View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

JPL PUBLICATION 77-37

VICAR Image Processing System Guide to System Use

(NASA-CR-154102) VICAR IMAGE PROCESSING SYSTEM GUIDE TO SYSTEM USE (Jet Propulsion Lab.) 59 p CSCL 09B
N77-28829
Unclas
G3/61 39280

AUG 1977
RECEIVED
NASA STI FACILITY
INPUT BRANCH

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

JPL PUBLICATION 77-37

VICAR Image Processing System Guide to System Use

Joel B. Seidman

May 1, 1977

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

Prepared Under Contract No NAS 7-100
National Aeronautics and Space Administration

PREFACE

The work described in this report was performed by the Earth and Space Sciences Division of the Jet Propulsion Laboratory.

This document is a major revision of the original issued on October 1, 1968 as an internal document and subsequently revised by H. Freiden in July 1971. This revision describes the OS version of VICAR and incorporates the descriptions of the pre-processor (EVIL) and the functions of the post-processor (VICARDSN).

ABSTRACT

This document describes the functional characteristics and operating requirements of the VICAR (Video Image Communication and Retrieval) system.

The information contained herein is applicable to the version of the VICAR system operating in conjunction with the IBM OS and VS operating systems.

Section 1 contains an introduction to the system describing the functional characteristics and the basic theory of operation. A brief description of the data flow as well as tape and disk formats is also presented.

Section 2 is a formal presentation of the control statement formats. This section is intended to serve as a reference guide for the programmer.

Section 3 is organized as a guide to the usage of the system. It provides a step-by-step reference to the creation of a VICAR control card deck. Simple examples are employed to illustrate the various options and the system response thereto.

TABLE OF CONTENTS

1.0 INTRODUCTION 1

1.1 Purpose 1
1.2 Pre-requisite Publications 1
1.3 Functional Characteristics 1
1.4 Task Characteristics 2
1.5 VTRAN Functions 2
1.6 VMAST and VMJC 3
1.7 Standard Tape Format 4
1.8 Standard Disk Data Set Format 5

2.0 LANGUAGE 6

2.1 Control Statement Format 6
2.2 Control Statement Functions 6
2.3 Control Statement Specifications 7

2.3.1 RESERVE,BLOCK,A,B 8
2.3.2 SAVE 9
2.3.3 FIND 10
2.3.4 RELEASE 11
2.3.5 READ 12
2.3.6 WRITE,TAPE 15
2.3.7 EXEC,E 16
2.3.8 PARAMS,P 20
2.3.9 LABEL,L 21
2.3.10 RELABEL,R 22
2.3.11 NOTE 22
2.3.12 END 23
2.3.13 TIME 24
2.3.14 REGION 24

2.4 Restrictions 25
2.5 Pre-Processor Statements (EVIL) 25

2.5.1 DEFINE,D 26
2.5.2 END 27
2.5.3 CALL,C 27
2.5.4 GET,G 28

2.6 Post-Processor Statement 29

3.0 USAGE 30

3.1 VICAR Job Set-Up 30
3.2 RESERVE Control Statement, The 31
3.3 SAVE Control Statement, The 32

- 3.4 FIND Control Statement, The 33
- 3.5 RELEASE Control Statement, The 33
- 3.6 READ Control Statement, The 33
- 3.7 WRITE Control Statement, The 34
- 3.8 EXEC Control Statement, The 35
- 3.9 PARAMS Control Statement, The 38
- 3.10 LABEL Control Statement, The 40
- 3.11 RELABEL Control Statement, The 41
- 3.12 NOTE Control Statement, The 41
- 3.13 END Control Statement, The 41
- 3.14 TIME Control Statement, The 42
- 3.15 REGION Control Statement, The 42
- 3.16 DEFINE Control Statement, The 42
- 3.17 CALL Control Statement, The 43
- 3.18 GET Control Statement, The 44

4.0 EXAMPLE JOBS 47

- 4.1 Example 1 47
- 4.2 Example 2 47

5.0 ERRORS 48

- 5.1 VTRAN Errors (First Job) 48
- 5.2 Execution Errors (Second Job) 49

APPENDICES

A. EVIL2LIB Procedure Library Maintenance Program 51

B. JCL Insertions in Second VICAR Job 52

TABLE

2.3.5 Tape Format Codes 14

1.0 INTRODUCTION

1.1 Purpose

The VICAR system is designed to facilitate the acquisition, digital processing and recording of image data. System objectives include ease of operation by personnel who may not be expert programmers as well as simplification of future programming effort. The system was originally designed for operation with the IBM 360/44 Programming System (44PS). This document describes a version subsequently adapted to the IBM OS Operating System; the version also runs properly under OS/VS systems. It provides the necessary routines to complement the operating system and adapt it to meet the needs of image processing.

1.2 Pre-requisite Publications

VICAR is designed so that the user needs very little knowledge of the operating system. However, it would be helpful if the user had available for reference the following IBM publications.

1.2.1 GC28-6534 "IBM System/360 Operating System: Introduction"

1.2.2 GC28-6704 "IBM System/360 Operating System: Job Control Language Reference"

1.2.3 GC28-6631 "IBM System/360 Operating System: Messages and Codes"

In addition, the reader will require user guides which describe the characteristics and requirements for the various problem programs to be executed.

1.3 Functional Characteristics

Operation of the VICAR system is very similar to that of a standard utility program. The programmer provides a card deck containing VICAR control statements which define the attributes of the job including the processing tasks. A system program VTRAN, which may be considered either as a utility program or as a simple compiler, uses these control statements to generate any required job control statements as well as a system task queue. This system task queue contains all of the data needed to execute the specified tasks.

The job control statements and system task queue are submitted as a separate job to the operating system's internal reader. In the second job, another system program VMAST is loaded and remains in main storage throughout the job. VMAST, in turn, loads a transient routine VMJC which reads the system task queue and initiates the first task. Upon task completion, VMJC is reloaded and the next task is initiated. When the last task has been completed, control is returned to the operating system for job termination.

VICAR processing is initiated by an OS job control language statement which invokes the VICAR cataloged procedure. This procedure in turn executes three OS job steps. The first step invokes a preprocessor which scans the control statements for preprocessor commands and translates them to one or more non-preprocessor commands. Non-preprocessor commands are unchanged by the preprocessor. Output from the preprocessor is passed to the second job step which invokes the main VICAR translator program VTRAN which generates the JCL and task queue for the second job as described previously. The final job step of the VICAR procedure passes the JCL and task queue to the operating system internal reader.

A separate procedure, VICARDSN, may be executed instead of VICAR. Its function is identical to VICAR, except it adds a post-processing step which allows VICAR to use data set names up to 44 characters in length.

1.4 Task Characteristics

A VICAR job consists of a sequence of processing tasks. Each task involves the execution of a program which must exist in the VICAR program library. The processing programs communicate with the supervisor through the system routine VMAST. This routine supervises all of the processing program's I/O operations. The programs may be written in any desired programming language, typically either Assembler Language or Fortran IV. VMAST supports up to fourteen simultaneous I/O files and the processing of any tape or direct access data file may be either sequential or random.

A processing program is required to open data sets prior to use. In certain cases, the program may close its data sets. If not, VICAR will automatically close any open data sets at the end of the task. A task may unintentionally terminate abnormally, causing the operating system to cancel the job and provide a dump if requested. On the other hand, a task may elect to intentionally terminate abnormally. VICAR may proceed to the next task and continue the job, or may cancel the job, depending on control statements.

1.5 VTRAN Functions

As mentioned, VTRAN generates any required job control statements. VTRAN also generates the requisite task, parameter, label and relabel statements which make up the task queue. In addition, VTRAN prints a list of the VICAR control statements on the system printer.

The user specifies disk data set allocation requirements with simple RESERVE control statements, and VTRAN generates the requisite "DD" job control statements.

The user specifies the characteristics of input and output tapes with simple READ and WRITE control statements. VTRAN generates the required "DD" job control statements.

The user specifies the number and order of the tasks which are to be performed with simple EXEC control statements, and VTRAN generates all of the necessary task statements in the system task queue.

Parameters for the above tasks may be specified by the user, either with simple PARAMS control statements and free-form parameter cards or in the EXEC control statements themselves. In either case, VTRAN generates all of the necessary parameter statements required in the system task queue.

In addition, where possible, VTRAN provides diagnostic messages upon detecting errors or possible errors in the VICAR control statements.

1.6 VMAST and VMJC

VMAST contains system service routines, and is resident in main storage throughout the execution of all tasks. When VMAST is initially loaded, and at the end of subsequent tasks, VMJC is loaded, overlaying the current processing program. VMJC reads task, parameter, label and relabel statements from the system task queue.

VMJC initializes control blocks (MCB's) in VMAST for each task. Processing programs are written using data set reference numbers (1-14). In initializing the MCB's, VMJC establishes the linkage between these data set reference numbers and a specific data set or device.

The processing programs, in general, expect a standard data format for all input and output data sets. This format consists of a set of label records followed by a number of data records. Normally, VMJC copies the labels from the primary input data set to any specified output data sets. The system label record (to be described) is updated. If the user has specified optional user labels to be added, VMJC adds these to the label set on the output data sets. This automatic label processing may optionally be suppressed.

In addition, VMJC positions all input and output data sets to a point just prior to the first data record; that is, just following the last label record.

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. VMJC translates all parameters to an internal computer representation which the processing programs can utilize. The translated parameters are written temporarily into a disk data set. Processing programs obtain these parameters by issuing a PARAM call to VMAST.

After VMJC has finished its processing, it loads the processing program specified for the task. This program is loaded overlaying VMJC.

At the conclusion of a processing program task, the task returns to VMAST. VMAST performs some minor end-of-task processing and reloads VMJC to continue.

1.7 Standard Tape Format

A standard format has been established for both tape and disk data sets. All of the standard VICAR processing programs are designed to operate with this standard data set format.

Video samples are normally represented as eight-bit data bytes. With the Data Converter Feature of the IBM 360 tape-controller, both seven and nine track tapes may be logically equivalent. Each reel of tape may contain up to 99 files, determined only by the size of the files. Each file contains one video frame, or image, and is followed by a trailing file mark. The last file on a reel must be followed by two trailing file marks.

All labels and data are recorded in an eight-bit mode, compatible with that specified in IBM SRL A22-6866. Label characters are recorded in EBCDIC. When written on 7 track tape, the eight-bit data samples are written in a mode where three eight-bit data bytes are recorded on tape as four six-bit characters.

Each file consists of a VICAR label set followed by data records. The label set consists of one or more 360-byte records. The 360-byte label records are subdivided into five 72-byte logical labels. The first 72-byte logical label in the label set is reserved for system use and must contain certain specified data. All subsequent logical labels may contain variable I-D information (text) as required.

Byte 72 of each logical label is used as a continuation character and contains a 'C' to indicate the presence of additional logical labels or an 'L' if it is the last such label. Likewise, a 'C' in byte 360 indicates an additional 360-byte label record. Variable data may be recorded in bytes 1 through 68. Bytes 69 through 71 are reserved for other system indicators.

1.8 Standard Disk Data Set Format

The format of the standard disk data set is logically identical to the standard tape format described above. All records in a direct access data set are the same length. To contain the label records, the minimum size of the records in a disk data set is 360 bytes. The maximum record size is the track size for the storage device. The programmer may specify a record size equal to the number of bytes in a video line or a greater amount.

2.0 LANGUAGE

Due primarily to the implementation of the VICAR system, there are two broad classes of VICAR control statements: standard statements and pre-processor statements. These statements may be freely intermixed as described subsequently. However, it will be convenient to describe them separately. Therefore, sections 2.1-2.4 describe the standard statements, while section 2.5 describes the preprocessor statements. In addition, there is an optional post-processor which may be used when data set names longer than 8 characters are needed. The post-processor is described in section 2.6.

2.1 Control Statement Format

VICAR control statements are designed for an 80-column punched card format. Statements may start in column 1 and cannot extend past column 71.

Each statement contains from one to ten fields. Fields are separated by commas. A field consists of one to ten sub-fields. Sub-fields are also separated by commas. If a field includes more than one sub-field, the field must be enclosed in parentheses.

Parentheses may be used if the field consists of only a single sub-field. Fields and sub-fields may be surrounded by blanks. Except where explicitly specified, sub-fields are limited to eight characters.

Certain statements permit a field or fields to be defaulted (not coded). If there are additional fields, the defaulted field must be indicated by coding a comma.

The following conventions are observed in this document. Parameters presented in upper case characters are required and must be coded exactly as shown.

Braces { } indicate that a choice must be made from among the optional parameters indicated.

Brackets [ ] indicate that the field or sub-field is optional. Default parameters are indicated in the text.

An ellipsis. . . indicates that additional parameters (sub-fields) may be coded.

2.2 Control Statement Functions

A list of the control statements and a brief description of their function follows.

1. RESERVE, BLOCK, A, B	Reserve temporary direct access storage (data sets)

2.	SAVE	Reserve permanent data sets for use in a subsequent job
3.	FIND	Access data sets created in a previous job
4.	RELEASE	Delete data sets created in a previous job
5.	READ	Specify device and data format for an input tape
6.	WRITE, TAPE	Specify device and data format for an output tape
7.	EXEC, E	Specify task, input and output data sets and required parameters
8.	PARAMS, P	Define a symbolic name for a set of parameters
9.	LABEL, L	Specify a label to be added to an output data set
10.	RELABEL, R	Delete existing labels and add a label to an output data set
11.	NOTE	Print a message on the output listing, and control certain VICAR functions
12.	END	Indicate the last control statement
13.	TIME	Set a limit on job CPU time
14.	REGION	Set a limit on job main storage utilization

2.3 Control Statement Specifications

The following specifications define control statement field and sub-field requirements.

2.3.1 RESERVE, BLOCK, A, B

The RESERVE control statement is used to allocate temporary disk storage space (data sets).

Operation	No. of Data Sets	Length	No. of Records	Volume ID
Field 1	Field 2	Field 3	Field 4	Field 5
$\begin{Bmatrix}\text{RESERVE}\\ \text{BLOCK}\\ \text{A}\\ \text{B}\end{Bmatrix}$	,n	,(rl [,bl])	,nr	,[volid]

Data Set Names
Field 6
[,(name1,name2, . . .)]

Field	Content
1	RESERVE or BLOCK or A or B
2	n is an integer from 1 to 9 specifying the number of data sets to be allocated.
3	rl is the record length (bytes per line). bl is the block length. bl may be omitted in which case VICAR will use the largest multiple of rl not exceeding 6447. (The maximum block length for a 2314 disk is 7294 bytes, and for a 3330 disk is 13030 bytes.)
4	nr is the number of records to be allocated in each data set. Sufficient records for labels should be included.
5	volid is the volume serial number of a disk pack, for example IPLSYS. Alternatively, an asterisk or null field may be coded, in which case the system will be allowed to assign a volume from the public SYSDA device pool.

Field	Content
6	name1,name2, . . . are names to be assigned to the allocated data sets. The names may be from one to eight characters in length. The first character must be alphabetic. Names may be omitted for any or all data sets. Unnamed data sets will be used by VICAR when undefined names appear in subsequent EXEC statements.

In this version of VICAR, BLOCK, A, and B are equivalent to RESERVE.

2.3.2 SAVE

The SAVE control statement is used to allocate permanent disk storage space (data sets).

Operation	No. of Data Sets	Length	No. of Records	Volume ID
Field 1	Field 2	Field 3	Field 4	Field 5
SAVE	,n	,(rl[,bl])	,nr	,[volid]

Data Set Names
Field 6
[,(name1,name2, . . .)]

Field	Content
1	SAVE
2	n is an integer from 1 to 9 specifying the number of data sets to be allocated.
3	rl is the record length (bytes per line).

Field	Content
4	nr is the number of records to be allocated in each data set. Sufficient records for labels must be included.
5	volid is the volume serial number of a disk pack, for example IPLSYS. Alternatively the field may be an asterisk or empty, in which case the system will be allowed to assign a volume from the storage SYSDA pool, and the data set will be entered in the operating system catalog.
6	name1,name2, . . . are names to be assigned to the allocated data sets. The names may be from one to eight characters in length. (Names exceeding eight characters may be employed using the post-processor.) The first character must be alphabetic. Although VICAR will assign names if they are defaulted by the programmer, it is recommended that the programmer always assign his own names.

2.3.3 FIND

The FIND control statement is used to access a disk data set created in a previous job. Its appearance is required in order to refer to the data set in an EXEC statement.

Operation	Data Set Names and Volume ID's
Field 1	Field 2
FIND	,(name1,vol1,name2,vol2, . . .)

Field	Contents
1	FIND
2	name1,vol1,name2,vol2, . . . are up to 5 pairs of data set names and volume serial numbers. The name is identical to the name specified when the data set was allocated. The volume serial number is that of the disk containing the data set. Alternatively, the volume id may be entered as an asterisk or omitted if the data set is entered in the operating system catalog.

2.3.4 RELEASE

The RELEASE control statement is used to find and delete a direct access data set created in a previous job. The function of this statement is identical with that of the FIND statement, except that the data set is deleted at the end of the job.

Operation	Data Set Names and Volume ID's
Field 1	Field 2
RELEASE	,(name1,vol1,name2,vol2, . . .)

Field	Contents
1	RELEASE
2	name1,vol1,name2,vol2, . . . are up to 5 pairs of data set names and volume serial numbers. The name specified is identical to the name specified when the data set was allocated. The volume serial number is that of the disk containing the data set. Alternatively, the volume id may be an asterisk or omitted if the data set is entered in the operating system catalog. (Caution: if the volume serial number is specified for a cataloged data set, the catalog entry will not be deleted even though the data set is.)

2.3.5 READ

The READ control statement is used to define the device and data format for an input tape. It may also be used to assign a symbolic name to the tape.

Operation	Symbolic Name	Tape ID
Field 1	Field 2	Field 3
READ	,dev	,volser

Tape Name	Format Code	Blocking Data
Field 4	Field 5	Field 6
,tname	,nx	[,(rl[,bl])]

Field	Contents
1	READ
2	dev should normally be an asterisk or omitted, causing a unique tape drive to be assigned to the tape. By entering a symbolic name of up to 8 characters for two or more such statements, the same physical tape drive can be assigned to several tapes.
3	volser is the volume serial number of the magnetic tape reel desired. It can be up to 6 alphanumeric characters. This name will appear in a message to the computer operator instructing him to mount the tape.
4	tname is a symbolic name assigned to the input tape. It may be from one to six characters in length and the first character must be alphabetic. This name is used in EXEC statements to refer to the tape. VICAR may append a two-digit file sequence number to the symbolic tape name to form a symbolic tape file name. The resultant tape file name will be from three to eight characters in length depending on the length of the tape name.

If Field 4 is defaulted, the system will assign the tape name INP to the input tape. When using more than one READ control statement, caution should be observed to prevent dual specification of the system assigned name INP.

Field	Contents
5	*nx* is a one- or two-character format code which specifies the data format of the input tape. The valid codes are listed in table 2.3.5.

If Field 5 is defaulted, the system will assign a format code equal to 8.

Field	Contents
6	This field is used for blocked tapes. *rl* is the record length and *bl* is the block length, specified in bytes. If *bl* is omitted, it is assumed equal to *rl*. If both are omitted, a default value of 7200 is assumed for both. If the assumed or specified values of *rl* and *bl* are equal, they need not be exact, but will usually be treated as upper limits for the actual sizes. In this case the actual record/block size will usually be equal to the number of bytes per data record as specified in the VICAR system label, if there is one. (Some application programs may be written to ignore the values in this field.)

Table 2.3.5

TAPE FORMAT CODES

Format Code	Description	Comments
9 8 5 6	These formats are logically identical, Formats 9 and 6 representing nine-track tape at 800 and 1600 bpi respectively, and Formats 8 and 5 representing seven-track tape at 800 bpi and 556 bpi respectively.	VICAR Standard Formats
8A	This format is a seven-track, 800 bpi, 8-bit data, unlabelled* tape.	Used for film recorder 8-bit data tapes.
8F 5F 2F	These formats are all logically identical, with 8F, 5F, and 2F representing seven track 800 bpi, 556 bpi and 200 bpi respectively. The data format is 6-bit, unlabelled.*	Used for film recorder 6-bit data tapes.
8L 5L 2L	These formats are all logically identical 8L, 5L and 2L representing seven-track 800 bpi, 556, and 200 bpi, respectively. The data format is 6-bit with 80 character labels.*	Used for obsolete Surveyor 6-bit data tapes.

Further information about the tape formats is contained in the Introduction (Section 1).

---

*Throughout this document, "label" means VICAR label. It should not be confused with operating system standard labels, which are not supported by the VICAR system.

2.3.6 WRITE, TAPE

The WRITE control statement is used to define the device and data format for an output tape. It may also be used to assign a symbolic name to the tape. In addition, the WRITE control statement may be used to specify symbolic names for data sets which are to be copied to the output tape.

Operation	Device Address	Tape ID and Data Set Names
Field 1	Field 2	Field 3
WRITE TAPE	,cuu	, (volser,name1,name2, . . .)

Tape Name	Format Code	Blocking
Field 4	Field 5	Field 6
, tname	, nx	, (rl, bl)

Field	Contents
1	WRITE or TAPE
2	cuu is the device address of a tape unit.
3	volser is the volume serial number of the magnetic tape reel desired. It can be up to 6 alphanumeric characters. This name will appear in a message to the computer operator instructing him to mount the tape. name1, name2, ... are names of data sets which are specified in Field 4 of EXEC control statements, and are copied to the output tape following execution of the program named in the EXEC statement.
4	tname is a symbolic name assigned to the output tape. It may be from one to six characters in length. The first character must be alphabetic.

If Field 4 is defaulted, the system will assign the tape name OUT to the output tape. When using one or more WRITE control statements, caution should be observed to prevent dual specification of the system assigned name OUT.

5-6 Same as READ card.

The TAPE control statement may be used for tapes that are both output and input. Its format is the same as the WRITE control statement.

2.3.7 EXEC, E

The EXEC control statement is used to specify a task or a sequence of tasks. It is also used to specify the input and output data sets as well as the parameters required for the specified tasks.

Operation	Program Name
Field 1	Field 2
[*] {EXEC E}	,pname

INPUT DATA SETS
Field 3
, { * (name1, name2, . . .) (tname/n1-n2,n3, . . .) (tname/n1-n2,n3,name1,name2, . . .) (*tname/n1-n2,n3,name1,name2, . . .) }

OUTPUT DATA SETS
Field 4
, { * (name1,name2, . . .) }

OUTPUT SIZE PARAMETERS
Field 5
, [{ (sl, ss, nl, ns) (SL=sl, SS=ss, NL=nl, NS=ns) }]

OPTIONAL PARAMETERS
Field 6
, [(param1, param2, . . .)]

Field	Contents
1	EXEC or E or *EXEC or *E. The optional asterisk (*) in this field indicates that the task specified is one of a sequence of tasks in a "DO group", and is not the last task in the sequence. See paragraph 3.8 for a discussion of VICAR DO groups.
2	pname is the name of the program to be executed. The name is 1 to 8 alphanumeric characters, beginning with an alphabetic. The program must exist in the VICAR program library. If the first character in the program name is the character 'V', automatic label processinq is suppressed, and the system will not write a system or user labels on any of the output data sets. In this case, the program itself must write any required system or user labels. Such programs are normally used to process data sets or tapes which are not in any of the standard formats.
3	As shown, there are several optional combinations which may be used to define the input data set field. The number and function of the input data sets is program-dependent. * A single asterisk is used to indicate that the task has no input data sets.

Field	Contents
	name1,name2, . . . are names of data sets and/or tape files to be used as parallel input data sets for the task. If a tape name is used without a file number, VICAR will assume the next sequential file following the last one processed on that tape.
	tname/n1-n2,n3, . . . A tape name followed by a slash '/' and a set of file sequence numbers, is used to specify multiple tasks, one for each tape file.
	tname/n1-n2,n3,name1,name2, . . . When multiple tasks are specified, Field 3 may also contain one or more data set names. For each task generated, the named data sets will be used as parallel inputs.

Note: For the preceding Field 3 formats, specification of a tape file name (with the file number included) will cause VICAR to generate a utility task which will copy the tape file to direct access storage, performing any required format conversion. An un-named disk data set of adequate size must be allocated using a RESERVE or equivalent statement, or the job will fail.

*tname/n1-n2,n3,name1,name2, . . . An asterisk may precede a tape name or a tape file name. This will cause the system to position the tape at the specified file and allow the task to read the tape directly.

Since most VICAR programs are designed to read only the standard VICAR format, a prerequisite for the use of the direct tape option is that the tape format is logically equivalent to the format used for disk data sets (5, 6, 8 or 9). A maximum of 10 input data sets/tape files may be specified.

4	* A single asterisk is used to indicate that the task has no output data sets.
	name1,name2, . . . Up to four output data sets or tape names may be specified for a task. The number and function of the data sets is program-dependent.
5	(sl, ss, nl, ns) The output size parameters are optional. If required, all four parameters must be coded as shown. Each parameter is a positive integer from 1 to 9999.

Field	Contents
	(SL=sl, SS=ss, NL=nl, NS=ns) Any or all of the size parameters may also be specified in keyword format as shown. When keywords are used, any or all four parameters may be coded, and the order is of no significance.
	nl and ns (number of lines and number of samples) are the number of data records (picture lines) and the number of bytes/record to be processed from the input data set. sl and ss (starting line and starting sample) are the starting data record and starting byte in the input picture at which processing is to begin. These parameters allow the user to specify that a rectangular sub-segment of the input picture is to be processed. The exact interpretation of these parameters is determined by the application program, and can differ from that given here.

If field 5 is defaulted, the system will assign values of 1 to SL and SS. Values for nl and ns will be assigned equal to the values of these parameters in the primary input data set system label record. In this way, the output data set will be equal in size to the input data set.

Note that tape files which do not have a VICAR label are copied to disk storage and that all input data sets processed by a standard program have the required VICAR label.

6	(param1,param2, . . .) are names of parameter sets as defined in PARAMS control statements. Optionally, all of the types of parameters permitted on parameter cards may be coded providing that each individual parameter is equal to or less than eight characters.

Note that the maximum number of subfields within a field is limited to 10. Thus, a maximum of 10 parameters may be coded in field 6. If the number of parameters is greater than 10, one or more parameter sets delimited by a PARAMS control statement must be used.

2.3.8 PARAMS, P

The PARAMS control statement is used to define a symbolic name for a set of parameter cards.

Operation	Parameter Set Name
Field 1	Field 2
{PARAMS P}	,psname

Field	Contents
1	PARAMS or P
2	psname is a symbolic name of from one to eight characters.

Parameter Cards

Parameters may start in column 1 and cannot extend past column 71. Blanks or commas are used as separators between parameters. Blanks and commas may be used in literal parameters which are enclosed between apostrophes.

Parameters may be continued on as many cards as required. No continuation character should be used in column 72. A parameter cannot be split between cards. Parameters may be integer, real, alphanumeric, hexadecimal, or literal, as listed:

2.3.8.1 Integer Parameters--Positive and negative integer parameters may consist of an optional sign digit and up to seven decimal digits.

2.3.8.2 Real Parameters--Positive and negative real parameters may be represented in the format $\pm$ n1 n2 ---$n_m.n_p$---n_rE $\pm$ M_1 M_2 (M_1 and M_2 are both required)

or

$\pm$ n1 n2 --- $n_m.n_p$ ---n_r

2.3.8.3 Alphanumeric Parameters--A parameter containing one or more alphabetic characters is padded with blanks on the right, or truncated, to exactly 8 characters.

2.3.8.4 Hexadecimal Parameters--are represented by an X and one to eight hexadecimal digits enclosed within apostrophes. The parameter is stored in a four-byte field padded with zeros on the left if less than eight digits.

2.3.8.5 Literal Parameters--A character string of any length up to 69, including numbers, blanks, and special characters, may be enclosed within apostrophes. The character string, less apostrophes, will be padded on the right if necessary to a length which is a multiple of 4.

2.3.9 LABEL, L

The LABEL control statement is used to add a label of arbitrary text up to 68 characters to an output data set of an EXEC statement. It follows the EXEC statement to which it applies.

Operation	Data Set Name	Text
Field 1	Field 2	Field 3
$\begin{Bmatrix}\text{LABEL}\\ \text{L}\end{Bmatrix}$	$,\left[\begin{Bmatrix}\text{name}\\ *\end{Bmatrix}\right]$	,text

Field	Contents
1	LABEL or L
2	name is a one to eight character symbolic name which is used to add more than one label record to a specific data set. * A single asterisk is used to indicate that the label record is to be added to the output data sets for each task where multiple tasks are specified.

Further information on the use of the LABEL control statement is contained in Section 3.

Field	Contents
3	text may consist of from one to 68 characters. Text may not extend beyond column 71. LABEL control statements cannot be continued.

2.3.10 RELABEL, R

The RELABEL control statement is used to delete all existing labels and add a new label record to an output data set.

Operation	Data Set Name	Text
Field 1	Field 2	Field 3
{RELABEL R}	,[{name *}]	,text

Field	Contents
1	RELABEL or R
2	This field is identical to Field 2 in the LABEL control statement.
3	text may consist of from one to 68 characters. Text may not extend beyond column 71. RELABEL control statements cannot be continued.

2.3.11 NOTE

The NOTE control statement is used to print a message on the output listing. It is also used to control certain VICAR functions.

Operation	Text
Field 1	Field 2
NOTE	,text

Field	Contents
1	NOTE
2	text may consist of one to 64 characters.

Certain text strings are significant to the VICAR system, as follows:

NOTE, CONTINUE causes the system to continue executing tasks, even if a subsequent task terminates abnormally.

NOTE, ABORT causes the job to be terminated if a subsequent task terminates abnormally.

NOTE, ABORT and NOTE, CONTINUE may be inserted anywhere and repeatedly as necessary in the sequence of EXEC statements to control VICAR response to abnormal task terminations. At the beginning of the job ABORT is in effect.

NOTE, WTO causes a message to be displayed to the computer operator at the beginning of each task. The message includes the program and job names. This feature allows a user to follow the progress of a VICAR job in execution if he has access to a system console.

2.3.12 END

The END control statement is used as a delimiter to indicate that all control statements have been read.

Operation
Field 1
END

Field	Contents
1	END

(No additional fields are required.)

The END statement is optional.

2.3.13 TIME

The TIME control statement is used to set the maximum CPU time allowed for the second of the pair of VICAR jobs. This statement has meaning only for systems with job step timing. If no TIME statement appears, VICAR assumes 'TIME,1439' was specified.

Operation	Maximum Time
Field 1	Field 2
TIME	,time

Field	Contents
1	TIME
2	time is an integer giving the number of minutes set as the time limit for the second VICAR job. In systems with job step timing the job will be aborted (ABEND code S322) after this interval.

2.3.14 REGION

The REGION control statement is used to set the size of the region of main storage to be used by the second of the pair of VICAR jobs. The effect of exceeding the specified region depends on the version of the operating system in use. In MVT systems, the job will be aborted. If no REGION statement appears, a region size of 150K is assumed by VICAR.

Operation	Maximum Storage
Field 1	Field 2
REGION	,size

Field	Contents
1	REGION
2	size is the requested region size in units of 1024 bytes, specified as an integer followed by the letter K.

2.4 Restrictions

Based upon the limits of available main storage, there are certain "implementation" restrictions which must be observed.

Number of Parameter Sets

The total number of parameter sets delimited by PARAMS control statements is limited to 200.

Number of Parameter Cards

The total number of parameter cards contained in all parameter sets delimited by PARAMS control statements is limited to 500.

Number of Control Statements within a DO-GROUP

A pseudo DO-LOOP, herein referred to as a DO-GROUP may be used to specify the application of several programs to a number of files without repeating the EXEC control statements. Use of the DO-GROUP is described in paragraph 3.8.

The total number of cards, both EXEC and LABEL control statements, contained within a DO-GROUP is limited to 40.

Number of System Units

The total number of disk data sets and tape units which may be referenced within a single job is limited to 32 or less. If the total number of such system units, as specified by a combination of RESERVE, SAVE, READ, WRITE, RELEASE, and FIND control statements, is greater than 32, a diagnostic message will be printed and the job will be terminated.

2.5 Pre-Processor Statements (EVIL)

Pre-processor statements are those which are detected and processed by the preprocessor step of the VICAR procedure. There is little significance to the separate processing except that implementation was simplified. However, the general syntax rules of the pre-processor statements are not always the same as for standard statements so they are described separately.

For historical reasons, the pre-processor is sometimes called EVIL, and pre-processor statements are called EVIL statements.

The preprocessor statements constitute a general macro capability within VICAR. It has found a great deal of use in repetitive processing of large amounts of data using similar or identical algorithms.

2.5.1 DEFINE, D

The DEFINE control statement introduces a body of VICAR control statements which are to constitute a procedure or macro. The statement names the procedure, and defines zero or more substitutable character-string parameters. The body of the procedure consists of any number of VICAR statements following the DEFINE statement. The procedure definition is terminated by an END statement or another DEFINE statement. Therefore, END or DEFINE statements may not appear in the procedure body. Except for the search for DEFINE or END statement, the procedure body is not scanned for validity at the time it is defined. Thus, text which would not constitute a valid VICAR statement may appear in a procedure.

Operation	Procedure Name	Argument	Argument	...
Field 1	Field 2	Field 3	Field 4	...
{DEFINE D}	,procname	[,argument 1]	[,argument 2]	...

Field	Contents
1	DEFINE or D
2	procname is the name of the procedure being defined. Any character string is valid.
3,4,···	argument 1, argument 2,··· are any number of character string arguments. The arguments should appear in the body of the procedure, and will be replaced by a character string parameter in a subsequent CALL statement.

2.5.2 END

The END control statement is used as a delimiter to terminate a procedure definition.

Operation
Field 1
END

Field	Contents
1	END

This statement is syntactically identical with the END statement (2.3.12) which terminates a VICAR job definition. The effect of the END statement depends on whether a procedure is being defined.

2.5.3 CALL, C

The CALL statement is used to invoke a procedure which has been previously defined using the DEFINE statement. All statements from the procedure body will be copied into the job stream, and substitutions of character-string parameters will be made. CALL statements may be nested and recursive.

Operation	Procedure	Parameter	Parameter	...
Field 1	Field 2	Field 3	Field 4	...
|CALL| |C|	,procname	[,parameter 1]	[,parameter 2]	...

Field	Contents
1	CALL or C
2	procname is the name of a procedure which must be previously defined. However, the effect of invoking an undefined procedure is specifically defined to be null. No error messages are given in this event.

Field	Contents
3,4···	parameter 1, parameter 2,··· are character string parameters which are to be substituted for corresponding arguments in the DEFINE statement. That is parameter 1 is substituted for argument 1, parameter 2 is substituted for argument 2, etc. If the number of parameters does not match the number of arguments, excess parameters will be ignored, while insufficient parameters will be assumed to have null value. Virtually any character string not containing a comma is a valid parameter. Parameter strings are delimited by the surrounding commas, and may therefore contain leading, trailing or embedded blanks, except that the last one may not have trailing blanks. Null strings are valid parameters.

2.5.4 GET, G

The GET control statement is used to access one or more sequences of statements stored on the VICAR procedure library. These stored sequences normally, but need not, consist of one or more procedures headed by DEFINE statements. It is important to understand that the GET statement does not invoke a procedure; it merely retrieves text from the library. A subsequent CALL statement must be used to invoke a procedure which is retrieved by a GET statement. It is also important to understand that the name under which the text is stored in the library is normally, but need not be, the same as the name of the procedure which the text defines.

Operation	Text Name	Text Name	···
Field 1	Field 2	Field 3	···
GET G	,textname 1	[,textname 2]	···

Field	Contents
1	GET or G
2,3,···	textname 1, textname 2,··· are names of text files in the VICAR procedure library. Normally the text name is the same as the name of procedure defined in the text. Each textname must have been previously stored in the library.

2.6 Post-Processor Statement

In a large scale computer environment, the VICAR restriction on 8-character names may become a severe burden. The optional post-processor allows data set names up to the length limited by the operating system (44 characters). Within the VICAR control statements the user employs a "dummy" name conforming to the VICAR restriction of 8 characters. Post-processor control statements allow the dummy name to be changed to the actual name desired. The syntax of the post-processor control statement is:

oldname = newname

where oldname is the dummy name conforming to the VICAR 8-character restriction, and appearing on a SAVE, FIND or RELEASE VICAR statement, and newname is the desired name conforming to operating system restrictions. The equal sign is required, and all blanks will be ignored. Post-processor control statements can not be mixed with standard VICAR statements. They must be proceded by special job control statements as shown in Section 3.

3.0 USAGE

3.1 VICAR Job Set-Up

A VICAR job is normally initiated by submitting to the computer a card deck containing the VICAR control statements, and the job control statements which cause the operating system to give control to VICAR. The VICAR job has the following form.

```
//jobname    JOB    . . .

//           EXEC  | VICAR    | [,DISP=SHR]
                   | VICARDSN |

#<EVIL2>
                :
                :

         VICAR control statements (including EVIL)
                :
                :

//VTR2.SYSIN DD  *
                :
                :

         post-processor statements
                :
                :
```

An optional job termination card with // in columns 1-2 and blanks in columns 3-72 may be used at the end. The "//VTR2...." card and post-processor statements may be included only if VICARDSN is used. The optional parameter DISP=SHR, if used, allows permanently allocated data sets to be processed by two or more VICAR jobs concurrently. Normally only reading of those data sets should be done in such jobs.

The JCL statements are processed by the operating system job control processor and invoke a cataloged procedure, VICAR or VICARDSN. These procedures contain job control statements which define necessary data sets and execute the VICAR system programs.

The cataloged procedure first invokes the TTM program which reads the #<EVIL2> statement. This statement accesses a TTM procedure from the TTM library which in turn performs the pre-processor scan, processing the specific pre-processor statements, and copying the others unchanged.

VTRAN will then proceed to read the VICAR control statements in the job deck. All VICAR control statements are listed as well as written into a direct access data set. VTRAN will then read the VICAR control statements from direct access storage and generate the needed job control and task queue statements.

3.2 The RESERVE Control Statement

The user must include one or more RESERVE control statements if it is necessary to allocate direct access storage space. BLOCK, B, and A are all synonyms for RESERVE. A formal description of the fields in the RESERVE control statement is presented in paragraph 2.3.1. The number and attributes of the disk data sets which are to be reserved will depend upon the specific requirements of the job.

A data set is thought of as a place where a picture can be stored. When first allocated, the data set is "empty". A picture is stored in the data set by an application program as the result of an EXEC statement. When a picture is stored in a data set which already contained a picture, the new picture replaces the old picture; the old picture is lost. The user's main concern when allocating a data set is its size (number of records, and number of bytes/record). The user must know the sizes, or at least an upper bound on the sizes, of the pictures with which he is working. Any picture which is stored in a data set should not exceed the size of the data set. If the user tries to store a picture in a data set which is too small, the job will usually be aborted. (If only the number of records is too small, the operating system will attempt to allocate additional disk space for the data set equal to 20% of the original number of records. If the attempt is successful, the job will continue, but may be using disk space wastefully.)

Normally the user will not specify the data set block size, but will allow VICAR to calculate it. However, he should be aware that, usually, the larger the block size, the more main storage the programs which process the data set require.

Some examples follow.

RESERVE , 5 , 1024 , 500 , SPOOL1

This statement will reserve five data sets on disk volume SPOOL1. Each data set will consist of five-hundred records of 1024 bytes each. Optional names have not been specified and the system will therefore assume that these data sets are scratch and available for assignment by VICAR if required.

RESERVE , 3 , (600 , 7200) , 700 , * , (A , B , C)

This statement will reserve three data sets on disk volumes assigned by the operating system. Each data set will consist of 700 records of 600 bytes each. The records will be grouped into blocks of 7200 bytes each. The data sets will be named A, B and C. The explicit assignment of optional names will cause the system to treat these data sets differently than the scratch data sets in the previous example. To modify a named data set, the programmer must specify the data set name in the output data set field (field 4) of an EXEC control statement.

RESERVE , 2 , 1024 , 1024 , IPL304 , (X04 , Y07)

This statement will reserve two data sets on disk volume IPL304. The data sets will be named X04 and Y07. The names consist of the tape names X and Y with a two-digit file sequence number appended thereto. Again, the system will treat these data sets differently than scratch data sets. If a tape-file name is specified in the input data set field (field 3) of an EXEC control statement, the system will generate a utility task to copy the specified tape-file into the reserved data set, where it will remain for the remainder of the job.

3.3 The SAVE Control Statement

The user may include one or more SAVE control statements to preserve disk data sets created in this job for a subsequent job. An example follows:

SAVE , 3 , 600 , 700 , IPL302 , (A , B , C)

The above statement will reserve three data sets, named A, B and C, on disk volume IPL302. All three data sets will be preserved for use in a subsequent job. A second example is:

SAVE , 1 , (1000, 7000) , 800 , * , JMS.G1

The above statement will allocate a permanent data set named JMS.G1 on a storage disk in the SYSDA pool. The data set will be cataloged in the operating system catalog. The data set will have 800 records of 1000 bytes each. There will be 7 records per physical block.

3.4 The FIND Control Statement

The user may include one or more FIND control statements to access disk data sets created in a previous job.

An example follows:

```
FIND , ( A , IPLSYS, JMS.G1 , * )
```

Data set A on disk volume IPLSYS and cataloged data set JMS.G1 are assumed to have been created in a previous job. The above statement will cause the system to access these data sets and will enable the user to reference the data sets in EXEC statements by their symbolic names.

3.5 The RELEASE Control Statement

The user may include one or more RELEASE control statements to delete disk data sets created in a previous job. An example follows:

```
RELEASE , ( A , IPLSYS , C , IPLSYS )
```

The above statement will cause the system to access data sets A and C on disk volume IPLSYS and will enable the user to reference the data sets in EXEC statements by their symbolic names. The data sets will be deleted at the end of the job.

3.6 The READ Control Statement

The user must include a READ control statement for each input tape. The READ control statement is used to define the device, data format, and to assign a symbolic name to the tape. An example follows:

```
READ , * , SCR001 , X , 8F
```

The above statement will cause VICAR to access tape SCR001 and assign the symbolic name X to this tape. In addition, VICAR will record the format of this tape as 8F. A second example follows:

```
READ , 71 , SAVE1 , Y , 8

READ , 71 , SAVE2 , Y , 5A , (510, 10400)
```

The first statement will cause VICAR to access tape SAVE1 and assign the symbolic name Y to this tape. VICAR will record the format of this tape as 8 (the default format). The second statement will cause VICAR to access tape SAVE2 and assign the symbolic name Z to this tape. In addition, VICAR will record the format of this tape as 5A and will record a record length of 510 bytes and a block size of 10400 bytes. Since the same symbolic device name was used in field 2 of both statements, the same physical tape drive will be used for both tapes. This of course implies both tapes are not needed simultaneously for the same task. (The VICAR program VMOUNT provides an alternate method of sharing a tape drive among several tapes if the tape formats are identical.)

3.7 The WRITE Control Statement

The user must include a WRITE control statement for each output tape. The WRITE control statement is used to define the device, data format, and to assign a symbolic name to the tape. In addition, the WRITE control statement may be used to specify symbolic names for disk data sets which are to be copied to the tape. The keyword TAPE is equivalent to WRITE.

An example follows:

```
WRITE , * , ( IPSZ81, C , D , E , F ) , Z , 8F
```

This statement will cause VICAR to access tape IPSZ81 to assign the symbolic name Z to this tape. The system will record the format of this tape as 8F. In addition, the data set names C, D, E and F will cause VICAR to create a utility task to copy each of these data sets to the output tape. These utility tasks will be created each and every time that any of the above symbolic names are specified in the output data set field of an EXEC control statement.

A second example follows:

```
WRITE , * , ( JBS005, AA , DD ) , Z , 8F

EXEC , FILTER , ( X / 1-10 ) , AA , , P1
```

The above EXEC control statement will create 10 tasks, each of which will filter one frame. The output of each task will be written in disk data set AA. Specification of the data set name AA in the WRITE control statement will cause the system to create a utility task immediately after each of the filter tasks, copying data set AA to tape JBS005 in format 8F. The format of the disk data set AA is equivalent to format 8 (the standard format). The utility task, therefore, will perform the necessary conversion to produce an output tape file with format 8F.

3.8 The EXEC Control Statement

The user must include EXEC control statements to define the image processing tasks desired.

A single EXEC control statement may be used to define one task or a sequence of tasks. In addition, the EXEC control statement is also used to specify input and output data sets, size parameters, and other variable parameters required for the specified tasks. The following paragraphs will explain the effect of the various options available with the EXEC control statement.

A formal description of the fields in the EXEC control statement is presented in paragraph 2.3.7.

The EXEC control statement has a variety of options. The following paragraphs illustrate the use of these options and defines the system response thereto. An example follows:

```
EXEC,  GEN, * , AA , (NL=300 , NS=400)
```

This statement will generate a single task using the program GEN. The asterisk in the input data set field (field 3) specifies that the task has no input data sets. The output data set is named AA and the size is specified as 400 samples by 300 lines.

If the first character in the program name is the character 'V', automatic label processing is suppressed. The system will not write a system or user labels on any of the output data sets. In this case, the program itself must write any required system or user labels.

Another example follows:

```
EXEC ,  LIST , AA , * ,, PLIST
```

This statement again will generate a single task using the program LIST. The asterisk in the output data set field (field 4) specifies that the task has no tape or disk output data sets.

Another example follows:

```
EXEC,PICAVE,(A,B,C),AA
```

This statement will also generate a single task using the program PICAVE. The task will have three parallel input data sets, A, B, and C. The output data set is AA.

Data set A (the first data set in the input data set field) is the primary input data set. The system will copy the labels from the primary input data set to A to the output data set AA, while adding a "history label" containing the program name, PICAVE.

Another example follows:

```
READ , * , XYZABC , X , 8F
A , 1 , 1000 , 1000 , *
EXEC , FILTER , ( X/1-5,7-10,12) , A , , WEIGHTS
```

The above statements will generate ten tasks using the program FILTER. The input data sets for each task will be X01--X05, X07--X10, and X12. These input data sets may be direct access data sets or they may be tape files. Note, the system appends a two-digit sequence number to the name (X). In general, the above configuration is employed with tape files. As shown, a READ control statement with the tape name X is required. VICAR will also automatically generate a utility task prior to each FILTER task. This utility task will copy the specified tape file to disk, performing the necessary format conversion.

The programmer must reserve disk data sets which the system can use. For the above example, the programmer has reserved one scratch (unnamed) data set which will be reused by the system. Another alternative would be to reserve named data sets with names matching the above files (ie. X01, X02, X03, etc.). The system will always direct the utility task output to an appropriately named data set, if available.

Another example follows:

```
READ , * , SCR001 , X
READ , * , SCR002 , Y
B , 2 , 500 , 1000 , * , A
EXEC , ICOR , ( *Y / 1-20 , X03 ) , A , , ICORPAR
```

The above statements will generate 20 tasks using the program ICOR. The asterisk preceding the tapename Y indicates the direct tape input option. VICAR will link the primary input data set for the ICOR program to tape SCR002. On the other hand, the secondary input data set (tape file X03) is not preceded by an asterisk. The system will, therefore, generate a utility task to copy file 3 from tape SCR001 into a disk data set using one of the data sets allocated in the "B" statement. Once this file is on disk, it will then be available for each of the 20 ICOR tasks.

To use a tape file directly, without copying it to disk, its format must be exactly equivalent to that of a disk file. As shown above, the formats on the READ control statements have been defaulted indicating Format 8. Since both tapes are Format 8, it would have been possible to include an asterisk prior to X03 in the input data set field. However, considering tape rewind time, it might be preferable to have the secondary input file X03 on disk, since it is used in each of the 20 ICOR executions.

Another example follows:

```
EXEC , STRETCH , A , B , , ( LINEAR , 98 , 371 )
```

The above statement shows how the user may include parameters in field 6 of the EXEC control statement. In this case, the restriction is that the field may contain up to 10 subfields separated by commas, and each subfield may contain a maximum of eight characters.

Alternatively, one or more symbolic parameter set names may be included in field 6 of the EXEC control statement. In this case, the symbolic name must be specified in a PARAMS control statement followed by one or more parameter cards. (See paragraph 3.9.) The system will incorporate the specified parameter sets into the task queue.

An example follows:

```
PARAMS , P2

555 666 777 989.05

PARAMS , P4

' THIS IS A SPECIAL TITLE FOR A SPECIAL TASK '

EXEC , PROGA , A , B , , ( P2 , P4 )
```

As shown, two parameter sets, P2 and P4 are specified in the above EXEC control statement. The parameters will be presented to the program in the order specified, P2 followed by P4.

Remember that each program has unique requirements for the parameters which may or must be specified, as well as the number of input and output data sets and their data content. Programs may have restrictions on the size of pictures which can be processed. This information must be obtained from the user guide for the individual program.

The EXEC control statement is normally used to specify one task, or a sequence of tasks where the same program is applied to a number of files. It is sometimes necessary to apply a sequence of programs to each of a number of files. To accomplish this, the programmer must code an asterisk prior to the keyword EXEC in the first EXEC control statement of a pseudo-DO LOOP. This statement must specify a set of files in its primary input data set field. All subsequent tasks in the DO group except the last must also have an asterisk coded prior to the EXEC keyword. The last task in the DO group is indicated by the absence of the asterisk.

An example follows:

```
*EXEC , ICOR , (Y/1-14 , X01), A , , P1

*EXEC , FILTER , A , B

*EXEC , LIST , B , * , , P 2

EXEC , MASK , B , Z , , P 3
```

The above example illustrates the EXEC control statements required to sequence fourteen selected files through 4 programs. As shown, each ICOR task has a secondary input data set X01, the calibration input. Also, the output of ICOR, data set A is the input to the FILTER tasks. The output of FILTER, data set B is the input to both LIST and MASK.

3.9 The PARAMS Control Statement

The user may include PARAMS control statements to define symbolic names for sets of parameter cards.

An example follows:

```
PARAMS, P1

555 666 777 888

21 22 23 24 25 26

PARAMS , P2

222 333 444 555

31 32 33 34 35 36

EXEC , PROGA , A , B , , P1

EXEC , PROGA , A , B , , P2
```

In the above example, the first EXEC control statement includes the symbolic name P1 in the parameter field, while the second EXEC control statement contains P2.

The system, upon recognizing that P1 and P2 have been specified in PARAMS control statements, will incorporate the proper parameter set for each task.

Parameter cards are described in paragraph 2.3.8. In general, parameters are free-form and blanks or commas may be used as separators between parameters. Blanks and commas may be used in literal parameters which are enclosed between apostrophes. Note the slight difference in parameter format from those in EXEC statements, which allows a maximum of 10 parameters, each of 8 characters or less, and separated by commas.

Some examples of valid parameters are as follows:

Integer: -47 385 25 +377 -2880

Real: +244.07 -377.002 -980.0E-02

Alphanumeric: ADD999 A4725 888ABC COPY

Hexadecimal: X'FF040402' X'07'

Literal: ' THIS MAY BE ANY CHARACTER STRING 123456789,$$$, '

A set of parameters is terminated by any statement which VICAR recognizes as a VICAR statement. There is an inherent ambiguity in this situation, which can cause trouble if the first parameter on a card is the same as a legal VICAR verb.

Users should avoid having the first parameter on a card match a legal VICAR verb; since many keyword parameters are order independent, this can usually be accomplished. (Application programmers should avoid designing programs with legal parameters that can be mistaken for VICAR verbs.)

3.10 The LABEL Control Statement

The user may include LABEL control statements to add output label records to the output data sets. L is equivalent to LABEL.

In general, LABEL control statements follow the EXEC control statement with which they are associated. However, in the case where several tasks are generated by a single EXEC control statement, certain restrictions must be observed.

An example follows:

```
EXEC , PROGA, A , B , , P1

LABEL , B , THIS IS LABEL RECORD NUMBER ONE

LABEL , B , THIS IS LABEL RECORD NUMBER TWO

LABEL , B , THIS IS LABEL RECORD NUMBER THREE
```

Since the above EXEC control statement specifies only a single task, all of the above label records are added to the label set of data set B.

Another example follows:

```
EXEC , PROGA , ( TX / 1-15 ) , A , , P1

LABEL , * , LABEL RECORD FOR ALL 15 FILES
```

The asterisk in the second field of the label statement causes VICAR to add the same label to the image written on data set A by each of the 15 tasks. In the case where an EXEC control statement specifies several tasks, LABEL control statements may be included to add a unique label record to each specified file. The LABEL control statements in a DO group must follow the first EXEC control statement in the DO-group. For example:

```
*EXEC , PROGA , ( TX / 2-5 ) , A
LABEL,, THIS IS THE LABEL RECORD FOR FILE 2
LABEL,, THIS IS THE LABEL RECORD FOR FILE 3
LABEL,, THIS IS THE LABEL RECORD FOR FILE 4
LABEL,, THIS IS THE LABEL RECORD FOR FILE 5
*EXEC , PROGB , A , B
*EXEC , PROGC , B , C
*EXEC , PROGD , C , D
*EXEC , PROGE , D , E
EXEC , PROGF , E , Y
```

As shown, the above label statements will generate a label for each file. Note that the LABEL control statements within the DO-group follow the first EXEC control statement. In the present version of VICAR, LABEL control statements may not be used after subsequent EXEC control statements within a DO-group.

3.11 The RELABEL Control Statement

The user may include RELABEL control statements to add a label to the output data sets while deleting all the existing labels.

Use and placement of the RELABEL control statements is exactly the same as for the LABEL control statement described in paragraph 3.10.

3.12 The NOTE Control Statement

The user may include NOTE control statements to print a message on the output listing.

In addition, the NOTE control statement with certain keywords will cause specific system actions. For example, ABORT will cause VICAR to respond to a deliberate abnormal end by a subsequent task with job termination and a dump if requested.

Two examples follow:

```
NOTE , ABORT
```

```
NOTE, THIS IS A MESSAGE FOR THE OUTPUT LISTING.
```

3.13 The END Control Statement

The user may include an END control statement to indicate that all control statements have been read. The END statement is optional.

3.14 The TIME Control Statement

The user may include a TIME control statement to limit the CPU time which will be used in the second VICAR job. In systems with job step timing the job will be aborted if the time is exceeded. At most one TIME statement may appear in a VICAR job. If none appears, a time limit of 1439 minutes is used by VICAR.

An example follows:

```
TIME , 30
```

In this example, the job will be aborted when it has used more than 30 minutes of CPU time.

3.15 The REGION Control Statement

The programmer may include a REGION control statement to limit the size of the main storage region which will be used by the second VICAR job. The job may be aborted when it attempts to use more than this amount of storage, depending on the version of the operating system. At most one REGION statement may appear in a VICAR job. If none appears, a size of 150K is assumed.

An example follows:

```
REGION , 200K
```

In this example, a region size of 200K is being requested.

It is extremely difficult to predict the region size which will be needed for a given execution of a given VICAR program. It depends on many variables, some of which are beyond the control of the programmer, such as data set block sizes and operating system resident routines. As a rule of thumb, 150K will usually be adequate. An increased region size is indicated if a VICAR job abends with a user code of 69 or a system code indicating insufficient storage.

3.16 The DEFINE Control Statement

The user may include DEFINE control statements to introduce and name a procedure, or macro, consisting of any VICAR statements except DEFINE and END. The DEFINE statement may also list character string parameters which are to be replaced when the procedure is invoked by a subsequent CALL statement. Procedure definitions must precede procedure invocation. Once defined, the procedure definition is effective until re-definition of the same procedure name in another DEFINE statement, or until the end of the job.

An example follows:

```
D,STRCLIP,FILE,BITS
E,SAR, (*X/FILE),A
E,STRETCH,A,B,, (CLIP,BITS)
D,STRLIN,LOWDN,HIDN,NLX,NSC
E,STRETCH,A,B,, (LINEAR,LOWDN,HIGHDN)
L,,LINEAR(LOWDN-HIGHDN)
E,BOXFLT,B,C,, (NLW,NLX,NSW,NSX)
L,, LOW PASS FILTER (NLX BY NSX)
END
```

Two procedures are defined, one called STRCLIP, and the other called STRLIN. The first is terminated by the DEFINE statement introducing the second. STRCLIP has two arguments, FILE and BITS, and STRLIN has four arguments, LOWDN, HIDN, NLX, and NSC. STRCLIP consists of two EXEC statements which will have the effect of executing the two application programs SAR and STRETCH. The syntax of the "E,SAR,..." is not valid in that the characters following the "/" should be a one or two character number. Also, the STRETCH user guide would show that the parameter following the keyword "CLIP" should be a number. These apparent errors are acceptable provided that when the procedure STRCLIP is invoked by a CALL statement, the arguments FILE and BITS are replaced by numbers.

3.17 The CALL Control Statement

The user may include CALL statements that invoke a procedure and specify its arguments. The procedure must have been previously defined, either by its appearance in the sequence of VICAR control statements, or by being obtained from the procedure library using the GET statement. The effect of the CALL statement is similar to invoking a macro in assembly language. The arguments provided in the CALL statement are substituted for the corresponding (positional) arguments in the procedure definition. The expanded procedure text is then substituted for the CALL statement in the sequence of VICAR statements. Of course the original procedure definition is unchanged and is available for repeated CALLing with different arguments.

If the name of the procedure in the CALL statement has not been previously defined, the effect is to ignore the statement. Even though there is a procedure library, there is no automatic searching of the library. The situation is not considered an error and no diagnostic message is produced. For this reason spelling errors in procedure names can have startling effects which may be hard to diagnose. The user must keep this feature of procedure usage in mind and use appropriate care.

As an example, suppose the example in Section 3.16 has appeared in the VICAR job. The following statements then appear.

```
CALL,STRCLIP,1,3
C,STRLIN,50,150,3,3
```

The effect of these statements is the same as if the following sequence had appeared in their place.

```
E,SAR,(*X/1),A
E,STRETCH,A,B,,(CLIP,3)
E,STRETCH,A,B,,(LINEAR,50,150)
L,,LINEAR(50-150)
E,BOXFLT,B,C,,(NLW,3,NSW,3)
L,,LOWPASS FILTER (3 BY 3)
```

3.18 The GET Control Statement

The user may include GET control statements to retrieve sets of VICAR statements which have previously been saved on the VICAR (EVIL) procedure library. (The process of storing text on the procedure library is not done by VICAR control statements, but by a TTM procedure. It is described in Appendix A.) Because the CALL statement does not cause automatic searching of the library, each procedure CALLed, but not defined in the job, must be explicitly retrieved from the procedure library by means of the GET statement.

Although the procedure library stores arbitrary sets of statements, normal practice is that each set of text constitutes one procedure, beginning with a DEFINE statement and ending with an END statement. Each set of statements stored in the EVIL library is stored under a unique name by which it is retrieved, the name specified in the GET statement. Normal practice is to have the name by which a set of statements is stored equal to the name of the procedure that is defined by that set of statements. Thus to CALL the procedure STRCLIP and STRLIN which have been previously stored on the library, the following statement is used.

```
GET,STRCLIP,STRLIN
```

It is not unreasonable nor uncommon to store under a single name the definitions of two or more procedures which are commonly used together. This simplifies usage by requiring the user to GET only one set of text from the library, even though a number of procedures are to be used.

3.18.1 The DO Library Procedure

The DO procedure provides an iterative capability which is much more powerful than the simple DO-group described in Section 3.8. The DO procedure allows the user to repeatedly invoke any other defined procedure, while the first argument to the other procedure takes on, in sequence, each of a set of values specified. The DO procedure is on the procedure library and may be invoked after it has been retrieved with a GET statement.

The general form of the DO procedure usage is as follows:

```
CALL, DO, procname, arglist, arg2, arg3, ...
```

procname is the name of a previously defined procedure. arg2, arg3, etc are arbitrary character strings not containing commas, arglist is a specification of a list of arguments of the form:

```
arg1/arg1'/arg1''/...
```

If the argument list consists of positive integers in ascending numerical sequence, then

```
arg1/arg1+1/arg1+2/arg1+3/.../arg1'
```

may also be written

```
arg1 - arg1'
```

The alternative forms may be mixed to produce a list of the following or similar form:

```
arg1/arg1' - arg1''...
```

The effect of the DO procedure is to invoke the procedure with name procname a number of times equal to the number of items specified by arglist.

```
CALL,procname,arg1,arg2,arg3,...
CALL,procname,arg1',arg2,arg3,...
CALL,procname,arg1",arg2,arg3,...
           .
           .
           .
```

or

```
CALL,procname,arg1,arg2,arg3,...
CALL,procname,arg1+1,arg2,arg3,...
CALL,procname,arg1+2,arg2,arg3,...
```

The second and subsequent arguments to the specified procedure are always the same, and are arg2, arg3, etc. The first time the specified procedure is CALLed, its first argument is the first item in the list specified by arglist. The second time it is CALLed, its first argument is the second item in the list, and so on until the last item in the list has been used.

As an example, suppose the procedure STRCLIP has been defined as in Section 3.16, and the following statements then appear.

```
G, DO
CALL, DO, STRCLIP, 3-5/7, 3
```

The effect of these statements is as if the following had appeared.

```
E,SAR,(*X/3),A
E,STRETCH,A,B,,(CLIP,3)
E,SAR,(*X/4),A
E,STRETCH,A,B,,(CLIP,3)
E,SAR,(*X/5),A
E,STRETCH,A,B,,(CLIP,3)
E,SAR,(*X/7),A
E,STRETCH,A,B,,(CLIP,3)
```

In this case the same result may also be obtained using the DO-group.

```
*E,SAR,(*X/3-5,7),A
 E,STRETCH,A,B,,(CLIP,3)
```

Consider another example.

```
G,DO
D,STRF,LO
E,STRETCH,A,B,, (LINEAR,LO,150)
E,FOTO,B,*
END
CALL,DO, STRF, 90/100/110
```

These statements are equivalent to the following.

```
E,STRETCH,A,B,,(LINEAR,90,150)
E,FOTO,B,*
E,STRETCH,A,B,,(LINEAR,100,150)
E,FOTO,B,*
E,STRETCH,A,B,,(LINEAR,110,150)
E,FOTO,B,*
```

4.0 EXAMPLE JOBS

This section consists of listings of actual VICAR job decks.

4.1 Example 1

```
//FILTER  JOB  (JBS51,41)
//  EXEC  VICAR
#<EVIL2>
READ,*,T-125,X,8
WRITE,*,SCRXYZ,VFC,8F
BLOCK,2,1000,1000,*,(A,B)
PARAMS,LPAR
NØHIST 101,101,30,30
*E,SAR,(*X/1-3),A
*E,LIST,A,*,,LPAR
*E,FILTER,A,B
*E,LIST,B,*,,LPAR
E,MASK,B,VFC,,COMP
```

Three files on tape T-125 are to be filtered and masked (formatted for film recording). In addition, a portion of each picture is to be listed before and after filtering. The SAR program is used to copy each file from tape to a disk data set.

4.2 Example 2

```
//FFT2TEST JOB (JBS51,41),'TEST VMAST'
// EXEC VICAR
#<EVIL2>
A,1,512,520,IPL306,A
A,1,512,520,IPL304,B
A,1,4096,520 ,IPL304,D
A,1,4096,520 ,IPL306,C
A,1,4096,520 ,LIB001,E
B,1,512,4100,LIB001,FFTI
E,GEN,*,A,(1,1,512,512),(0,0,0)
E,QSAR,A,B,,PQ
P,PQ
1,1,5,10,100
D,PROC,NPOW
E,FFT22,(B,FFTI),D,,(POW,NPCW)
E,FFT2,(B,E    ,C),D,,(FMT,PCW,NPOW)
E,FFT2,(B,E    ,C),D,,(    PCW,NPOW)
END
C,PROC,9
```

5.0 ERRORS

In a system as complex to use as VICAR, error situations occur frequently. This section gives some guidance on what to do if a problem occurs.

Errors may be categorized as occurring during the first job or the second job. Errors which occur during the first job are almost invariably due to incorrect syntax in one or more VICAR statements. These are caught by the program VTRAN during its execution, and a helpful message is printed. Normally the second job is suppressed. Errors which occur during the second job usually result in abnormal termination of the job, accompanied by a user or system "completion code." A user code means the error was detected by the VICAR system, while a system code means the error was detected by the operating system.

5.1 VTRAN Errors (First Job)

The most common kinds of errors occurring during the first VICAR job are given below, along with some suggested actions to take.

PARENTHESIS ERROR
Parentheses are unbalanced.

TOO MANY FIELDS
The maximum number of fields in a VICAR statement is 8. Look for an extra comma.

TOO MANY SUB-FIELDS
The maximum number of sub-fields within any one field is 10. Sub-field delimiters are comma and slash. Look for an extra comma.

ILLEGAL VTRAN CARD
There is an un-classifiable syntax error.

PARAMETER CARD PUNCHED IN COLUMN 72
Column 72 of a parameter card must be blank.

LABEL AT END OF DO-GROUP
LABEL IN MIDDLE OF DO-GROUP
Only the first task of a DO-group may be labeled. Use of preprocessor features may allow the desired processing without using a DO-group.

ILLEGAL SIZE FIELD
Look for an extra or missing comma if the size field looks correct.

NUMBER FIELD ON RESERVE CARD NOT BETWEEN 1 and 9
The field referred to is the second field on a RESERVE, A, B, BLOCK or SAVE statement.

MORE THAN 8 TAPE DATA SETS REQUESTED
The total number of READ, WRITE and TAPE statements may not exceed 8. (Implementation restriction)

TOO MANY PARAMETER SETS SPECIFIED
The total number of PARAMS and P statements may not exceed 200. (Implementation restriction.)

NO DATA SET AVAILABLE FOR OUTPUT
A "*" may have been unintentionally omitted from a tape name in an EXEC statement. The use of the "tape-file no." form of a data set name without a "*" requires the definition of a suitable data set with a RESERVE-type statement.

LABEL CARD ENCOUNTERED UNEXPECTEDLY
The data set name field may have been omitted from a LABEL or RELABEL statement.

5.2 Execution Errors (Second Job)

Errors resulting in abnormal termination during the second job may be either user errors or program errors. Program errors usually must be solved by the application programmer responsible for the program which terminated. User errors can be corrected by changes to VICAR statements in the user's job. Distinguishing between user errors and program errors can be quite difficult, and may ultimately depend on the intentions of the programmer as to how his program should work.

The most common ABEND completion codes associated with VICAR jobs are given below, along with possible user errors. (Since this is not a programmer's guide, the associated possible program errors are not given.)

USER 69
There was insufficient main storage available for buffers. Try a larger region size.

USER 71
An attempt was made to write a record larger than the allocated record size. Be sure the data set record size specified is large enough to accommodate the picture being processed.

USER 72
End of extent on output. Not normally a user error. (This code has not been observed with this version of VICAR.)

USER 73
Write error under unusual conditions. Not normally a user error. (This code has not been observed with this version of VICAR.)

USER 240
This code is produced when the ABEND is intercepted by the Fortran run-time routine. The actual ABEND code is found in another printed message.

USER 324
The application program intentionally terminated abnormally, and "NOTE, ABORT" was in effect. There should be an associated explanatory message produced by the program. The predominant reason is an error in parameters specified for the program. Be sure the spelling of names of parameter sets on the EXEC statement matches the spelling on the PARAMS statement.

USER 999
The requested program was not in the program library or was not executable. Check correct spelling.

USER 1111
The op code in PS44 SVC simulation routine is illegal. This is not a user error. (This code has not been observed with this version of VICAR.)

USER 1112
The VICAR system index for a data set. Be sure all the required data sets have been specified for the program.

USER 1200
The SYSOUT system file cannot be opened. This is not a user error. (This code has not been observed with this version of VICAR.)

USER 1492
The VICAR data set reference number is illegal. This is not normally a user error.

USER 1970
There is an error in the task list passed to the second VICAR job. This would occur if an error occurred in the first job but the second job was not suppressed. Look for a syntax or usage error in the VICAR control statements.

USER 1980
The VICAR system encountered an i/o error in processing parameters or data set labels. Be sure the correct data sets and tapes are specified as input files, that the correct tape format is specified and that the tape does not have a permanent i/o error in the label data.

USER 2400
The maximum number of tape units requested exceeds the implementation limit of 8. This error is normally caught in the first job.

Appendix

A. EVIL2LIB Procedure Library Maintenance Program

To Be Supplied

Appendix

B. JCL Insertions in Second VICAR Job

It is occasionally necessary to modify JCL statements generated by the VICAR system. JCL procedure statements may be overridden using standard operating system methods; see Ref. 1.2.2. These methods cannot be applied to the second VICAR job because the JCL is generated by a program, VTRAN. However, JCL statements can be inserted in the second job in either of two locations, just before and just after the "..EXEC PGM=VMAST" statement. This is accomplished by supplying the statements to be inserted as either of two data sets processed by the first job.

To insert statements before the EXEC statement, the following sequence is used.

```
     //VTR.FT08F001 DD  DATA
                    .
                    .
                    .
     JCL statements to be inserted
                    .
                    .
                    .
     /*
```

To insert statements after the EXEC statement, the following sequence is used

```
     //VTR.FT10F001  DD  DATA
                    .
                    .
                    .
     JCL statements to be inserted
                    .
                    .
                    .
     /*
```

Either of the above sequences is placed in the job deck following the VICAR control statements, and preceding the "// VTR2.SYSIN" statement if any. If both sequences are used, the "FT08" sequence precedes the "FT10" sequence.

As an example, JCL insertions may be used to obtain a core dump after a job ABEND. The following sequence inserted after the VICAR control statements provides a dump.

```
     //VTR.FT10F001  DD  DATA
     //SYSUDUMP      DD  SYSOUT=A
     /*
```

Similarly, a private program library will be searched ahead of the standard library if the following sequence is included.

```
//VTR.FT10F001  DD  DATA
//STEPLIB       DD  DSN=privatelibrary,DISP=SHR
//              DD  DSN=IPL1.SDSRUN,DISP=SHR
/*
```

PUBLICATION 77-37