overlapping I/O operations and computation by buffering, optimal blocking

and disk separation if separate disks on independent channels are available. An interactive image processing system can probably not be implemented in such an environment if many other people request the use of the computer system.

3. FUNCTIONAL CHARACTERISTICS

The SMIP system is designed for digital processing and recording of pictoral data in an interactive mode. Objectives of the system are convenient communication with the computer by users who are not expert programmers, fast response to requests for processing of pictures, complete error recovery as well as simplification of future programming efforts for extension of the system.

The SMIP system is intended for operation on an IBM 360 computer equipped with an IBM-2250 Model 1 display unit. The current implementation runs on either the 360/75 or the 360/91 computers at SESD of GSFC under OS/MVT-Release 20.6. The system needs a core region of 200K bytes and uses one 9-track and one 7-track tape unit and space on five scratch disk units. The 2250 display unit is used as display device, its alphameric keyboard, function keys and light pen provide for convenient communication between user and computer.

To allow the use of already existing image processing programs and to avoid considerable reprogramming effort, the SMIP system was made fully compatible with the VICAR system designed at the Jet Propulsion Laboratory. A SMIPS user can execute any image processing program contained in the VICAR Library on the 360/75.

Operation of the SMIP system is similar to the use of a desk calculator. It is a repeated sequence of single requests followed by responses. The user requests a computation by typing a command, pressing a function key or using the light pen. The system responds with the display of a message or picture on the screen of the graphic terminal or by indicating the completion of the request with an audible alarm signal. Thus, SMIPS is an interpretive system.

Three modes of operation can be distinguished. In the dialogue mode each command issued by the user is interpreted and the appropriate processor is called for execution of the command. In the input mode each command is interpreted as a request for an image processing task, the corresponding program should exist in the systems library. These requests are inserted into the task queue TFILE. Here lies the extendability of the system without increasing the core size, new image processing programs are added to the library and can then be specified as tasks in the input mode. The functions for these two modes are performed by the system module VICINT which handles the communication between user and

computer and the input/output operations and by the system module INTRPROC which contains the command interpreter and the processors associated with each command.

The image processing tasks queued in the task queue are processed in the task processing mode. Each task involves the execution of a program which must exist in the system program library. Execution is started by deleting INTRPROC from core. VICINT loads in its place a transient routine TASKPROC which reads the task queue and initiates the first task. This task replaces TASKPROC in core. Upon completion, TASKPROC is reloaded and the next task in initiated. When the last task has been completed, control is returned to VICINT which reloads INTRPRET and waits for further input from the user.

The operation of SMIPS is controlled by a simple command language. There are two groups of commands available. Commands of the first group involve the manipulation of pictures, like creation and release of picture names, transfer of pictures between magnetic tape and disk, display of pictures on the display device and printing of pictures or histograms. Commands of this group are:

RESERVE	create a picture name
FREE	release a picture name
READ	transfer a picture from tape to disk
WRITE	transfer a picture from disk to tape
DISPLAY	display a picture or its histogram
THRESHOLD	segment a picture
PRINT (key 3)	print the picture displayed on screen
PRINT (key 20)	print the histogram displayed on screen
MOVE UP (key 12)	shift the histogram displayed on the screen
MOVE DOWN (key 24)	
SHRINK (key 17)	

The second group of commands involves control of system operation and changing of system parameters and operational environments. These commands are:

SET	set various system parameters
EXECUTE	execute the tasks in the task queue
DISCONNECT	terminate after processing the task queue

EXIT	terminate immediately
TASKS	display tasks in task queue
TIME (key 4)	display remaining time
NAMES (key 9)	display currently used picture names
INPUT (key 10)	enter input mode
PARAM (key 16)	display system parameters
GUIDE (key 22)	display users guide
EXIT (key 31)	terminate immediately

Errors in the commands are detected, an explanatory message is displayed on the screen and the system waits for specification of the correct command. This error recovery feature makes the use of an interactive system very convenient for there is little delay due to a mistake. A detailed description of the commands is given in [1].

4. STRUCTURE OF THE SMIP SYSTEM

The SMIP system disposes of three operating environments, the dialogue environment, the input environment and the task processing environment. In this chapter the three environments as well as the structure of the program modules involved in their operation will be described in some detail.

4.1 STATE DIAGRAM OF SMIPS

The structure of the SMIP system can be described using a finite state machine model. In the dialogue and input environment the system accepts input (control information) from the user. These two environments provide for the interaction between user and the system. In the task processing environment no interaction is possible while the system processes previously specified tasks. The user has to wait for completion of processing which is indicated by display of the cursor and a single stroke audible alarm to attract the users attention.

The transfer from one environment to another is accomplished with commands or function keys. Fig. 1 represents the state diagram of the SMIP system. It shows the commands and events which leave the system in the same environment, as well as the conditions for transfer from one environment to another. A detailed description of the commands is contained in the SMIPS Users Manual [1].

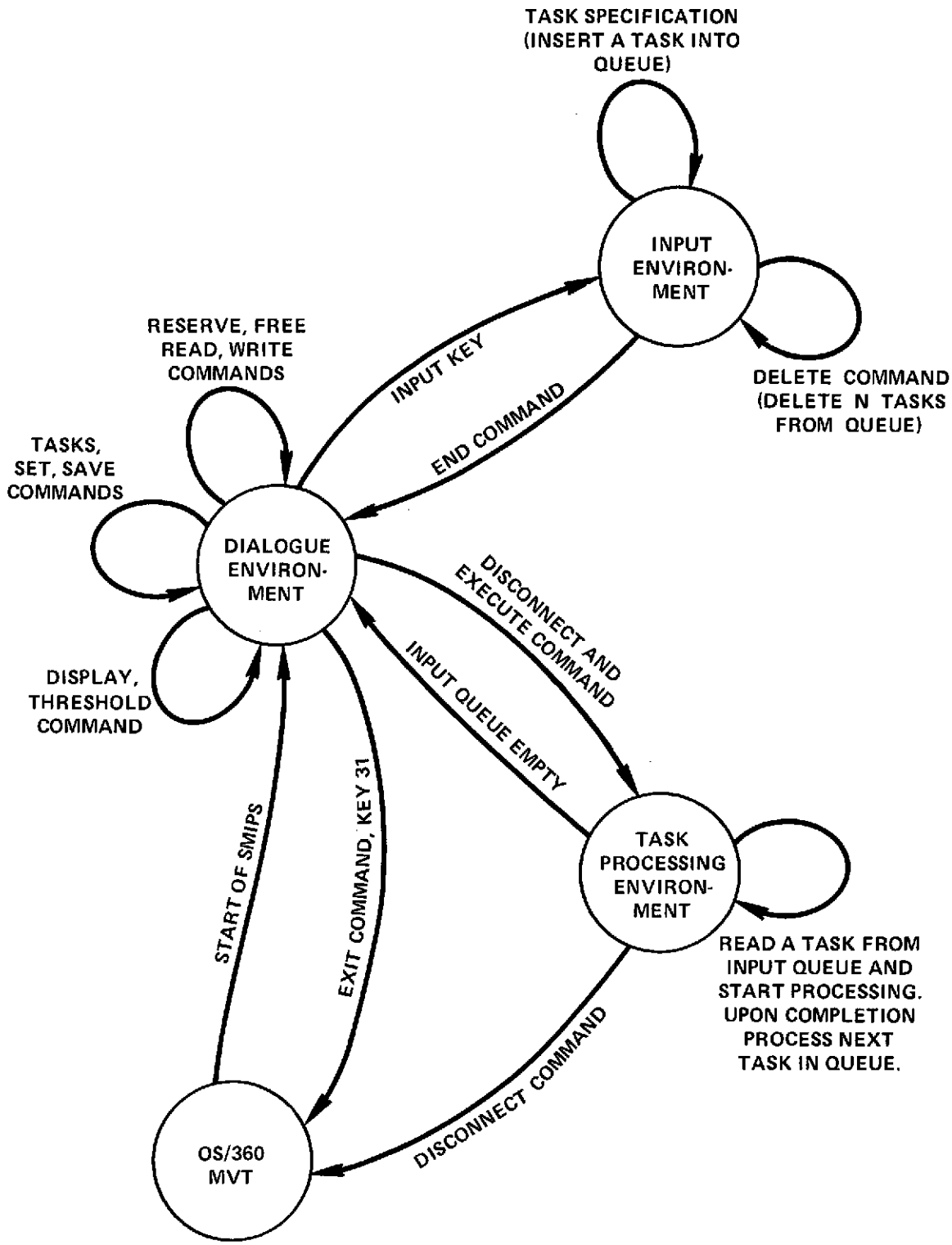


Figure 1. State Diagram of SMIPS

4.2 DIALOGUE ENVIRONMENT

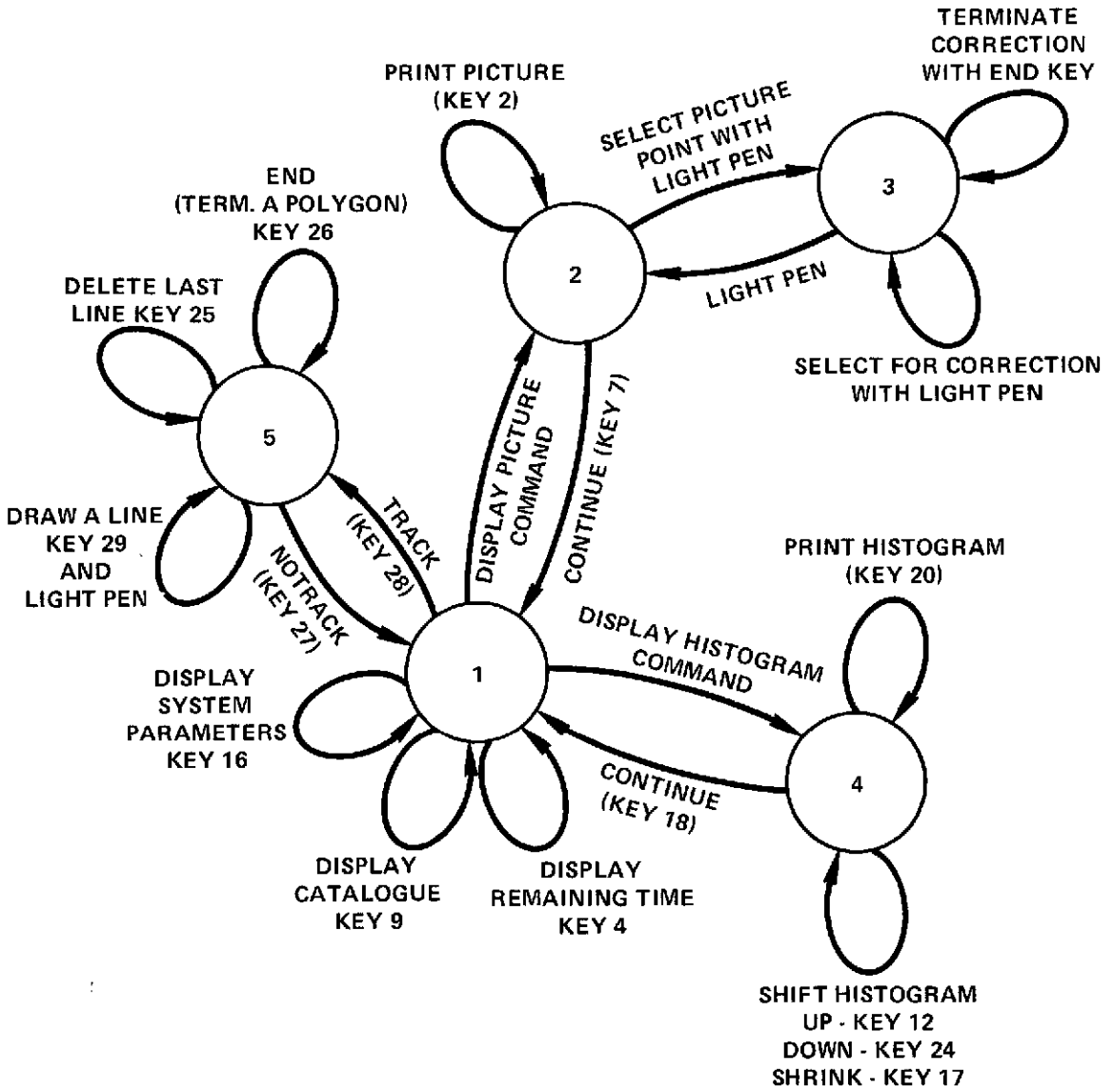
The basic set of image processing functions is available in the dialogue environment. Each request is immediately executed. The commands available can be used to perform allocation and deallocation of disk space for pictures (RESERVE, FREE), transfer of pictures between disk and tape (READ, WRITE), display of pictures, histograms and label information (DISPLAY) and change of various system parameters (SET). Utility functions are provided to inform the user on the current state of the system, i.e. remaining processing time (KEY 4), tasks in input queue (TASKS), current value of system parameters (KEY 16) and current status of the catalogue of existing pictures in the system (KEY 9). Control functions permit display of a brief users guide (KEY 22), execution of the image processing programs in the task queue (EXECUTE), termination with previous processing of the programs in the task queue (DISCONNECT) and immediate termination (EXIT or KEY 31).

In the dialogue mode two SMIP system modules VICINT and INTRPRET are in main memory. VICINT is loaded into main memory when the system is started and is the only permanent core resident module of SMIPS. Its main functions are to enable the communication between user and computer, to provide the interface to the data management functions of OS/MVT and to load and delete the modules INTRPRET and TASKPROC as well as the image processing programs into main memory. The state diagram for the dialogue environment is shown in Fig. 2.

4.2.1 Supervisor Module VICINT

The supervisor module VICINT is the core resident part of SMIPS. It contains the interactive handler INTPROC, the Input/Output handler and some utility routines. INTPROC allocates the IBM-2250 display device, assigns buffer storage to it, establishes the routines to handle attentions from light pen, function keys and end-of-block key and allocates a work area where display files are assembled for efficient transfer between main memory and the 2250-buffer. A brief guide to the usage of the SMIP system is displayed on the screen and INTPROC waits for a continuation signal. Upon receiving this signal from key 0 the permanently displayed figures (headline, frame) and the cursor for input from alphameric keyboard are displayed and the module INTRPRET is loaded into main memory.

This module contains the scanner, interpreter and the routines to construct the display files for the display of pictures, histograms and label information, the routines for transfer of pictures between tape and disk, routines for light pen tracking and for processing the task specification statements. After loading, the transfer vector containing the addresses of the core resident routines referenced in INTRPRET is built and the system enters a wait state awaiting attentions from the user.



STATES:

1 DIALOGUE STATE (WAIT FOR ANY COMMAND VALID IN THE DIALOGUE ENVIRONMENT)

2 PICTURE DISPLAY STATE (WAIT FOR NUMERICAL DISPLAY, PRINT OR CONTINUE COMMAND)

3 NUMERICAL PICTURE DISPLAY STATE (WAIT FOR PICTURE ELEMENT CORRECTION OR RETURN TO STATE 2)

4 HISTOGRAM DISPLAY STATE (WAIT FOR PRINT, SHIFT OR CONTINUE COMMAND)

5 LIGHT PEN TRACK STATE (WAIT FOR DRAW A LINE, DELETE A LINE, TERMINATE A POLYGON OR NOTRACK COMMAND)

Figure 2. State Diagram of Dialogue Environment

When an attention occurs it is interpreted by INTPROC and the appropriate action is taken. Attention from an enabled function key causes transfer of control to the routine associated with this key.

Upon an attention from the END key the character string up to the cursor in the input figure is transferred to main memory, the length of the string is determined and control is passed to entry point SCAN in the module INTRPRET. SCAN checks the command for syntactic correctness, interprets the command and branches to the corresponding routine or returns immediately with an appropriate code.

The following return codes and the associated actions are possible:

RC = 0 wait for attentions from the user

RC = 1 process the image processing tasks waiting in the task queue. The module INTRPRET is deleted from memory to free core for linking the module TASKPROC. This module reads one task specification from the task queue, constructs the load sequence to link the appropriate image processing program and performs label and parameter processing for this program. This sequence is repeated until the task queue is empty and control returns to INTPROC which reloads the module INTRPRET and waits for user attentions. RC = 1 is caused by the EXECUTE command.

RC = 2 disconnect the 2250 display device, close all open data sets, free buffers and work areas and return to OS. This is caused by the EXIT command.

RC = 3 disconnect the 2250 display device, free the work area and then perform the actions RC = 1. After that close all open data sets and return to OS. This sequence is caused by the DISCONNECT command.

The flowchart of INTPROC is given in Fig. 3, Fig. 4 shows the JCL statements required to link VICINT.

4.2.2 Dialogue Processor INTRPRET

The function of the dialogue processor INTRPRET is to process the commands issued by the user. It is loaded automatically into core upon start of SMIPS and each time after completion of processing of the tasks in the input queue. The main parts of INTRPRET are the scanner which checks the command string entered by the user at the alphameric keyboard for illegal characters, the interpreter which interprets the correct commands and the different command processors for processing the commands. Upon completion control is returned to

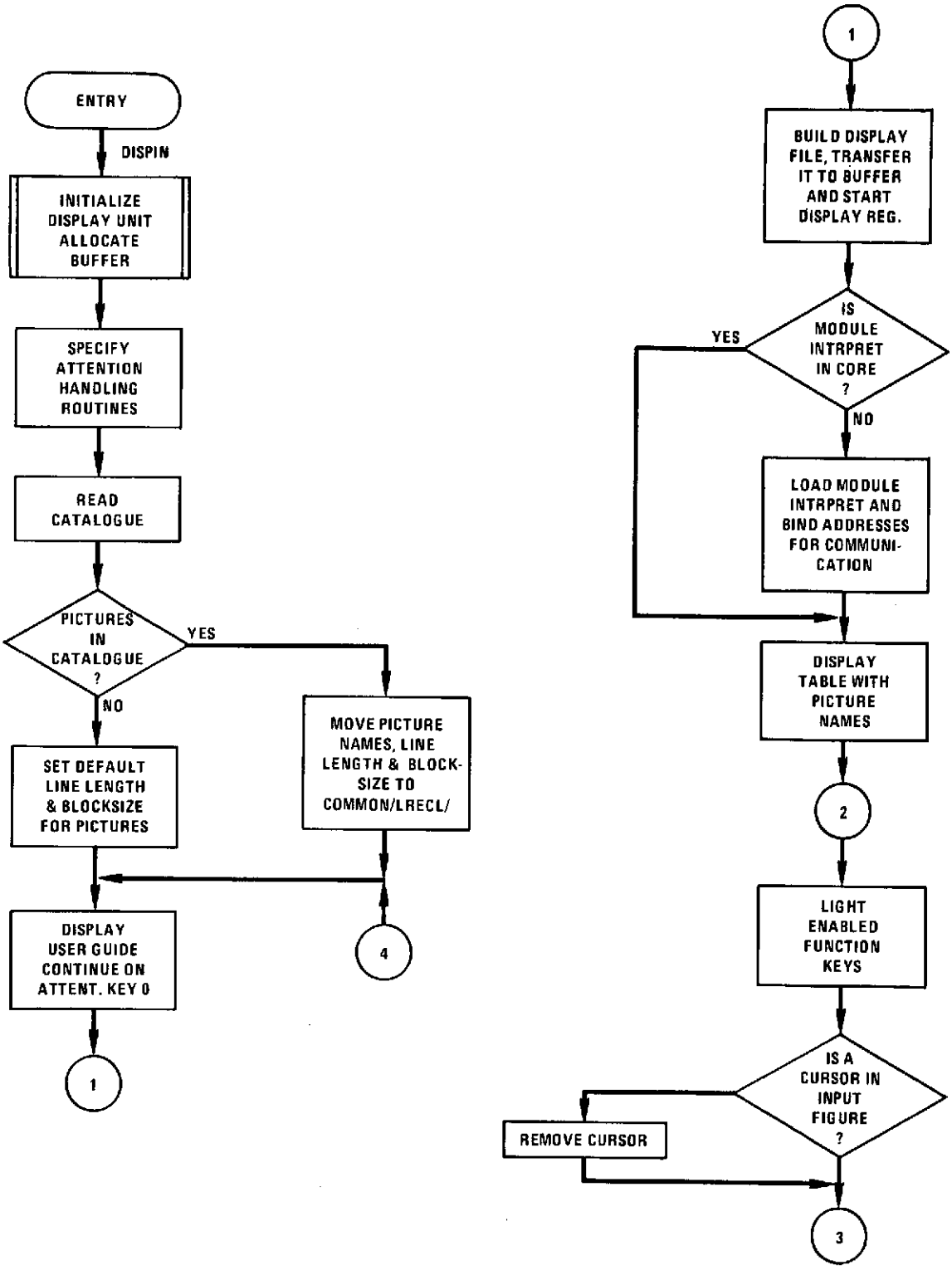


Figure 3. INTPROC Flowchart (1 of 3)

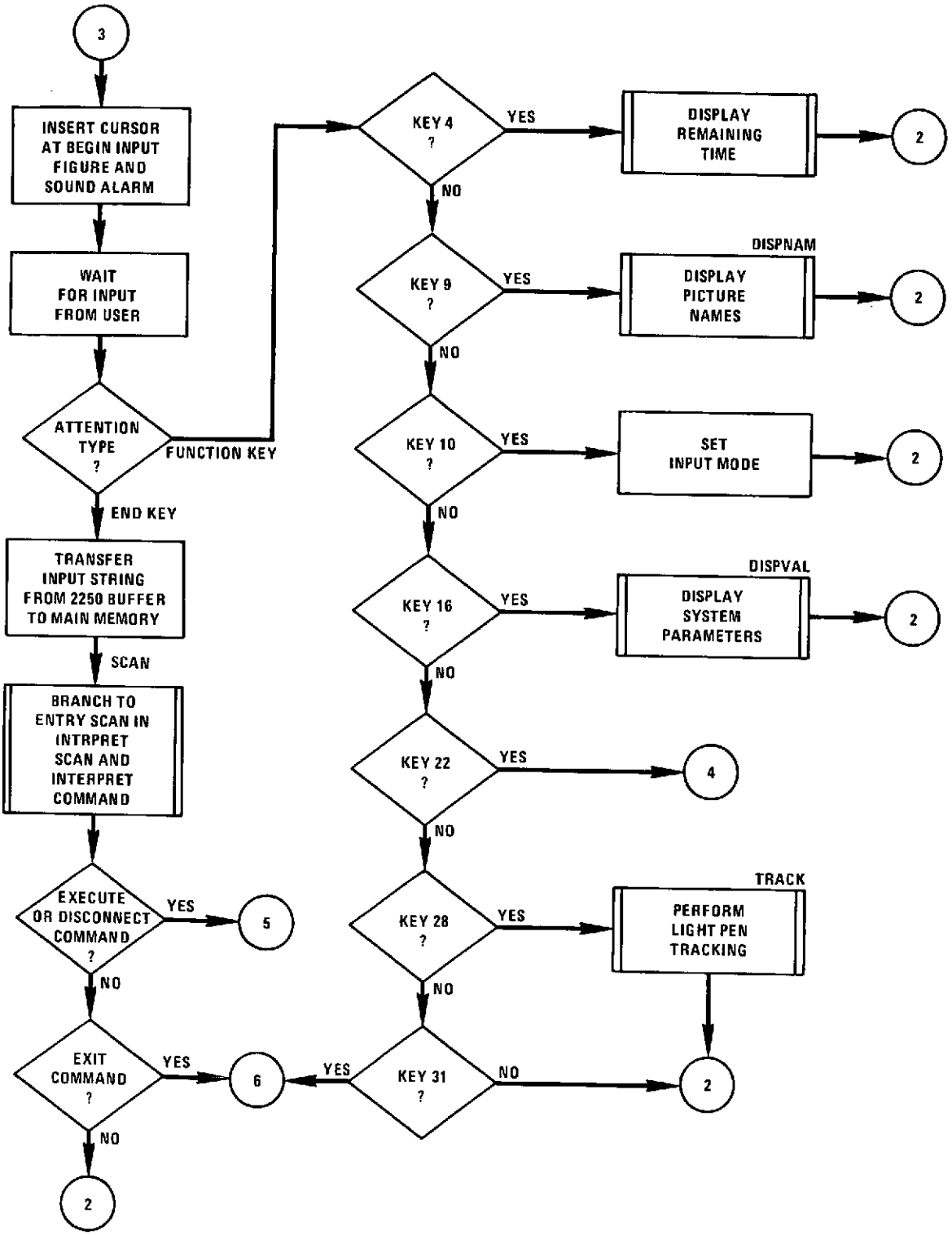


Figure 3. INTPROC Flowchart (2 of 3)

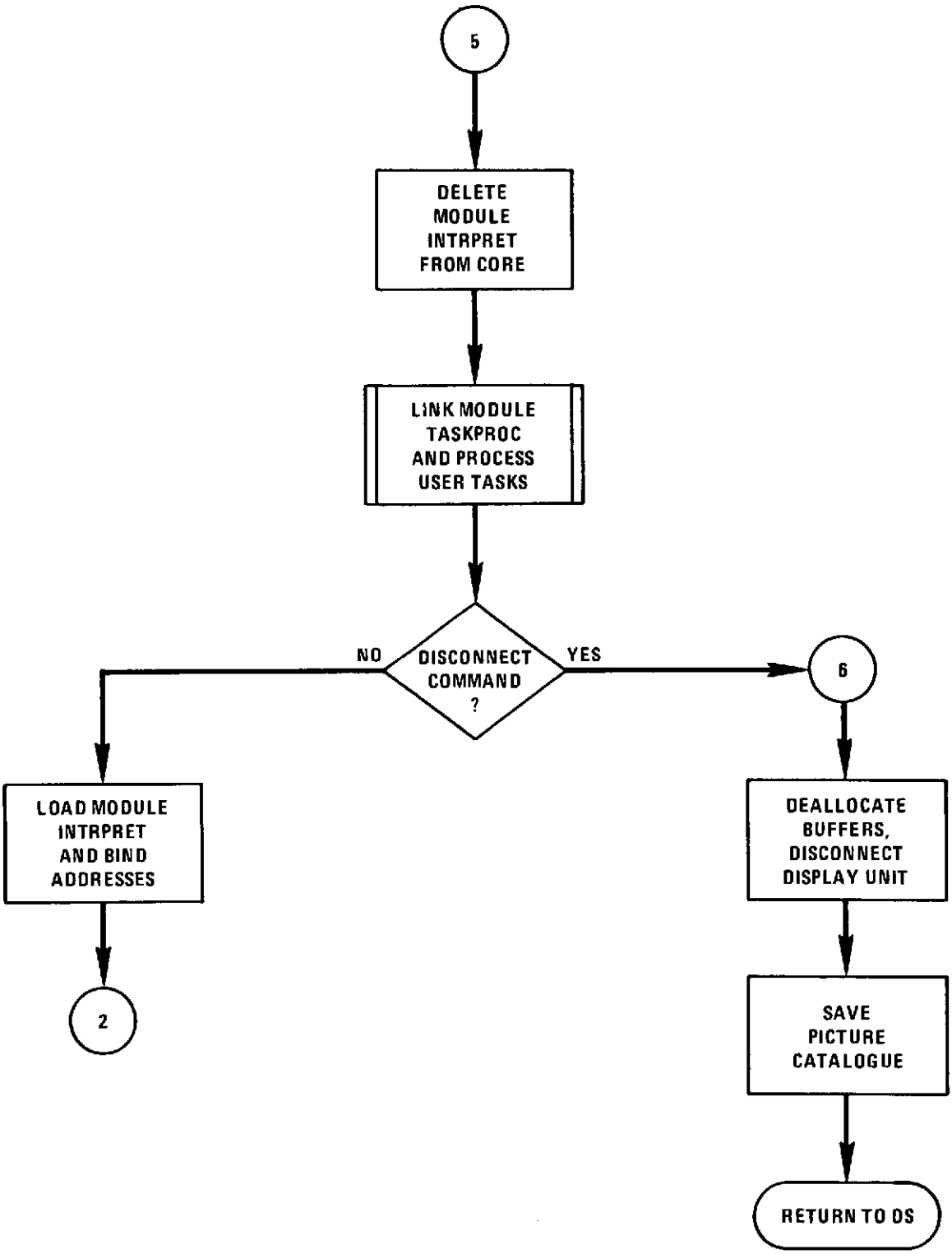


Figure 3. INTPROC Flowchart (3 of 3)

```
// EXEC LINK,PARM='XREF,LIST,MAP'
//SYSLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//        DD DSN=SYS2.LINKLIB,DISP=SHR
//        DD DSN=K3.T1DAK.T1005.LVICARSB,DISP=SHR
//        DD DSN=SYS1.FORTLIB,DISP=SHR
//        DD DSN=SYS2.FORTLIB,DISP=SHR
//SYSLMOD DD  DSN=K3.SOJGM.S1011.VICINT,DISP=OLD
//SYSLIN DD *
 INCLUDE LIBRY(INTPROC)
 INCLUDE LIBRY(DISPIN)
 INCLUDE LIBRY(VBLOCK)
 INCLUDE LIBRY(SYSMSG)
 INCLUDE LIBRY(DISPNAM)
 INCLUDE LIBRY(DISPVAL)
 ENTRY INTPROC
 NAME VICINT(R)
//LIBRY DD DSN=K3.SOJGM.S0002.PAX11,DISP=SHR
%
```

Figure 4. JCL for VICINT

VICINT with the appropriate return code. The flowchart for INTRPRET is shown in Fig. 5.

Each command has a processor associated with it to perform processing of the request. The flowchart for the READ command processor is shown in Fig. 6. The flowchart of the routine TAPOS in the READ command processor is given in Fig. 7. Fig. 8 shows the flowchart for the DISPLAY command processor.

4.3 INPUT ENVIRONMENT

Image processing with the functions provided by the module INTRPROC is very restricted. To allow more advanced operation and the expansion of the system within the limited amount of main memory available, the program module INTRPROC has to be replaced by a particular image processing program. Upon completion of this task INTRPROC is reloaded into main memory, its previous state is restored and the system waits for further interaction with the user.

In many cases a sequence of such image processing functions can be performed without a required intermediate action by the user. To avoid the overhead of deleting and reloading the dialogue processor the input mode is provided by the system. In the input environment each input from the alphameric keyboard is interpreted as a task specification. Each such image processing request is inserted into a queue of tasks for later processing. All the image processing programs in the SMIP- and VICAR system library can be used in this mode.

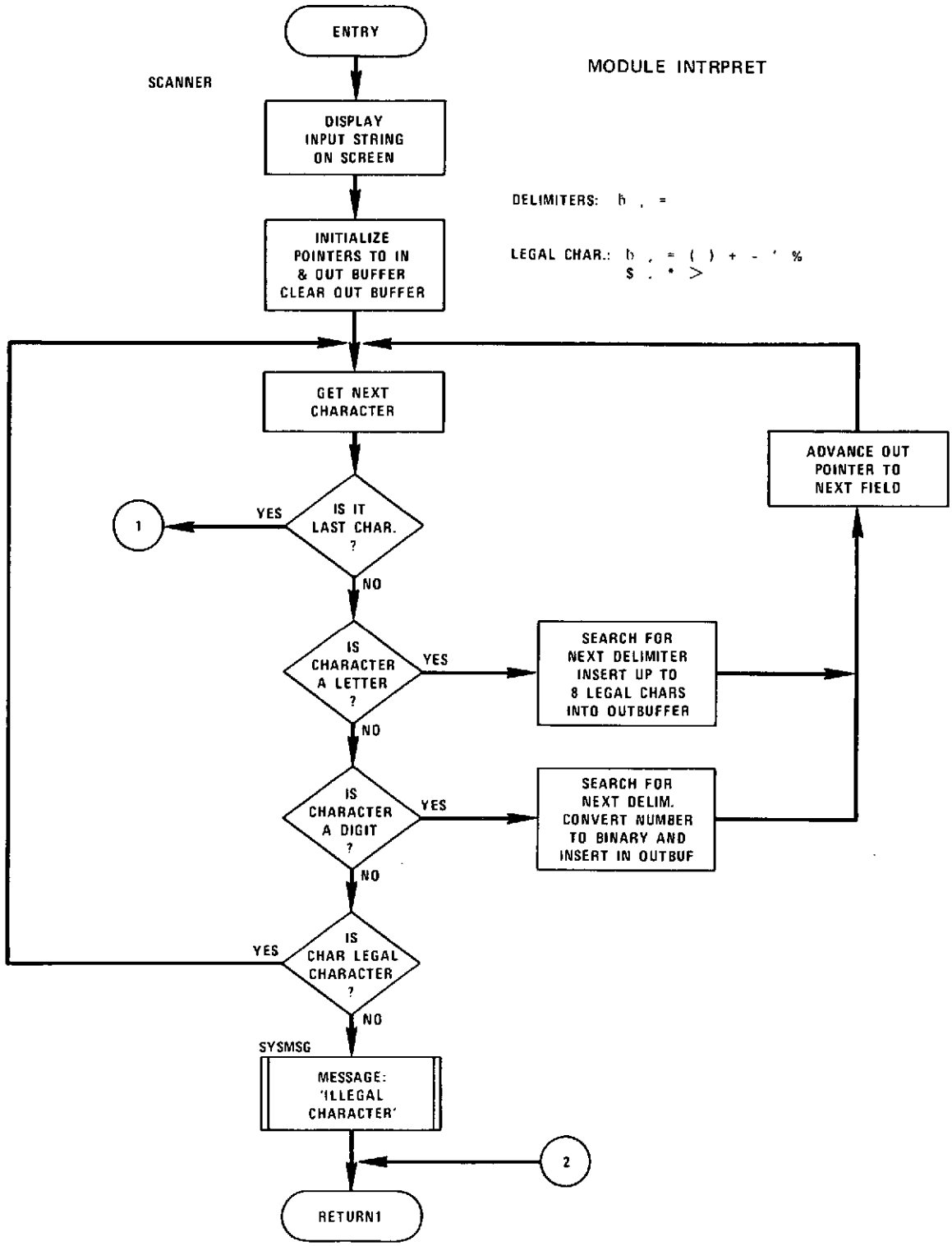


Figure 5. INTRPRET Flowchart (1 of 3)

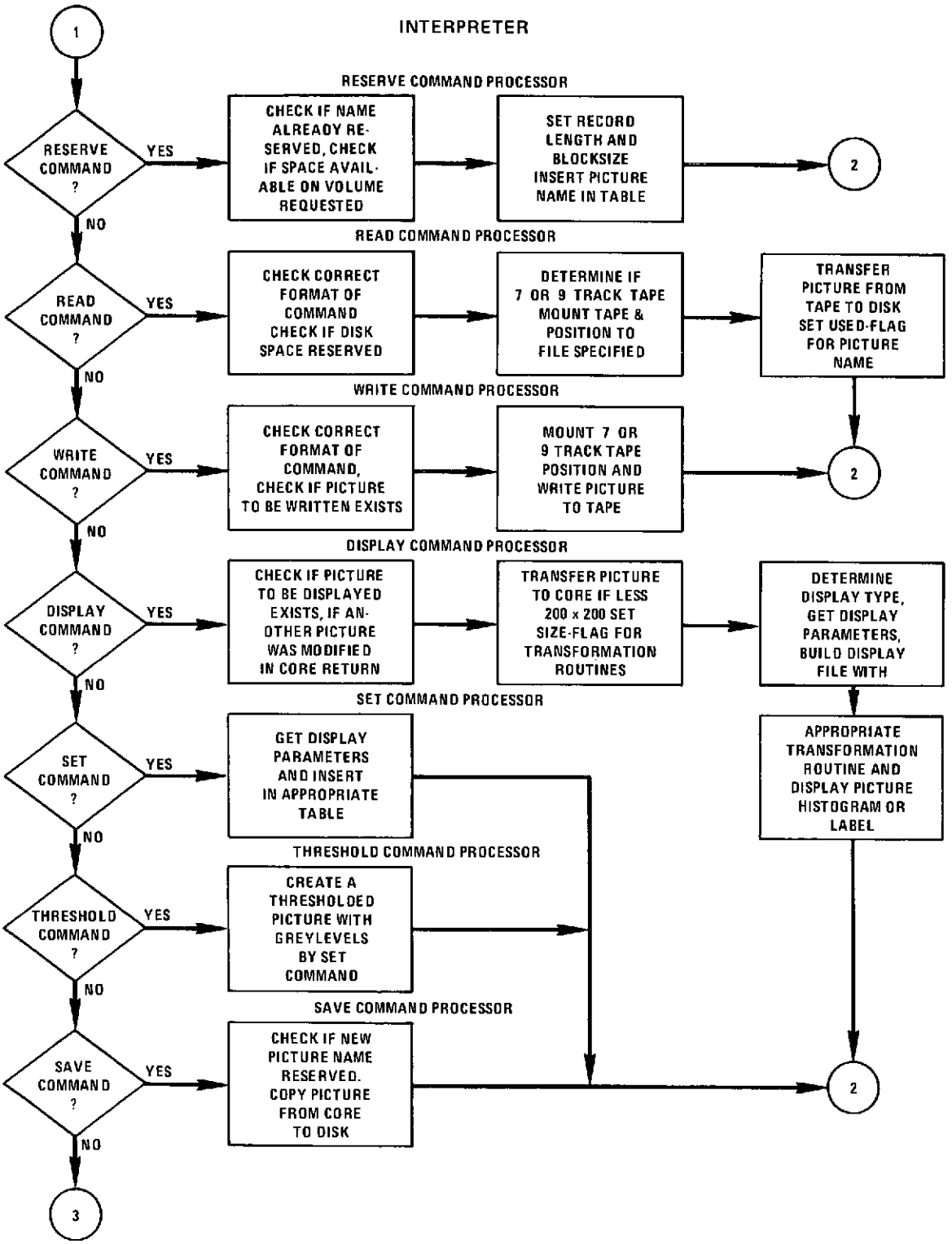


Figure 5. INTRPRET Flowchart (2 of 3)

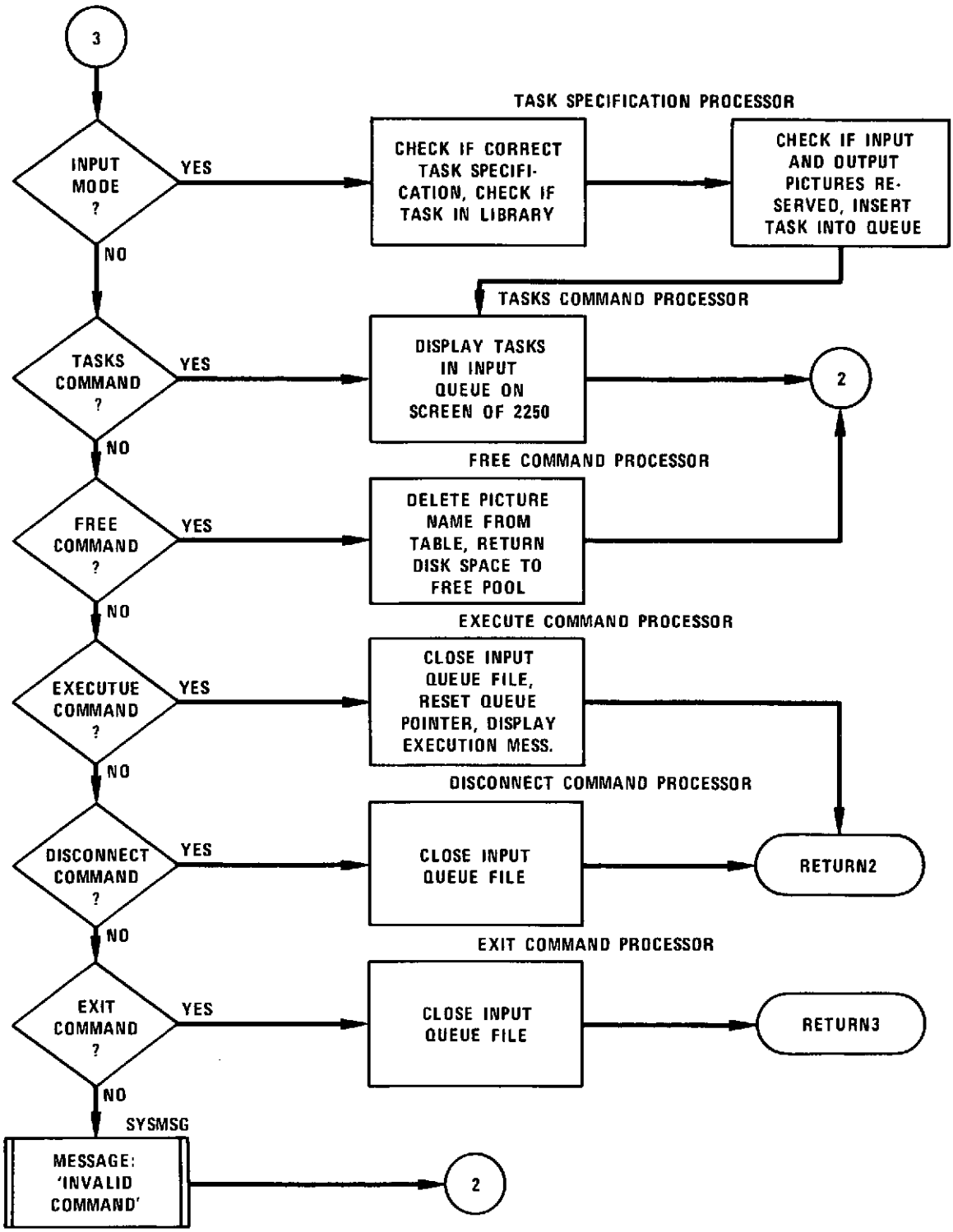


Figure 5. INTRPRET Flowchart (3 of 3)

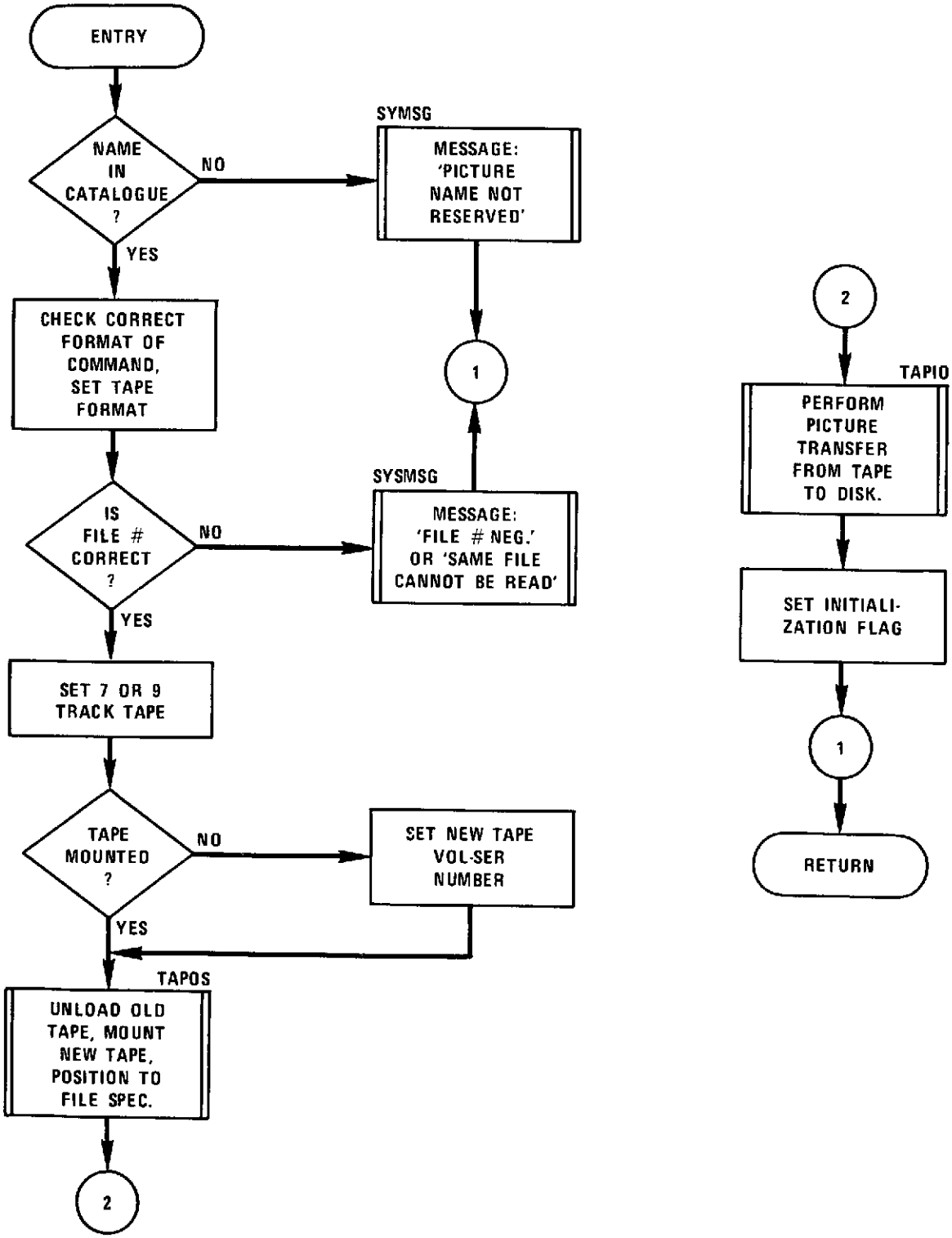


Figure 6. READ Command Processor Flowchart

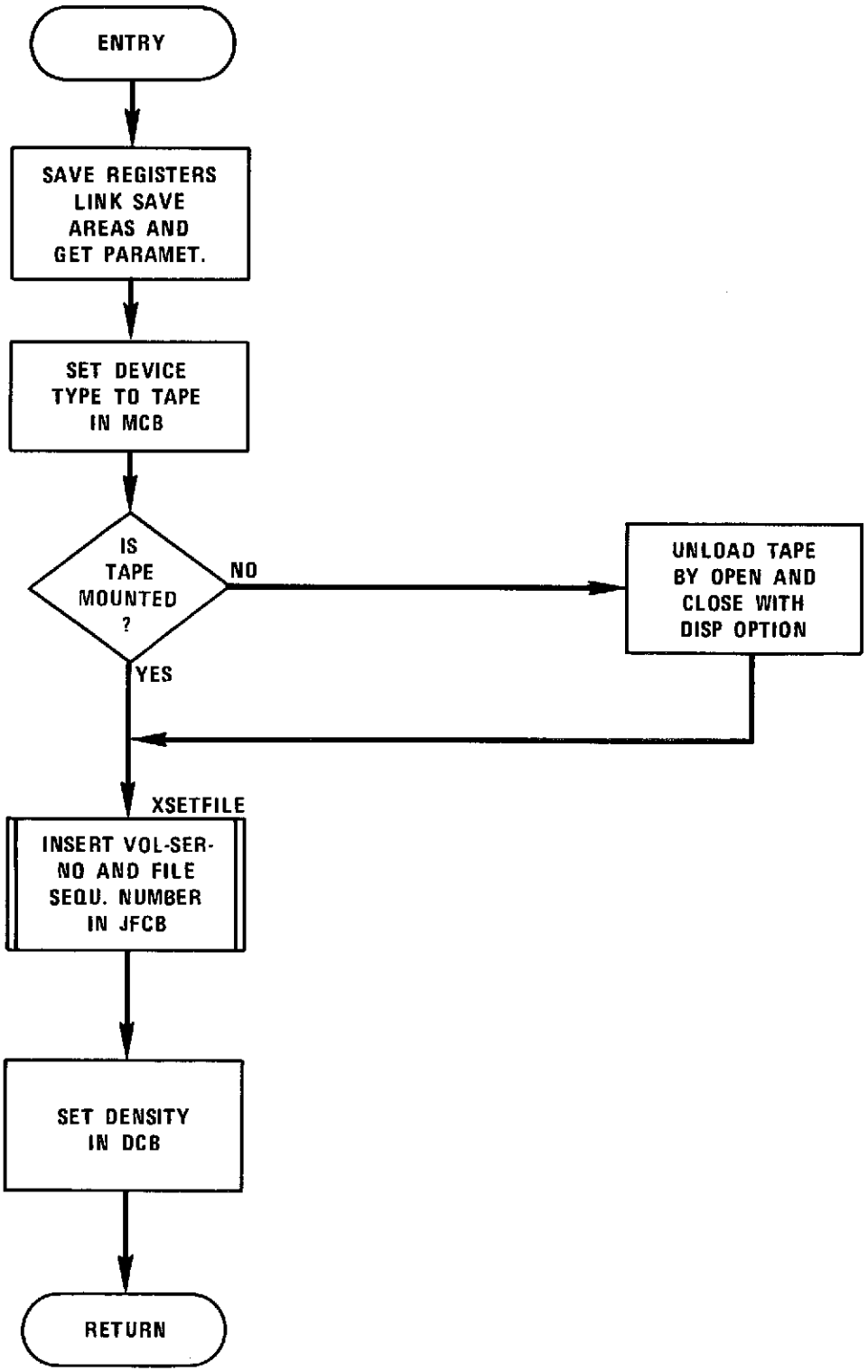


Figure 7. TAPOS Flowchart

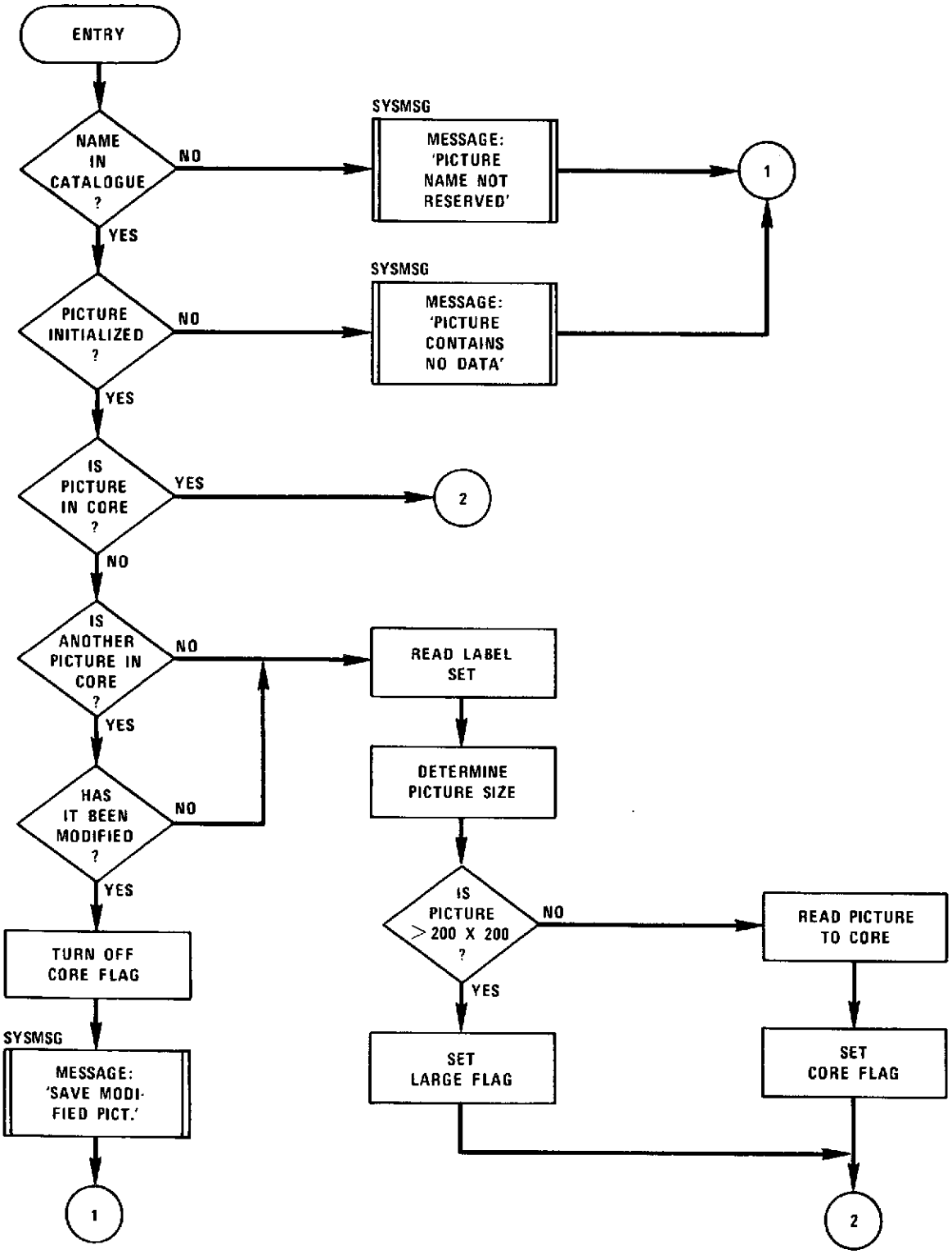


Figure 8. DISPLAY Command Processor Flowchart (1 of 2)

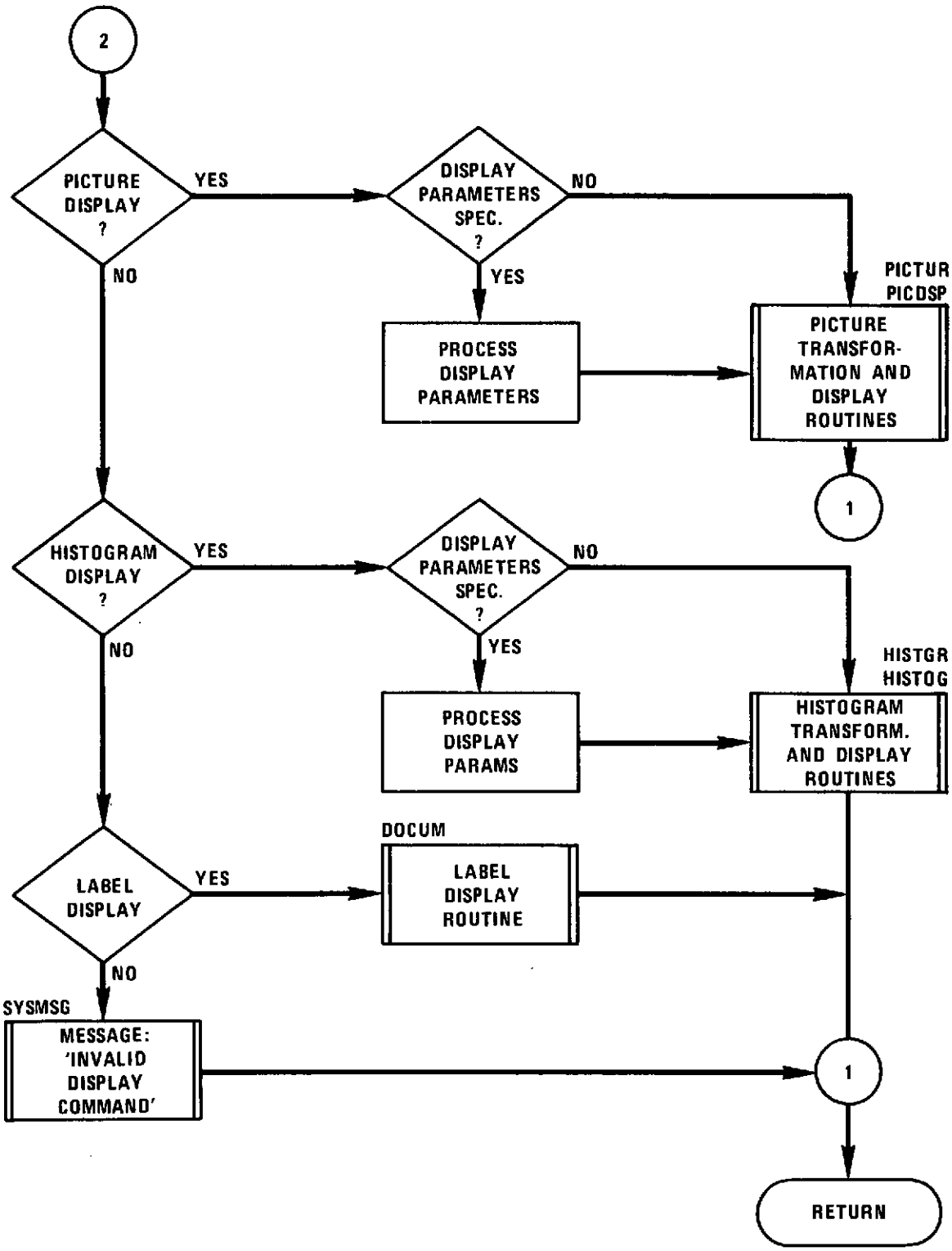


Figure 8. DISPLAY Command Processor Flowchart (2 of 2)

Checking the syntactic correctness of the command as well as the availability of the specified program is done in this mode. Only correct requests for available programs are inserted into the task queue. If the program is not in the library, the user will be notified with the message 'taskname NOT IN LIBRARY'. These operations are performed by the routine TASKQ in INTRPROC. The command END returns control to the dialogue environment. Fig. 9 shows the JCL statements for linking all routines of the module INTRPRET.

```
// EXEC LINK,PARM='XREF,LIST,MAP,LET'
//SYSLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//        DD DSN=SYS2.LINKLIB,DISP=SHR
//        DD DSN=K3.T1DAK.T1005.LVICARSB,DISP=SHR
//        DD DSN=SYS1.FORTLIB,DISP=SHR
//        DD DSN=SYS2.FORTLIB,DISP=SHR
//SYSLMOD DD  DSN=K3.SOJGM.S1011.VICINT,DISP=OLD
//SYSLIN DD *
 INCLUDE SYSLIB(FAKIBCOM)
 INCLUDE LIBRY(LINK)
 INCLUDE LIBRY(SCAN)
 INCLUDE LIBRY(DISPQ)
 INCLUDE LIBRY(TRACK)
 INCLUDE LIBRY(NUMRIC)
 INCLUDE LIBRY(NUMDSP)
 INCLUDE LIBRY(PICTUR)
 INCLUDE LIBRY(PICDSP)
 INCLUDE LIBRY(THRESH)
 INCLUDE LIBRY(PICPRT)
 INCLUDE LIBRY(PUTMAIL)
 INCLUDE LIBRY(DOCUM)
 INCLUDE LIBRY(HISTGR)
 INCLUDE LIBRY(HISTOG)
 INCLUDE LIBRY(PRHIST)
 INCLUDE LIBRY(PRTITL)
 INCLUDE LIBRY(PRCHK)
 INCLUDE LIBRY(TAPOSIT)
 INCLUDE LIBRY(TAPIO)
 INCLUDE LIBRY(CHCKRT)
 INCLUDE LIBRY(TASKQ)
 ENTRY INTRPRET
 NAME INTRPRET(R)
//LIBRY DD DSN=K3.SOJGM.S0002.PAXII,DISP=SHR
%
```

Figure 9. JCL for INTRPRET

NOT REPRODUCIBLE

4.4 TASK PROCESSING ENVIRONMENT

The tasks waiting in the input queue are processed in the task processing environment. Processing is started with the EXECUTE command. The module INTRPRET is deleted from core and the module TASKPROC is loaded and receives control. TASKPROC is a modified version of the module VMJCPHAS in the VICAR system. It reads a task from the input queue and processes its parameters and labels. Parameters for a task are submitted in free-form format and may be in one of several types (Integer, Real, Alphameric, Hexadecimal, Literal). The parameters are stored in the task queue in their original EBCDIC form. TASKPROC translates all parameters to an internal computer representation which the processing programs can utilize. The translated parameters are written temporarily into the disk data set VSYS00. Image processing programs obtain these parameters by calling the routine PARAM.

Image processing programs are written using the data set reference numbers 2 to 11 for input data sets and the data set reference numbers 1 and 12 to 14 for output data sets. TASKPROC initializes the MCB's in VICINT and establishes thereby the linkage between these data set reference numbers and a specific data set or device. It builds also a table with the task name which is later used to load the particular image processing program into the main memory. Then the image processing program is fetched from the library, replaces TASKPROC in core and starts processing.

The processing programs, in general, expect a standard data format for all input and output data sets. This consists of a label set followed by a number of data records. Normally TASKPROC copies the label set from the primary input data set to any specified output data sets. The system label record is updated. If the programmer has specified optional user labels to be added, TASKPROC adds these to the label set on the output data sets. This automatic label processing is suppressed if the first character in the taskname is a V.

In addition, VMJC positions all input and output data sets to a point just prior to the first data record. An image processing program does, therefore, not have to skip the label records. Upon completion of a task, TASKPROC is reloaded and the next task is initiated. When the last task from the queue has been completed, the dialogue module INTRPRET is loaded into core, its previous state is restored and the system waits for further requests to process. This is indicated with an audible alarm and the display of a cursor in the lower left corner of the screen. Fig. 10 lists the JCL statements for linking TASKPROC and putting it into the library.

```
// EXEC ASMG
//SYSIN DD DSN=TASKPROC.SOJGM,UNIT=2400,LABEL=(4,SL,,IN),
//  VOL=SER=Z1610,DISP=(OLD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200
// EXEC LINK,PARM='XREF,LIST,MAP,LET'
//SYSLIB DD DSNAME=SYS1.LINKLIB,DISP=SHR
//       DD DSNAME=SYS2.LINKLIB,DISP=SHR
//       DD DSN=K3.T1DAK.T1005.LVICARSB,DISP=SHR
//       DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSLMOD DD DSN=K3.SOJGM.S1011.VICINT,DISP=OLD
//OBJECT DD *
 INCLUDE LIBRY(SYSMSG)
 ENTRY VMOD03
 NAME TASKPROC(R)
//LIBRY DD DSN=K3.SOJGM.S0002.PAXII,DISP=SHR
%
```

Figure 10. TASKPROC JCL

4.4.1 Task Characteristics

Each image processing task involves execution of a program which must exist in the SMIPS library (data set name K3.SOJGM.S1011.VICINT) or in the VICAR library. The programs may be written either in assembly language or FORTRAN IV and may include any number of subroutines. All features of these languages may be used with the exception that for input/output of pictorial data the FORTRAN I/O statements READ and WRITE and the assembly I/O macros should not be used. The application programs communicate with the data management facilities of OS/MVT through the I/O handler in VICINT. This routine supervises all the I/O operations of the application program. Up to 14 simultaneous I/O requests are supported and the processing of any direct access or tape data file may be either sequential or random.

Application programs are required to open data sets prior to use. The program may close its data sets. If not, the system will automatically close any open data sets at the end of the task. SMIPS uses the VICAR I/O routines which are described in [3]. FORTRAN application programs may be written in two forms:

1. The FORTRAN program is coded as a subroutine with no arguments. In this case no FORTRAN I/O statements are allowed and the program has to use the PRINT routine of VICAR for printing on the high speed printer. Linkage-editing of the program module should include the VICAR routine FAKIBCOM. This routine provides some service functions which are normally handled by the IBM routine IBCOM#. IBCOM# is a very complex program which handles FORTRAN input/output control, program

linking, library routine error handling and program termination at a cost of over 10 K bytes of core. FAKIBCOM provides IBCOM# initialization, library routine error handling and program exits at a cost of only 200 bytes of core.

2. The FORTRAN program uses the FORTRAN I/O statements. In this case the program has to be written as a main program and FAKIBCOM should not be included in linkage-editing. IBCOM# will be automatically linked.

4.4.2 Communication between SMIPS and Application Programs

The communication between a task program and the system is accomplished with a transfer vector. The transfer vector provides the entry points to the I/O handler and to the image display facilities provided by VICINT. The routine LINKUSER which builds the transfer vector and provides the addresses of the entry points should be included when linking the program module. This routine receives control when execution of an image processing task is started, binds the addresses in the transfer vector and passes control to the program. Application programs should exit with CALL END which will assure the continuation of task processing.

There are two sources of input to an application program, the input label and input parameters. The input label contains in word 9 the number of lines in the picture and in word 10 the number of bytes in one line. The input parameter table is placed by TASKPROC into the data set VSYS00. By a call to PARAM in the application program, the I/O handler reads the parameter table into the buffer designated by the program. The parameter table is comprised of two sections, (1) system parameters and (2) user parameters. The first 10 words of the parameter table (Fig. 11) are reserved for system parameters. The first four words contain binary integers representing the output picture dimensions. This data are taken from the size field in the task specification. The fifth and sixth words are binary integers representing the input picture dimensions. These data are taken from the input label. The seventh and eighth word contain the number of input and output data sets for this task respectively. The tenth word contains the length of the parameter table in full words.

4.4.3 Communication between Application Programs

Application programs cannot only receive the parameter table provided by the system but they may also access parameters generated by another application program. Any task is allowed to generate parameters for other tasks. The communication problem is solved by a message transfer mechanism. Messages

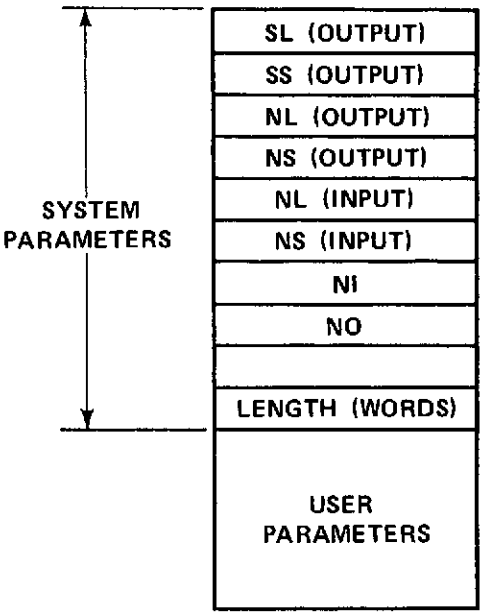


Figure 11. Parameter Table

(mail) are put into a circular list by the sending task and picked up from the list by the receiving task.

The sending task requests a mail block, puts its name and the name of the receiver into the block as well as the length of the mail and the mail itself. A call to PUTMAL inserts the mail block into the list. The receiving task which can only proceed with the information from a certain sender inquires the list by calling the routine GETMAL. If mail for the task coming from the specified sender is found, GETMAL transfers the information to a specified buffer. Otherwise, the task cannot proceed and returns control with an appropriate message. Some tasks may continue processing with default parameters. The format of the mail block is shown in Fig. 12.

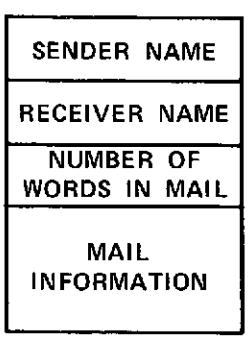


Figure 12. Mail Block Format

4.4.4 Adding or Replacing a Program in a Library

SMIPS programs are stored in two partitioned data sets named K3.S0JGM.S1011.VICINT for the modules VICINT, INTRPROC, TASKPROC and the executable image processing programs and K3.S0JGM.S0002.PAXII for all subroutines. The following examples show how to add a SMIPS program and a subroutine to the libraries.

The JCL statements required to add an image processing program (FOTO in the example) to the data set K3.S0JGM.S1011.VICINT are listed in Fig. 13. The JCL statements needed to add a subroutine (PICTUR in the example) to the subrouting library K3.S0JGM.S0002.PAXII is shown in Fig. 14.

```
// EXEC LINK,PARM='XREF,MAP,LIST,LET'
//SYSLIB DD DSN=SYS1.LINKLIB,DISP=SHR
//        DD DSN=SYS2.LINKLIB,DISP=SHR
//        DD DSN=K3.T1DAK.T1005.LVICARSB,DISP=SHR
//        DD DSN=SYS1.FORTLIB,DISP=SHR
//        DD DSN=SYS2.FORTLIB,DISP=SHR
//SYSLMOD DD DSN=K3.S0JGM.S1011.VICINT,DISP=OLD
//SYSLIN DD *
 CHANGE  USERPROG(PICMAG)
 INCLUDE LIBRY(LINKUSER)
 INCLUDE SYSLIB(FAKIBCOM)
 INCLUDE LIBRY(PICMAG)
 INCLUDE LIBRY(COLPIC)
 INCLUDE LIBRY(CHCKRT)
 INCLUDE LIBRY(TEXTGEN)
 INCLUDE LIBRY(LINECH)
 ENTRY USER
 NAME FOTO(R)
//LIBRY DD DSN=K3.S0JGM.S0002.PAXII,DISP=SHR
%
```

Figure 13

```
// EXEC LINK,PARM='NCAL,MAP,LIST'
//SYSLMOD DD  DSN=K3.S0JGM.S0002.PAXII,DISP=OLD
//OBJECT DD *
 ENTRY PICTUR
 NAME PICTUR(R)
%
```

Figure 14

NOT REPRODUCIBLE

5. DATA ORGANIZATION IN SMIPS

5.1 DATA STRUCTURES

For the user of an image processing system the pictoral information has to be structured in terms of his terminology. This information is converted to another form for storage and processing by the computer. In a computer system the information is usually structured in files, records and fields. This section reviews briefly some operations on data structures and describes the data organization in the SMIP system.

A record is a collection of information; it is divided into fields. In SMIPS an image line is treated as a record of variable length. All fields of a record are of equal length (byte, halfword, fullword, doubleword) and hold one picture element (1 pixel). A file (data set) is a collection of records. In SMIPS a picture is represented by one file. Picture files have a sequential organization. A block is the amount of information transferred between an I/O device and memory during one I/O operation. Blocking, the storing of a number of records in one block, is used in SMIPS to save space and computer time.

Each picture existing in the system has location, name and value as attributes. Each of these elements is structured and there exist relations between them. There are three sets known to the system.

N set of names

L set of locations

V set of values

L, the set of physical locations, is in one to one correspondence with N.

Name Space

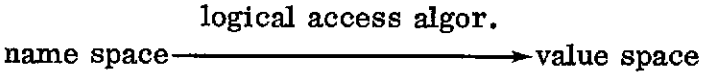
The name of a picture is the attribute which is given to it when it is created. The name is unique; at the same time several pictures with the same name cannot exist.

To access a picture by its name a relation between the name and the location of the picture (the table of contents) has to be established. This relation is called the catalogue of picture names.

$$\text{name space} \xrightarrow{\text{catalogue}} \text{table of contents}$$

Another role of the catalogue is to determine the uniqueness of the names. Upon creation of a new picture name the catalogue is consulted to determine if a picture with the same name already exists. A name is a string of alphameric characters beginning with a letter and is created by the user (external name). The system then automatically assigns an internal name to the picture.

The logical access algorithm gives the value from the name.



Value Space

The value of a picture is the information stored in its location. It is the result of the execution of the logical access function with the picture name given as a parameter. The data value has to be interpreted according to the coding convention. In an image processing system the data are binary coded intensity values and alphameric characters representing a description of the picture (size of picture, origin, date of creation, spectral band, history of processing, etc.).

During the existence of a picture in the system its value can change as well as the size of the location.

There are certain defining rules associated with picture data.

-range and coding of the values
-different methods of naming

This set of rules defines the type of data and the data of same type have a generic name. The table of generic names is called symbol table.

In a sequentially organized file where only the data item following the last accessed is accessible, it is not necessary to have a name for each item. In this case the use of the generic name is not ambiguous. Thus, a name in SMIPS refers to a file holding an image.

The second type of information in an image processing system are application programs. These programs exist in a file with partitioned organization. The name of the file is the generic name and each member is referred to by its proper name. Several of these files concatenated form a library.

5.2 OPERATIONS ON THE DATA STRUCTURES

Five sets have been defined:

L	set of locations
N	set of names
NG	set of generic names
V	set of values
0	set of undefined values

The following operations on the data structures representing these sets are used in SMIPS:

1. Creation of a picture
   The creation consists of three steps:

   a) Allocation ($0 \rightarrow L$)
   Find a free place in the table of contents for the disk on which space for the picture is to be allocated. The size is furnished by the symbol table. Allocation of disk space for SMIPS images is accomplished by the operating system in a separate job (ALLOCATE).

   b) Naming ($0 \rightarrow N$)
   The picture name is specified (created) by the user with the RESERVE command. A name already used for another picture is refused. The RESERVE command allows allocation and naming at the same time. In the OS/360 operating system dynamic allocation of disk space is not possible. Therefore, the allocation of the disk space used for storing images is accomplished in a previous job. The RESERVE command selects only an area of the preallocated space and assigns a name to it.

   c) Initialization ($0 \rightarrow V$)
   This operation takes place after allocation and naming. In the meantime, the picture exists but has no value. SMIPS does not allow to access a picture before its initialization. Initialization is accomplished with the READ, SAVE or THRESHOLD commands or as output of an image processing task.

2. Logical access ($N \rightarrow V$)
   Logical access is the product of localization ($N \rightarrow L$) and physical access ($L \rightarrow V$). These operations are performed by SMIPS routines and the operating system.

3. Suppression of a picture
   The suppression of a picture consists of:

   a) Deallocation ($L \rightarrow 0$)
      The space occupied by the picture is returned to the list of available space.

   b) Unnaming ($N \rightarrow 0$)
      The picture name referred to is deleted from the catalogue. This name could be used to name another picture.

These operations are accomplished with the FREE command.

Access to image processing programs in the library is entirely accomplished by functions of the operating system.

5.3 INTERFACE SMIPS TO OS/360

Data manipulated by OS are called data sets (corresponding to a file), their symbol table is constituted by the Data Control Blocks (DCB's). Each data set is located on a volume (disk, tape) and each volume has a table of contents, called VTOC, associated with it. An element of the VTOC is called a Data Set Control Block (DSCB). Pictures are stored in sequential data sets, image processing programs in partitioned data sets which form the system library.

The DDNAME is the external generic name for a class of data sets, the DCB address is used as internal generic name. The Job File Control Block (JFCB) gives the correspondence between DDNAME and the corresponding volume. Due to the sequential organization of a picture data set there is a one to one correspondence between a picture name and the generic name of a data set. Therefore, the correspondence between a picture name and the data set allocated for that picture is established in the following way. The index of the picture name in the catalogue is used to calculate another index for retrieval of the DCB address from the symbol table (see Fig. 15). The same index is inserted in the Maintenance Control Block (MCB) for the associated logical Data Set Reference Number (DSRN). The DSRN's are used in the I/O calls to the I/O handler.

For image processing programs specified in the input mode, the routine TASKQ determines the indices from the referenced picture names and passes them to TASKPROC, which in turn inserts the indices into the MCB's. Upon execution, the I/O routines called in the image processing program use these indices in the MCB's to retrieve the associated data sets. Routines doing I/O operations in the modules VICINT and INTRPRET should set themselves the appropriate symbol table indices in the MCB's.

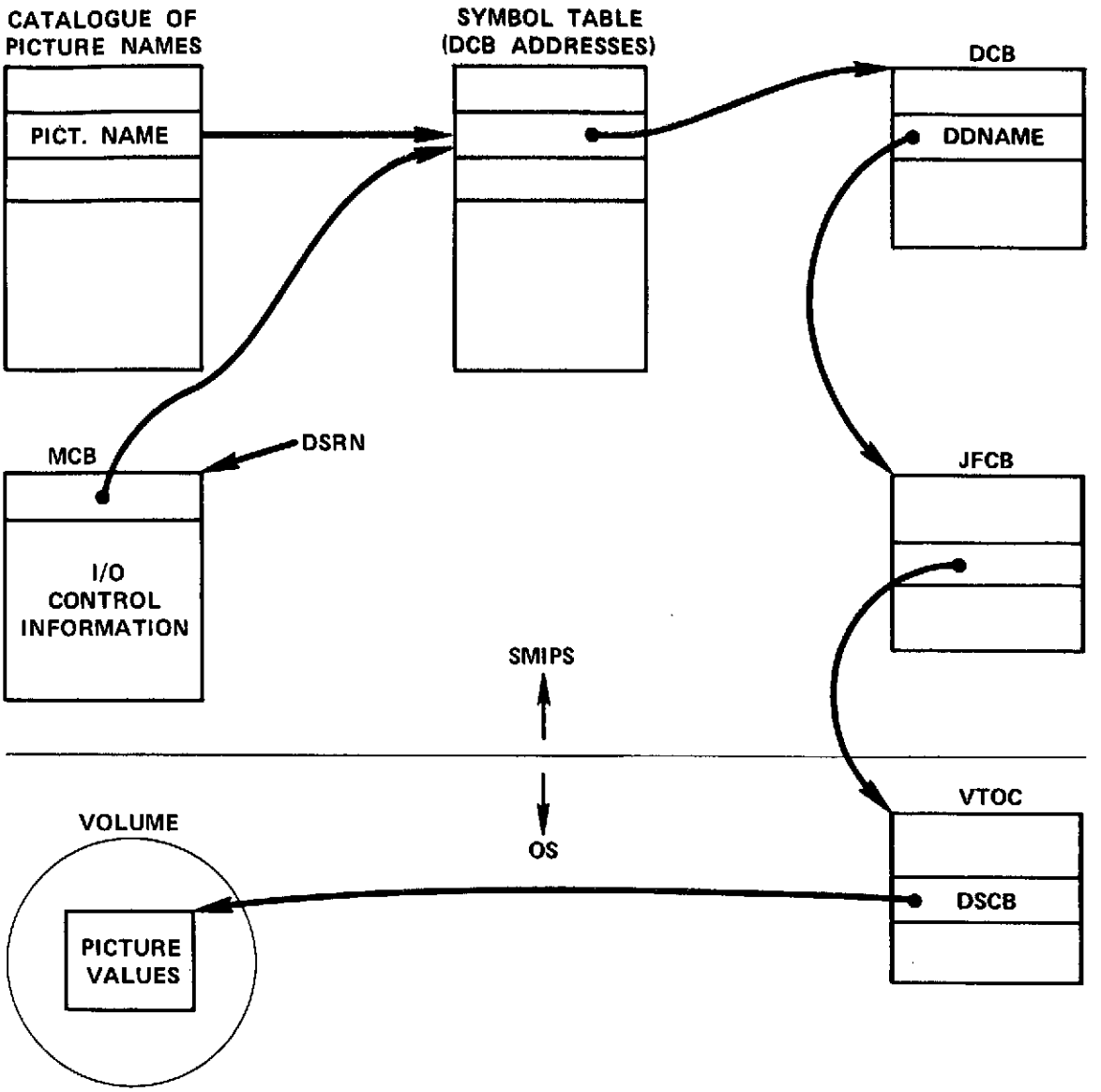


Figure 15. Correspondence Picture Name-Picture Values

5.4 DATA MANAGEMENT IN SMIPS

Data management in SMIPS is disk oriented. The display transformation routines and application programs read pictures only from disk and result pictures are written to disk data sets in the standard format. The transfer of pictures between magnetic tape and disk data sets has to be explicitly specified by the user with the READ and WRITE commands respectively. An exception are the application programs which produce tapes for the Calcomp plotter and the E.I.S.

machine. These programs write their output directly to tape because of the special format used.

Before a picture can be used it must have been created in the system. The three steps of creation (allocation, naming, initialization) are described in section 5.2. A picture can be used if it has been initialized by a READ, SAVE or THRESHOLD command or as output picture in an image processing task.

The catalogue of picture names is the field NTABLE in COMMON/LRECL/. In the current implementation the catalogue can hold up to 20 picture names. Each picture name has several attributes associated with it. The initialization flag which indicates if the picture is initialized and the core flag indicating if the picture is in core are contained in the field USFLAG in COMMON/LRECL/. The record length (line length) of the picture and the blocksize used to pack several lines into a physical record are contained in the field LRECLS in COMMON/LRECL/. These fields are accessed with the same index as the catalogue.

The RESERVE command inserts the picture name into the catalogue, computes the blocksize and inserts record length (equal to NS) and blocksize into the field LRECLS. The FREE command deletes the picture name from the catalogue and sets the flags, record length and blocksize to zero. The released space can then be allocated to another picture.

The INPUT/OUTPUT handler is a multientrant part of the core-resident module VICINT. It consists of a set of routines, callable from FORTRAN and assembly programs, that provide the interface between the image processing programs and the data management of OS/MVT. The Basic Sequential Access Method (BSAM) is used. The I/O handler supports input/output operations and other data management functions like opening, closing and double buffering of data sets. These procedures are controlled by Maintenance Control Blocks (MCB's). There is an MCB for each data set. An MCB contains fields for control information such as:

data set opened or closed

single or double buffer

input or output data set

device type (tape or disk)

I/O status

current record number pointer

blocking factor

number of labels

There are also fields to keep track of buffer addresses and lengths and for the index in the symbol table (DCB Table). Before opening a data set, record length and blocksize are automatically copied from the table LRECLS into the corresponding DCB. This is necessary because the user can allocate the same data set for pictures of various sizes during a session.

The data set reference number used in the I/O statements specifies which MCB is associated with a data set. Thus, the MCB is the interface between a logical data set reference number and a physical device. The I/O handler is a modified version of the VICAR I/O routines which are described in [3].

6. GRAPHIC PROGRAMMING TECHNIQUES

The problem of graphic programming is to find an efficient representation of an image and the relations among its objects in memory. This representation must permit the easy manipulation of the represented objects.

Each graphic terminal has its own language which allows to describe the operations necessary to obtain an image on the screen. This is usually a very low level language and it is laborious to write graphic programs with it. The instruction set is also very limited allowing only to display points, vectors and possibly characters. It is thus necessary to define a high level graphic language whose instruction permits the specification of more general graphic objects and to allow for control of luminosity, density of points, use of light pen etc. . . High level graphic languages have been implemented by embedding in a general high level language [4,5].

Considering the restricted number of display types in the SMIP system it was decided not to implement a high level graphic language but to use the basic orders and macro-instructions provided for the IBM 2250 display terminal [6]. This allows to optimize the design for this specific application on a given display processor.

6.1 DEFINITION OF AN IMAGE

The basic images (points, vectors, characters) provided by a graphic terminal are called elements. A graphic object is a set of elements illuminated with the same intensity. It is obtained with one instruction (e.g. a string of characters). An object can be composed of only one element. A figure is a collection of objects to which a name is assigned.

An image is the set of figures displayed at one instance on the screen. A certain ambiguity seems to exist in the fact that by using the terms figure and image

it is not clear if the representation in memory or the display on the screen is meant. There is virtually no difference between these two notions. In the following sections image or figure will be generally used for their representation in memory.

6.2 IMAGE STRUCTURES

The representation of an image has also to express the relations that exist among the figures forming an image. Such a representation is called an image structure. A display file is the list of instructions and data of an image. By repeatedly executing this list, the display processor attempts to maintain a flicker-free picture on the screen of the cathode-ray tube. In such a simple structure each figure is repeated in the list at each occurrence. If an image has to be modified the entire display file has to be changed.

An improvement of this technique is the use of structured display files. Here figures are constructed by subroutines and the display file is organized into a tree structure as is shown in Fig. 16. The display processor refreshes the picture by threading its way through this structure, interpreting the pointers as subroutine calls. Structured display files are more easily modified than the unstructured files, since one element of the display can be modified without disturbing the rest. Objects which occur repeatedly can be represented by subroutines, and in this way the size of the display file can be reduced. It is also possible to link interrupts, received from a pointing device such as a light pen, to the appropriate parts of the structure. Structured display files have some drawbacks. They are designed for a specific display device so that refreshing may be carried out as fast as possible. As a result it is usually very troublesome

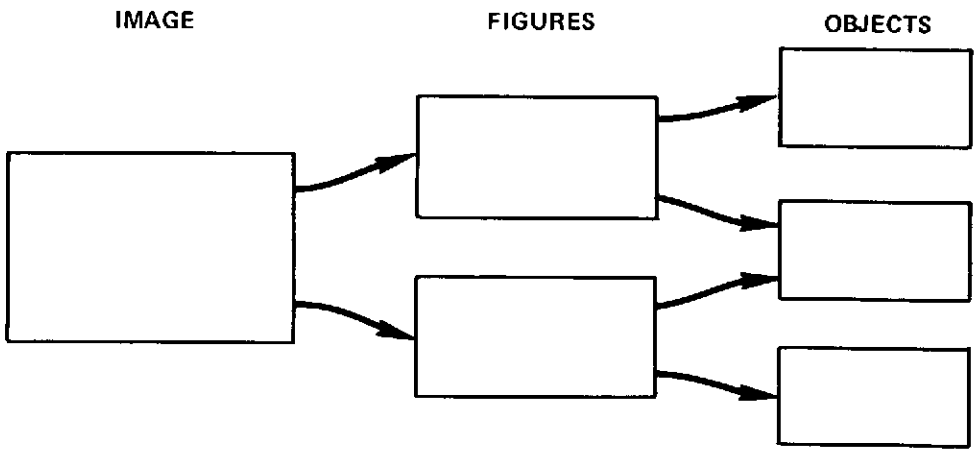


Figure 16. Structured Display File

to convert graphical programs from one machine to another. Because structured display files are designed for a particular display device, they are bound by the limitations of the device. In a display processor incapable of modifying the scale of the information passed to it, the picture must be displayed at a fixed scale. Scaling is done by software and two versions of the picture are necessary, one for refreshing the display and the other from which scaled display files are generated [6].

Because scaling is a very important procedure in an interactive image processing system, the SMIP system uses a scaled display file. The transformation routines taking pictorial information from the general picture data structure are combined with the graphic output routines which feed into the scaled display file (Fig. 17). For picture display, the routine PICTUR is the transformation routine which scales and possibly truncates the picture in order to fit on the screen of the display device. PICDSP is the graphic output routine whose function is to assign characters to greylevels and to write the result of PICTUR to the figure PICT in the scaled display file which is maintained in the buffer of the display device.

The scaled display file of the SMIP-System consists of six figures. They are called HEAD, FRAME, MSG, INPUT, GRID and PICT. HEAD and FRAME are rigid figures representing the headline and the frame of the image displayed. MSG is used to display messages from the system for the user, the figure INPUT displays the user input on the alphameric keyboard. PICT is the figure displaying information in the form of pictures, histograms and alphameric descriptions. The figure GRID is overlaid on PICT and is used for the display of grids or polygons outlining areas with same greyvalues.

The graphic orders and data constituting the display files are organized in a linked tree structure and are continuously regenerated for visual display. A figure is included in the visual display by setting a pointer to its display file and deleted by removing the pointer. The last instruction in each figure is a transfer back to the following location in the root of the display file tree. Interrupts from the light pen can only occur in the figure PICT, they can therefore be easily related to the item at which the light pen pointed.

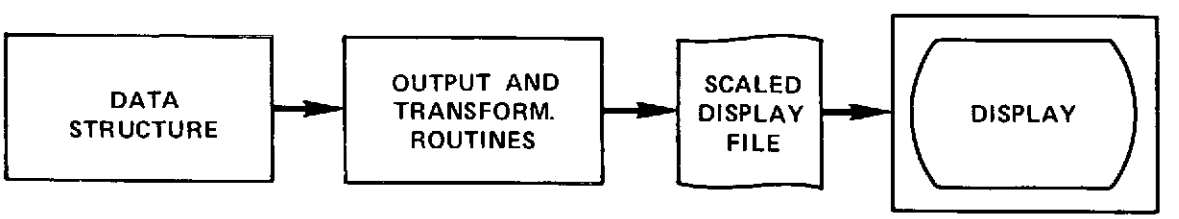


Figure 17. Scaled Display File

The root of the display file tree is shown in Figure 18.

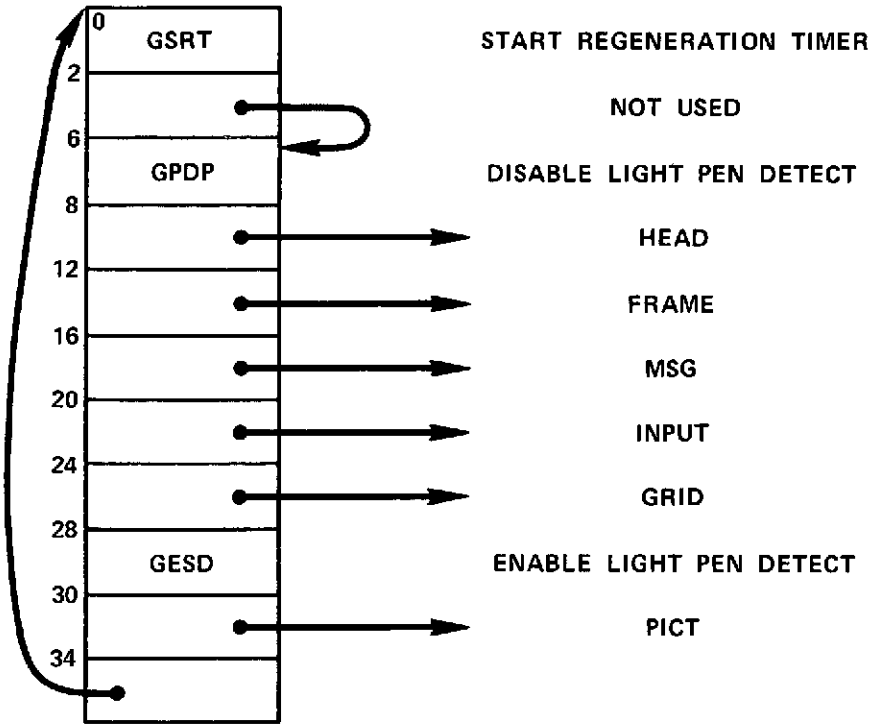


Figure 18. Display File Tree

6.3 DESCRIPTION OF THE IBM 2250 DISPLAY UNIT [8]

The 2250 is organized around a cathode ray tube on which graphic and alphanumeric information is displayed, thereby providing visual communication between the computer and its user. Keyboards and light pen provide a means of entering and modifying information. The 2250 equipped with the absolute vector graphics features can display points and vectors at any angle on a raster of 1024 by 1024 points. Points plotted four or more raster units apart can be distinguished as discrete points. A standard character set of 63 alphabetics, numerics and special symbols is provided by the character generator. 74 basic size characters can be displayed in each of 52 lines. The distance between the center of two characters is 56 raster units, the distance between two lines is 80 raster units. The display area is 12 inches by 12 inches. The alphameric keyboard is a typewriter like keyboard with which the user can compose commands for entry into the system. 44 keys and a space bar provide a selection of 63 characters. In addition there are the keys:

SHIFT	allows selection of the upper character in a dual character key.
ALT	unlocks the keyboard when depressed with SHIFT.
END	signifies the end of manually entering alphameric characters from the keyboard. When depressed with ALT it initiates the transfer of the character string entered to main memory.
SPACE	advances the cursor one buffer position.
BACKSPACE	backspaces the cursor one buffer position.

Depression of a character key provokes insertion of the character into the buffer position indicated on the screen by the cursor symbol. The cursor symbol is a dash displayed beneath the character position at which the character selected at the keyboard will be placed. If the cursor is not displayed, characters can not be inserted.

The programmed function keyboard consists of 32 keys and light indicators. The function of each key is program defined and is identified to the user by the overlay. When a key is depressed, an attention signal is sent to the CPU and the computer acts as directed by the subroutine associated with the selected key. The indicators can be lit or extinguished under program control and inform the user that certain keys can be activated.

The light pen enables the user to communicate with the computer by pointing a pen-like device at a portion of the displayed image. Once the light pen is properly positioned, the user depresses the light pen to the screen to close the springloaded tip switch. This causes an attention signal sent to the CPU and the system acts on the identified portion of the display as determined by the associated subroutine.

The visible display on the CRT is produced by action of an electron beam causing the phosphor coating to glow briefly. The glow fades within a fraction of a second, therefore, the display must be regenerated at such a rate that it appears steady to the observer. The regeneration rate is variable up to a rate of 40 cps (25 ms) dependant on the amount of information displayed. If the time required to execute a sequence of graphic orders that follow a Start Regeneration order (GSRT) exceeds 25 ms, the display will flicker. A suitable display is obtained with a regeneration rate of 30 to 40 cps.

The 2250 disposes of a local buffer to store images for display regeneration. This enables the 2250 to operate concurrently with the computer system freeing

main storage and the channel for other purposes. I/O interface activity is only required when a new display file is sent to the buffer. The buffer size is 8192 bytes. For SMIPS the buffer of the 2250 display device is divided into fixed partitions. Each partition can hold a figure of the structured display file. This allows efficient transfer of parts of an image between buffer and main memory rather than the transfer of the whole image at each modification. The addresses of the partitions are contained in the buffer control table. The table contains also the address of the work area in which the display file or figures of it are constructed prior to transfer to the buffer (Fig. 19).

6.4 PROGRAMMING THE 2250 DISPLAY UNIT

A sequence of graphic orders interleaved with data constitutes a program for the display device. The orders determine a specific operation, such as drawing a line, displaying a character or transferring to another buffer address for the next instruction. The data following each order contain the information necessary to perform the specified operation. The first instruction in a graphic program must be the start regeneration order, the last order should be a transfer to the beginning of the program. While executing this program the display unit produces an image on the screen. The graphic program can be executed only in the device buffer.

The basic programming technique for the 2250 display unit consists in the following steps:

1. Build a scaled display file or parts of its structure as a graphic program in a workarea in main memory. For the SMIP system the output and transformation routines provided in the system are to be used.

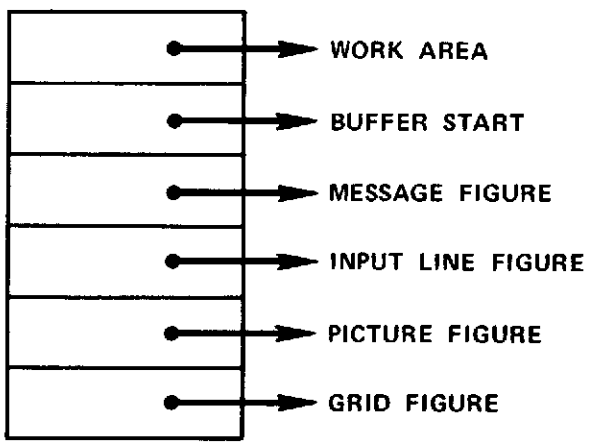


Figure 19. Buffer Control Table

2. Transfer the display file to the appropriate location in the 2250 buffer.
3. Start execution of the regeneration sequence.

Commands from the user are assembled and edited in the buffer. On request they are transferred to main memory for further processing. The graphic orders and the instructions to initiate the data transfer between main memory and device buffer are listed in [6].

6.5 COMMUNICATION HANDLING

Communication with the system is initiated by depressing keys on the keyboards or by touching a part of a display with the light pen. Any of these actions results in an attention. The SMIP system provides routines to be entered on receipt of such an attention.

The SMIP system supplies information to the operating system concerning which routines are to respond to what types of attentions. The operating system detects the occurrence of an attention, interrupts the currently processing routine and passes control to the appropriate attention handling routine. After this routine has performed its function, control returns to the interrupted routine. Attentions for which no attention handling routine is available are ignored.

Each SMIPS routine which has to process communications with the user is designed according to the following rules:

1. Define the attention handling capabilities of the routine to the operating system. In particular specify the attention handling routines and the types of attentions to be serviced by these routines. Define a communication area in main storage to which the operating system will pass attention information (type of attention, key number, coordinates of light pen position).
2. Enable operating system references to the attention routines.
3. Wait for specific attentions.
4. Upon return from the attention handling routines disable operating system references to these routines.
5. Process the attention information in the communication area.
6. Go to step 2 if no exit condition was specified in step 5.

The only function of the attention handling routines is to notify the operating system on completed attention processing. The actual processing is performed in

the main routine rather than in the attention handling routine to facilitate communication between various parts of the system.

6.6 COMMUNICATION WITH THE 2250 DISPLAY UNIT

All system routines and application programs are allowed to access the display unit and to send figures to its display file. Communication with the 2250 is established with the Graphic Control Block (GRCBLK). The GRCBLK is built by the routine DISPIN in VICINT. It contains the address of the DCB for the 2250 display unit (DDNAME = GRAVIC), the address of the Data Event Control Block (DECB) for I/O operations of the 2250, the address of the Output Area Control Block (OACB) used for I/O operations, the address of a 100 word work area used by problem oriented graphic routines and the address of the buffer control table (Fig. 20). Any routine which is not in the core resident module VICINT establishes communication with the 2250 by a CALL GRCBLK which returns the address of GRCBLK in register 1 from the transfer vector.

7. LIGHT PEN TRACKING

The SMIP System provides for a light pen tracking capability. Light pen tracking could be used to outline areas by border detection and to produce maps from the pictorial data. Basically connected regions in form of polygons are generated on the screen with the aid of the light pen. The coordinates of the corner points of each polygon are recorded in a data structure (a sequential list).

The Depression of key 28 labeled TRACK enables the system for light pen tracking. The user may now select a picture point with the light pen and then press key 29 labeled PICK which will trigger the procedure to record the coordinates of the selected point (Note: if a horizontal or vertical line pattern appears on the screen, move the light pen slightly perpendicular to the direction of the pattern).

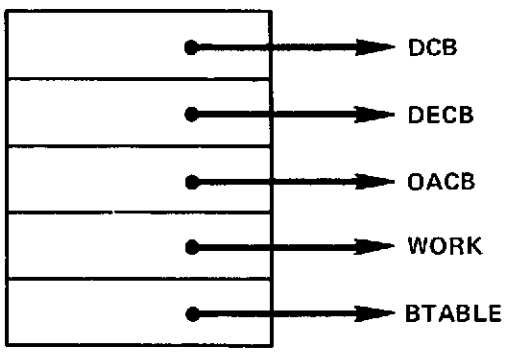


Figure 20. Graphic Control Block

Repetition of this procedure will result in drawing a straight line segment between the current and the previously selected points. The last line segment drawn can be removed by pressing key 25 labeled DELETE.

To terminate a polygon and start construction of a new polygon depress key 26. Terminate light pen tracking and return to the normal dialogue mode by depressing key 27 labeled NOTRACK.

REFERENCES

[1] Moik, J. G.: Small Interactive Image Processing System (SMIPS), Users Manual. GSFC X-Document X-650-73-283.

[2] Newman, W. M.: A System for Interactive Graphical Programming. Proc. 1968 Spring Joint Computer Conference. Thompson Books 1968.

[3] JPL Digital Image Processing System Manual VICAR-Version 3, August 1968. Re-order No. 68-369.

[4] Rully, A. D.: A subroutine package for FORTRAN. IBM Systems Journal 7(1968), 248-270.

[5] Kulsrud, H. E.: A General-Purpose Graphic Language. Comm. ACM 11, 4 (1968), 247-254.

[6] IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit. Form C27-6909-5.

[7] Newman, W. M.: Display Procedures. Comm. ACM 14, 10(1971), 651-660.

[8] IBM System/360 Component Description: IBM 2250 Display Unit Model 1. Form A27-2701-1.