A GUIDE TO VORTEX OPERATING SYSTEM

CONTENTS

SECTION 1
INTRODUCTION

The Varian Omnitask Real-Time EXecutive (VORTEX) is a modular software operating system for controlling, scheduling, and monitoring tasks in real-time multiprogramming environment. VORTEX also provides for background operations such as compilation, assembly, debugging, or execution of tasks not associated with the real-time functions of the system. Thus, the basic features of VORTEX comprise:

- Real-time I/O processing
- Provision for directly connected interrupts
- Interrupt processing
- Multiprogramming of real-time and background tasks
- Priority task scheduling (clock time or interrupt)
- Load and go (automatic)
- Centralized and device-independent I/O system using logical unit and file names
- Operator communications
- Batch-processing job-control language
- Program overlays
- Background programming aids: FORTRAN and RPG IV compilers, DAS MR assembler, load-module generator, library updating, debugging, and source editor.
- Use of background area when required by foreground tasks
- Disc/drum directories and references
- System generator

SYSTEM REQUIREMENTS

VORTEX requires the following minimum hardware configuration:

a. Varian 70 series or 620/f-100 computer with 24K of main memory

b. Direct memory access (DMA)

c. 33/35 ASR Teletype on a priority interrupt module

d. Real-time clock

e. Memory protection

f. Power failure/restart

g. Optional instruction set

h. Priority Interrupt Module (PIM)

i. Rotating memory device (RMD) on a PIM with either a buffer interlace controller (BIC) or priority memory access (PMA)

j. One of the following on a PIM:
   (1) Card reader with a BIC
   (2) Paper-tape system or a paper-tape reader
   (3) Magnetic-tape unit with a BIC

SYSTEM FLOW AND ORGANIZATION

VORTEX executes foreground and background tasks scheduled by operator requests, interrupts, or other tasks. All tasks are scheduled, activated, and executed by the real-time executive component on a priority basis. Thus, in the VORTEX operating system, each task has a level of priority that determines what will be executed first when two or more tasks come up for execution simultaneously.

The job-control processor component of the VORTEX system manages requests for the scheduling of background tasks.

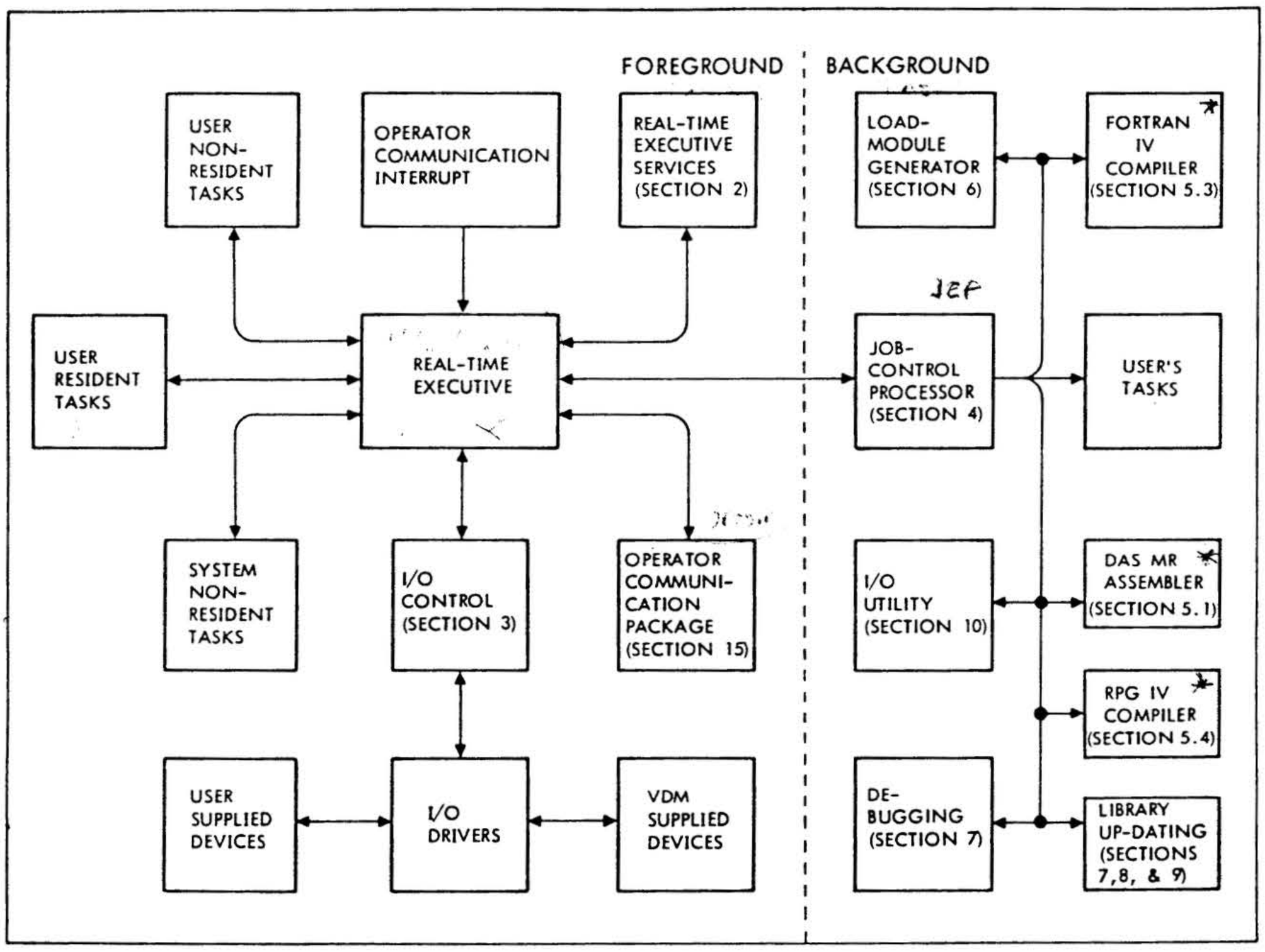
Upon completion of a task, control returns to the real-time executive. In the case of a background task, the real-time executive schedules the job-control processor to determine if there are any further background tasks for execution.

BIBLIOGRAPHY

The following gives the stock numbers of Varian manuals pertinent to the use of VORTEX and the 73/620 computers:

Title	Document Number
V73 Handbook	98 A 9906 01x
620-100 Computer Handbook	98 A 9905 00x
FORTRAN IV Reference	98 A 9902 03x
RPG IV Manual	98 A 9947 03x
VTAM Reference	98 A 9952 22x
Microprogramming Guide	98 A 9906 07x

Where x is a revision level number subject to change.



VORTEX Operating System Flow

SECTION 2
REAL-TIME EXECUTIVE SERVICES

The VORTEX **real-time executive (RTE) component** processes, upon request by a task, operations that the task itself cannot perform, including those involving linkages with other tasks. RTE **service requests** are made by macro calls to V$EXEC, followed by a parameter list that contains the information required to process the request.

The contents of the volatile A and B registers and the setting of the overflow indicator are saved during execution of any RTE macro. After completion of the macro, these values are returned. The contents of the X register are lost.

There are 32 priority levels in the VORTEX system, numbered 0 through 31. Levels 0 and 1 are for background tasks and levels 2 through 31 are for foreground tasks. If a background task is assigned a foreground priority level, or vice versa, the task automatically receives the lowest valid priority level for the correct environment. Lower numbers assign lower priority.

Background and foreground RTE service requests are similar. However, a level 0 background RTE request causes a memory-protection interrupt and the request is checked for validity. If there is an error, the system prints the error message **EX11** with the name of the task and the location of the violation of memory protection. The background task is aborted. RTE service request macros are listed in the table below.

RTE Service Request Macros

Mnemonic	Description	Level 0	FORTRAN
SCHED	Schedule a task	Yes	Yes
SUSPND	Suspend a task	Yes	Yes
RESUME	Resume a task	No	Yes
DELAY	Delay a task	No	Yes
PMSK	Store PIM mask register	No	Yes
TIME	Obtain time of day	Yes	Yes
OVLAY	Load and/or execute an overlay segment	Yes	Yes
ALOC	Allocate a reentrant stack	No	Yes
DEALOC	Deallocate the current re-entrant stack	No	No
EXIT	Exit from a task (upon completion)	Yes	Yes
ABORT	Abort a task	No	Yes
IOLINK	Link background I/O	Yes	No

SECTION 3
INPUT/OUTPUT CONTROL

The VORTEX **input/output-control component (IOC)** processes all requests for I/O to be performed on peripheral devices. The IOC comprises an I/O-request processor, a find-next-request processor, an I/O-error processor, and I/O drivers. The IOC thus provides a common I/O system for the overall VORTEX operating system and eliminates the programmer's need to understand the computer hardware.

The contents of the volatile A and B registers and the setting of the overflow indicator are saved during execution of any IOC macro. After completion of the macro, these data are returned. The contents of the X register are lost.

If a physical-device failure occurs, the I/O drivers perform error recovery as applicable. Where automatic error recovery is possible, the recovery operation is attempted repeatedly until the permissible number of recovery tries has been reached, at which time the I/O driver stores the error status in the user I/O-request block, and the I/O-error processor posts the error on the OC logical unit. The user can then try another physical device or abort the task.

LOGICAL UNITS

A **logical unit** is an I/O device or a partition of a rotating-memory device (RMD). It is referenced by an assigned number or name. The logical unit permits performance of I/O operations that are independent of the physical-device configurations by making possible references to the logical-unit number. The standard interfaces between the program and the IOC, and between the IOC and the I/O driver, permit substitution of peripheral devices in I/O operations without reassembling the program.

VORTEX permits up to 256 logical units. The numbers assigned to the units are determined by their reassignability:

a. *Logical-unit numbers 1-100* are used for units that can be reassigned through the operator communications component (OPCOM, section 15) or the job-control processor (JCP, section 4).

b. *Logical-unit numbers 101-179* are used for units that are not reassignable.

c. *Logical-unit numbers 180-255* are used for units that can be reassigned through OPCOM only.

d. *Logical-unit number 0* indicates a dummy device. The IOC immediately returns control from a dummy device to the user as if a real I/O operation had been completed.

The following table shows the valid logical-unit assignments.

Valid Logical-Unit Assignments

Logical Unit	OC	SI	SO	PI	LO	BI	BO	SS	GO	PO	DI	DO	CU	SW	CL 103	OM 104
Unit No.	1	2	3	4	5	6	7	8	9	10	11	12	101	102	BL 105	FL 106
Device																
Dummy				DUM	DUM	DUM	DUM	DUM		DUM		DUM				
Card punch					CP		CP			CP						
Card reader		CR		CR		CR					CR					
CRT device	CT	CT	CT	CT	CT					CT	CT	CT				
RMD (disc/drum) partition		D		D	D	D	D	D	D	D			D	D	D	D
Line printer					LP					LP		LP				
Magnetic-tape unit		MT		MT	MT	MT	MT	MT	MT	MT						
Paper-tape reader/ punch		PT		PT	PT	PT	PT			PT						
Teletype	TY	TY	TY	TY	TY					TY	TY	TY				
Remote Teletype		TC	TC	TC	TC						TC	TC				

I/O-CONTROL MACROS

I/O requests are written in assembly language programs as I/O macro calls. The DAS MR assembler provides the following I/O macros to perform I/O operations, thus simplifying coding:

- OPEN — Open file
- CLOSE — Close file
- READ — Read one record
- WRITE — Write one record
- REW — Rewind
- WEOF — Write end of file
- SREC — Skip one record
- FUNC — Function
- STAT — Status
- DCB — Generate data control block
- FCB — Generate file control block

The IOC performs a validity check on all I/O requests. It then queues (according to the priority of the requesting task) each valid request to the controller assigned to the specified logical unit. Finally, the IOC schedules the appropriate I/O driver to service the queued request.

The assembler processes the I/O macro to yield a macro expansion comprising data and executable instructions in the form of assembler language statements.

Certain I/O operations require parameters in addition to those in the I/O macro. These parameters are contained in a table, which, according to the operation requested, is called either a file control block (FCB) or a data control block (DCB). Embedded but omitted parameters (e.g., default values) must be indicated by the normal number of commas.

I/O Macros: *The general form of I/O macros* is:

label **name** **cb,lun,***wait,mode*

If the **cb** is for an FCB, it is mandatory. If it is for a DCB, it is optional.

SECTION 4
JOB-CONTROL PROCESSOR

The job-control processor (JCP) is a background task that permits the scheduling of VORTEX system or user tasks for background execution. The JCP also positions devices to required files, and makes logical-unit and I/O-device assignments.

ORGANIZATION

The JCP is scheduled for execution whenever an unsolicited operator key-in request to the OC logical unit has a slash (/) as the first character.

Once initiated, the JCP processes all further JCP directives from the SI logical unit.

If the SI logical unit is a Teletype or a CRT device, the message JC** is output to indicate the SI unit is waiting for JCP input. The operator is prompted every 15 seconds (by a bell for the Teletype or tone for the CRT) until an input is keyed in.

If the SI logical unit is a rotating-memory-device (RMD) partition, the job stream is assumed to comprise unblocked data. In this case, processing the job stream requires an /ASSIGN directive.

A JCP directive has a maximum of 80 characters, beginning with a slash. Directives input on the Teletype are terminated by the carriage return.

JOB-CONTROL PROCESSOR DIRECTIVES

This section describes the JCP directives:

a. Job-initiation/termination directives:

/JOB	Start new job
/ENDJOB	Terminate job in progress
/FINI	Terminate JCP operation
/C	Comment
/MEM	Allocate extra memory for background task

b. I/O-device assignment and control directives:

/ASSIGN	Make logical-unit assignment(s)
/SFILE	Skip file(s) on magnetic-tape unit
/SREC	Skip record(s) on magnetic-tape unit or RMD partition
/WEOF	Write end-of-file mark
/REW	Rewind magnetic-tape unit or RMD partition
/PFILE	Position rotating-memory-unit file
/FORM	Set line count on LO logical unit
/KPMODE	Set keypunch mode

c. Language-processor directives:

/DASMR	Schedule DAS MR assembler
/FORT	Schedule FORTRAN compiler

d. Utility directives:

/CONC	Schedule system-concordance program
/SEDIT	Schedule symbolic source-editor task
/FMAIN	Schedule file-maintenance task
/LMGEN	Schedule load-module generator
/IOUTIL	Schedule I/O-utility processor
/SMAIN	Schedule system-maintenance task

e. Program-loading directives:

/EXEC	Schedule loading and execution of a load-module from the SW unit file
/LOAD	Schedule loading and execution of a user background task
/ALTLIB	Schedule the next background task from the specified logical unit rather than from the background library
/DUMP	Dump background at completion of task execution

JCP directives begin in column 1 and comprise sequences of character strings having no embedded blanks. The character strings are separated by commas (,) or by equal signs (=). The directives are free-form and blanks are permitted between the individual character strings of the directive, i.e., before or after commas (or equal signs). Although not required, a period (.) is a line terminator. Comments can be inserted after a period.

Each JCP directive begins with a slash (/).

The general form of a job-control statement is

/name,*p(1)*,*p(2)*,...,*p(n)*

where

name	is one of the directive names given (any other character string produces an error)
p(n)	is a parameter required by the JCP or by the scheduled task and defined below under the descriptions of the individual directives

Example - Card Input: Assemble a DAS MR program (with source listing and load-and-execute) and generate a concordance listing. The DAS MR program is cataloged on RMD partition D00K under file name USER1 with protection key U. Assign the PI logical unit to RMD partition D00K, open file name USER1 for the assembler, assemble the program, and execute the program with a dump.

```
/JOB,EXAMPLE2
/ASSIGN,PI=D00K
/PFILE,PI,U,USER1
/DASMR,L
/PFILE,SS,,SS
/CONC
/EXEC,D
/ENDJOB
```

SECTION 5
LANGUAGE PROCESSORS

DAS MR Assembler

DAS MR is a two-pass assembler scheduled by job-control directive /DASMR. DAS MR uses the secondary storage device unit for pass 1 output. It reads a source module from the PI logical unit and outputs it on the PO unit. The source input for pass 2 is entered from the SS logical unit.

CONCORDANCE PROGRAM

The background **concordance program (CONC)** provides an indexed listing of all source statement symbols, giving the number of the statement associated with each symbol and the numbers of all statements containing a reference to the symbol. CONC is scheduled by job-control directive /CONC. Upon completion of the concordance listing, control returns to the JCP via EXIT.

Input to CONC is through the SS logical unit. The concordance is output on the LO unit. CONC uses system global file control block SSFCB. If the SS logical unit is an RMD, a /REW or /PFILE directive (section 10) establishes the FCB before the /CONC directive is input to the JCP.

FORTRAN IV COMPILER

The **FORTRAN IV compiler** is a one-pass compiler scheduled by job-control directive /FORT. The compiler inputs a source module from the PI logical unit and produces an object module on the BO and/or GO units and a source listing on the LO unit. No secondary storage is required for a compilation.

If a fatal error is detected, the compiler automatically terminates output to the BO and GO units. LO unit output continues. The compiler reads from the PI unit until an END statement is encountered or a control directive is read. Compilation also terminates on detection of an I/O error or an end-of-device, beginning-of-device, or end-of-file indication from I/O control.

The output comprises relocatable object modules under all circumstances: main programs and subroutines, function, and block-data subprograms.

FORTRAN IV has conditional compilation facilities implemented by an X in column 1 of a source statement. When the X appears in the /FORT directive, all source statements with an X in column 1 are compiled (the X appears on the LO listing as a blank). When the X is not present, all conditional statements are ignored by the compiler. X lines are assigned listing numbers in either case, but the source statement is printed only when the X is present.

VORTEX RPG IV SYSTEM

The VORTEX RPG IV System is a software package for general data processing applications. It combines versatile file and record defining capabilities with powerful processing statements to solve a wide range of applications. It is particularly effective in the processing data for reports. The VORTEX RPG IV system consists of the RPG IV compiler and RPG IV runtime/loader program.

The VORTEX RPG IV compiler and the runtime/loader execute as level zero background programs in unprotected memory. Both the compiler and the runtime/loader will operate in 6K of memory with limited work stack space. The stack space may be expanded and consequently larger RPG programs compiled and executed by use of the /MEM directive.

SECTION 6
LOAD-MODULE GENERATOR

The **load-module generator (LMGEN)** is a background task that generates background and foreground tasks from relocatable object modules. The tasks can be generated with or without overlays, and are in a form called **load modules**.

To be scheduled for execution within the VORTEX operating system, all tasks must be generated as load modules.

ORGANIZATION

LMGEN is scheduled for execution by inputting the job-control processor (JCP) directive /LMGEN.

INPUTS to the LMGEN comprise:

- *Load-module generator directive* input through the SI logical unit.
- *Relocatable object modules* from which the load module is generated.
- *Error-recovery inputs* entered via the SO logical unit.

OUTPUTS from the LMGEN comprise:

- *Load modules* generated by the LMGEN
- *Error messages*
- *Load-module maps* output upon completion of a load-module generation

Load modules are LMGEN-generated absolute or relocatable tasks with or without overlays. They contain all information required for execution under the VORTEX operating system. During their generation, LMGEN uses the SW logical unit as a work unit. Upon completion of the load-module generation, the module is thus resident on the SW unit. LMGEN can then specify that the module be cataloged on another unit, if required, and output the load module to that unit.

Overlays

Load modules can be generated with or without overlays. Load modules with overlays are generated when task requirements exceed core allocation. In this case, the task is divided into overlay segments that can be called as required. Load modules with overlays are generated by use of the OV directive and comprise a root segment and two or more overlay segments, but only the root segment and one overlay segment can be in memory at any given time. Overlays can contain executable codes, data, or both.

When a load module with overlays is loaded, control transfers to the root segment, which is in main memory. The root segment can then call overlay segments as required.

Called overlay segments may or may not be executed, depending on the nature of the segment. It can be an executable routine, or it can be a table called for searching or manipulation, for example. Whether or not the segment consists of executable data, it must have an entry point.

The generation of the load module begins with the root segment, but overlay segments can be generated in any order.

The root segment can reference only addresses contained within itself. An overlay segment can reference addresses contained within itself or within the root segment. Thus, all entry points referenced within the root segment or an overlay segment are defined for that segment and segments subordinate to it, if any.

Common

Common is the area of memory used by linked programs for data storage, i.e., an area common to more than one program. There are two types of common: named common and blank common.

Named common is contained within a task and is used for communication among the subprograms within that task.

Blank common can be used like named common or for communication among foreground tasks.

The extent of blank common for foreground tasks is determined at system generation time. The size of the foreground blank common can vary within each task without disturbing the positional relationship of entries but cannot exceed the limits set at system generation time.

The extent of blank common for background tasks is allocated within the load module. The size of the background blank common can vary within each task, but the combined area of the load module and common cannot exceed available memory.

Each blank common is accessible only by the corresponding tasks, i.e., foreground tasks use only foreground blank common, and background tasks use only background blank common.

All definitions of named and blank common areas for a given load module must be in the first object module loaded to generate that load module.

SECTION 7
DEBUGGING AIDS

The VORTEX system contains two debugging aids: the *debugging program (DEBUG)* and the *snapshot dump program (SNAP)*.

DEBUGGING PROGRAM

The 816-word VORTEX **debugging program (DEBUG)** is added to a task load module whenever the DEBUG option is specified by a load-module generator TIDB directive. The DEBUG object module is the last object module loaded if the root segment of the task is an overlay load module. The load-module generator sets the load-module execution address equal to that of DEBUG.

If the load module has been cataloged, DEBUG executes when the module is scheduled. Otherwise, JCP directive /EXEC is used to schedule the module and DEBUG.

During the execution of DEBUG, the A, B, and X pseudoregisters save the contents of the real A, B, and X registers, and restore the contents of these registers before terminating DEBUG.

When debugging is complete, the input of any job-control directive returns control to the VORTEX system.

INPUTS to DEBUG comprise the directives summarized in the table below. When DEBUG is first entered, it outputs on the Teletype or CRT device the message **DG**** followed by the TIDB task name and the address of the first allocatable memory cell. This message indicates that the system is ready to accept DEBUG directives on the DI logical unit.

Each DEBUG directive has from 0 to 72 characters and is terminated by a carriage return. Directive parameters are separated by commas, but DEBUG treats commas, periods, and equal signs as delimiters.

SNAPSHOT DUMP PROGRAM

The 294-word **snapshot dump program (SNAP)** provides on the DO logical unit both register displays and the contents of specified areas of memory. It is added to a task load module if the task contains a SNAP request and calls the SNAP external routine. SNAP is entered directly upon execution of the SNAP display request CALL SNAP. The SNAP display request is an integral part of the task and is assembled with the task directives. Thus, no external intervention is required to output a SNAP display.

DEBUG Directives

Directive	Description
A	Display and change the contents of the A pseudoregister
Ax	Change, but do not display, the contents of the A pseudoregister
B	Display and change the contents of the B pseudoregister
Bx	Change, but do not display, the contents of the B pseudoregister
Cx	Display and change the contents of memory address x
Gx	Load the contents of the pseudoregisters into the respective A, B, and X registers, and transfer to memory address x
Ix,y,z	Initialize memory addresses x through y with the value of z
O	Display and change the overflow indicator
Sx,y,z,m	Search memory addresses x through y for the z value, using mask m
Ty,x	Place a trap at memory address y, starting execution at address x
Ty	Place a trap at memory address y, starting execution at the last trap location
X	Display and change the contents of the X pseudoregister
Xy	Change, but do not display, the contents of the X pseudoregister
xxxxxx	Display the contents of memory address xxxxxx
xxxxxx,yyyyyy	Display the contents of memory addresses xxxxxx through yyyyyy

SECTION 8
SOURCE EDITOR

The VORTEX operating system source editor (SEDIT) is a background task that constructs sequenced or listed output files by selectively copying sequences of records from one or more input files. SEDIT operates on the principle of forward-merging of subfiles and has file-positioning capability. The output file can be sequenced and/or listed.

ORGANIZATION

SEDIT is scheduled by the job-control processor (JCP) upon input of the JCP directive /SEDIT. Once activated, SEDIT inputs and executes directives from the SI logical unit until another JCP directive (first character = /) is input, at which time SEDIT terminates and the JCP is again scheduled.

SEDIT has a buffer area for 100 source records in MOVE operations. To increase this, input a /MEM directive, immediately preceding the /SEDIT directive, where each 512-word block will increase the capacity of the buffer area by 12 source records.

INPUTS to SEDIT comprise:

a. *Source-editor directives* input through the SI logical unit.

b. *Old source records* input through the IN logical unit.

c. *New or replacement source records* input through the ALT logical unit.

d. *Error-recovery inputs* entered via the SO logical unit.

Source-editor directives specify both the changes to be made in the source records, and the logical units to be used in making these changes. The directives are input through the SI logical unit and listed as read on the LO logical unit, with the VORTEX standard heading at the top of each page. If the SI logical unit is a Teletype or a CRT device, the message **SE**** is output to it before directive input to indicate that the SI unit is waiting for SEDIT input.

There are two groups of source-editor directives: the copying group and the auxiliary group. **The copying group directives** copy or delete source records input on the IN logical unit, merge them with new or replacement source records input on the ALT unit, and output the results on the OUT unit. Copying-group directives must appear in sequence according to their positioning-record number since there is no reverse positioning. If the remainder of the source records on the IN unit are to be copied after all editing is completed, this must be explicitly stated by an FC directive. Ends of file are output only when specified by FC or WE directives. The processing of string-editing directives is different from that of record-editing directives. A string-editing directive affects a specified record, where source records on the IN unit are copied onto the OUT unit until the specified record is found and read into memory from the IN unit. After editing, this record remains in memory and is not yet copied onto the OUT unit. This makes possible multiple field-editing operations on a single source record. **The auxiliary group directives** are those used for special I/O or control functions.

OUTPUTS from the SEDIT comprise:

a. *Edited source-record sequences* output on the OUT logical unit.

b. *Error messages.*

c. *The listing of the SEDIT directives* on the LO logical unit.

d. *Comparison outputs* (compare-inputs directive).

e. Listing of source records on the LO logical unit when specified by the LIST directive.

SOURCE-EDITOR DIRECTIVES

This section describes the SEDIT directives:

a. Copying group:

- **AS** Assign logical units
- **AD** Add record(s)
- **SA** Add string
- **REPL** Replace record(s)
- **SR** Replace string
- **DE** Delete record(s)
- **SD** Delete string
- **MO** Move record(s)

b. Auxiliary group:

- **FC** Copy file
- **SE** Sequence records
- **LI** List records
- **GA** Gang-load all records
- **WE** Write end-of-file
- **REWI** Rewind
- **CO** Compare records

SEDIT directives begin in column 1 and comprise sequences of character strings having no embedded blanks. The character strings are separated by commas (,) or by equal signs (=). The directives are free-form and blanks are permitted between individual character strings of the directive, i.e., before or after commas (or equal signs). Although not required, a period (.) is a line terminator. Comments can be inserted after the period.

The general form of an SEDIT directive is

name,*p(1)*,*p(2)*,...,*p(n)*

where

name	is one of the directive names given above or a longer string beginning with one of the directives names (e.g., AS or ASSIGN)
each *p(n)*	is a parameter defined below under the descriptions of the individual directives

SECTION 9
FILE MAINTENANCE

The VORTEX file-maintenance component (**FMAIN**) is a background task that manages file-name directories and the space allocations of the files. It is scheduled by the job-control processor (JCP) upon input of the JCP directive /FMAIN.

Only files assigned to rotating-memory devices (disc or drum) can be referenced by name.

File space is allocated within a partition forward in contiguous sectors of the same cylinder, skipping bad tracks. The only exception to this continuity is the file-name directory itself, which is a sequence of linked sectors that may or may not be contiguous.

ORGANIZATION

FMAIN inputs file-maintenance directives received on the SI logical unit and outputs them on the LO logical unit and on the SO logical unit if it is a different physical device from the LO unit. Each directive is completely processed before the next is input to the JCP buffer.

If the SI logical unit is a Teletype or a CRT device, the message **FM**** is output on it before input to indicate that the SI unit is waiting for FMAIN input.

If there is an error, one of the error messages is output on the SO logical unit, and a record is input from the SO unit to the JCP buffer. If the first character of this record is /, FMAIN exits via the EXIT macro. If the first character is C, FMAIN continues. If the first character is neither / nor C, the record is processed as a normal FMAIN directive. FMAIN continues to input and process records until one whose first character is / is detected, when FMAIN exits via exit. (An entry beginning with a carriage return is an exception to this, being processed as an FMAIN directive).

FMAIN outputs four types of listing to the LO logical unit:

- **Directive listing** lists, without modification, all FMAIN directives entered from the SI logical unit.
- **Directory listing** lists file names from a logical unit file-name directory in response to the FMAIN directive LIST.
- **Deletion listing** lists file names deleted from a logical unit file-name directory in response to the FMAIN directive DELETE.
- **Object-module listing** lists the object-module input in response to the FMAIN directive ADD.

Relocatable Object Modules

Outputs from both the DAS MR assembler and the FORTRAN compiler are in the form of relocatable object modules. Relocatable object modules can reside on any VORTEX-system logical unit. Before object modules can be read from a unit by the FMAIN INPUT and ADD directives, an I/O OPEN with rewinding is performed on the logical unit, i.e., the unit (except paper-tape or card readers) is first positioned to the beginning of device or load point for that unit. Object modules can then be loaded until an end-of-file mark is found.

The system generator (section 13) does not build any object-module library. FMAIN is the only VORTEX component used for constructing user object-module libraries.

A VORTEX physical record on an RMD is 120 words. Object-module records are blocked two 60-word records per VORTEX physical record. However, in the case of an RMD assigned as the SI logical unit, object modules are not blocked but assumed to be one object-module record per physical record.

FILE-MAINTENANCE DIRECTIVES

This section describes the file-maintenance directives:

- CREATE file
- DELETE file
- RENAME file
- ENTER new file name
- LIST file names
- INIT (initialize) directory
- INPUT logical unit for object module
- ADD object module

The general form of a file-maintenance directive is

directive,lun,*p(1),p(2),...,p(n)*

where

directive	is one of the directives listed above in capital letters
lun	is the number or name of the affected logical unit
each *p(n)*	is a parameter defined under the descriptions of the individual directives

SECTION 10
INPUT/OUTPUT UTILITY PROGRAM

The I/O utility program (IOUTIL) is a background task for copying records and files from one device onto another, changing the size and mode of records, manipulating files and records, and formatting the records for printing or display.

ORGANIZATION

IOUTIL is scheduled for execution by inputting JCP directive /IOUTIL on the SI logical unit. If the SI logical unit is a Teletype or a CRT device, the message **IU**** is output to indicate that the SI unit is waiting for IOUTIL input. Once activated, IOUTIL inputs and executes directives from the SI unit until another JCP directive (first character is a slash) is input, at which time IOUTIL terminates and the JCP is again scheduled.

I/O UTILITY DIRECTIVES

This section describes the IOUTIL directives:

- COPYF Copy file
- COPYR Copy record
- SFILE Skip file
- SREC Skip record
- DUMP Format and dump
- WEOF Write end of file
- REW Rewind
- PFILE Position file
- CFILE Close file

IOUTIL directives begin in column 1 and comprise sequences of character strings having no embedded blanks. The character strings are separated by commas (,) or by equal signs (=). The directives are free-form and blanks are permitted between individual character strings of the directive, i.e., before or after commas (or equal signs). Although not required, a period (.) is a line terminator. Comments can be inserted after the period.

The general form of an IOUTIL directive is

name,*p(1)*,*p(2)*,...,*p(n)*

where

name is one of the directive names given above

each *p(n)* is a parameter defined below under the descriptions of the individual directives

SECTION 11
SUPPORT LIBRARY

The VORTEX system has a comprehensive subroutine library directly available to the user. The library contains mathematical subroutines to support the execution of a program, plus many commonly used utility subroutines. To use the library, merely code the proper call in the program, or, for the standard FORTRAN IV functions, implicitly reference the subroutine (e.g., A = SQRT(B) generates a CALL SQRT(B)). All calls generate a reference to the required routine, and the load-module generator brings the subroutine into memory and links it to the calling program.

The performance of several routines in the support library is improved through the use of the V73 Floating Point Firmware on V73 systems having Writable Control Store (WCS). The necessary firmware and library routines which call the firmware are added to the Object Module Library (OM) by executing the supplemental WCS job stream supplied with the System Generation Library.

SECTION 12
REAL-TIME PROGRAMMING

VORTEX real-time applications allow the user to interface directly with special devices, develop software that is interrupt-driven, and utilize reentrant subroutines. Four areas are covered in this section:

- Interrupts
- Task-scheduling
- Coding reentrant subroutines
- Coding I/O drivers

INTERRUPTS

External Interrupts

Priority interrupt module (PIM) hardware: A PIM comprises a group of eight interrupt lines and an eight-bit register. The register holds a mask where each set bit disarms a line. VORTEX allows up to eight PIMs for a maximum of 64 lines. The system of PIMs and lines is called the *external interrupt system.*

VORTEX interrupt line handlers: At system-generation time, a user specifies all interrupt-driver tasks. These include those that allow VORTEX to service the interrupt, as well as those that are directly connected and service the interrupt themselves. Then, VORTEX constructs a line-handler for each interrupt in the system.

Directly connected routines preempt VORTEX and are thus used only when response time demands it. The rules for the use of directly connected routines are:

a. All volatile registers used by the routine are restored before returning to the interrupted task.

b. Interrupts remain disabled during processing.

c. IOC and RTE calls are not allowed.

d. Execution time is minimal.

e. PIM interrupts are enabled before returning to the interrupted task through word 0 of the line handler. The real time clock (RT clock) is enabled only if the task is not the VORTEX RT clock processor (location 0300, V$CTL, contains 037 if the VORTEX RT clock processor is interrupted).

Internal Interrupts

VORTEX recognizes and services internal interrupts related to various hardware components. The processing routines are all directly connected and are the highest-priority tasks in the system.

Memory protection interrupt: When the background area is active, it is run as an unprotected area of memory with the rest of the system protected. In such a situation, memory protection interrupts are generated when the background task attempts to execute a "privileged" instruction such as external control or halt, or attempts to jump into, write into, or perform I/O on protected memory. The memory protection routine processes all protection violation interrupts and is the highest-priority interrupt in the system.

Real-time clock interrupt: The real-time clock interrupt provides the basis for timekeeping in VORTEX. It can be set to a minimum resolution of 5 milliseconds. However, one greater than 5 milliseconds (i.e., 10-20 milliseconds) reduces overhead when the system does not have high-resolution timekeeping requirements. Upon receipt of an interrupt, the time-of-day is updated and the TIDBs are scanned for any time-driven task requiring activation. PIMs are disabled for approximately 18 cycles during real-time clock interrupt-processing. The clock routine is the third-highest priority interrupt in VORTEX.

Power failure/restart interrupt: An interrupt occurs when the system detects a power failure. The VORTEX power-failure processor saves the contents of volatile registers and the status of the overflow indicator, sets a power-failure flag, and halts with the I register set to 0123. Following the power-up sequence, the PF/R hardware generates an interrupt. Upon entry to the VORTEX power-up processor, the power-failure flag is checked.

SCHEDULING

System Flow

VORTEX is designed around the TIDB. This block contains all of the information about a task during its execution. The setting and clearing of status bits in the TIDB causes a task to flow through the system. Two register stacks are saved within the TIDB: a reentrant (suspend register) stack, and an interrupt stack.

The dispatcher is the prime mover of tasks through the system. When any function has reached a termination point or has to wait for an I/O operation, the task gives control to the dispatcher, which then finds another task to execute. A task maintains control until it gives control to the dispatcher, or to the interrupt task if the interrupt-processing task has a higher priority. The contents of the interrupted task's volatile registers are saved in its TIDB interrupt stack and control goes to the dispatcher, which searches for the highest-priority active task for execution.

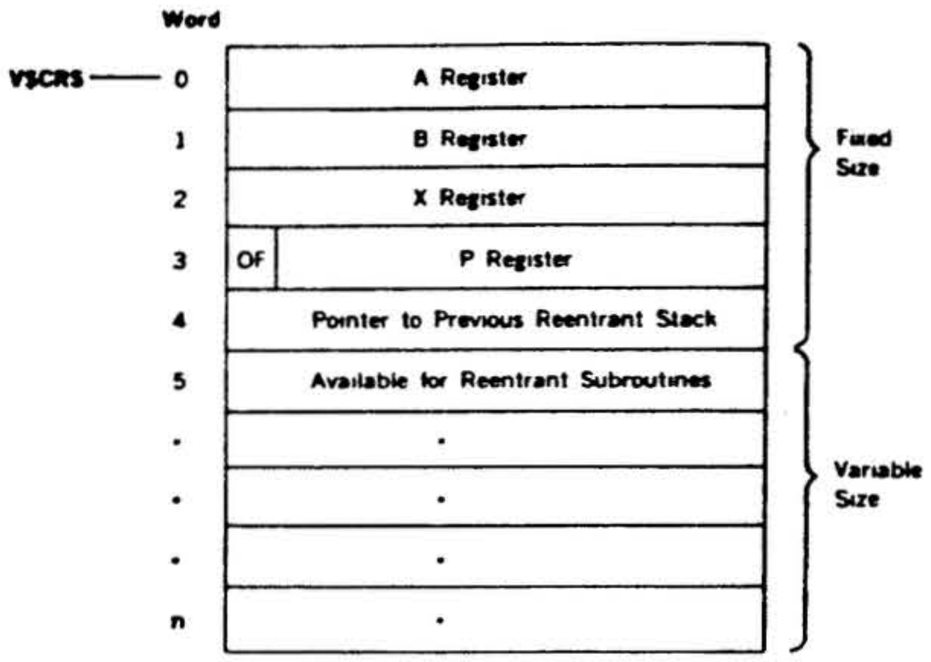
Each TIDB is placed in sequence by priority level and threaded. Two stacks are maintained in the system: a busy stack and an unused stack. When a task is scheduled for execution, a TIDB is allocated from the unused stack and threaded onto the busy stack according to priority level.

REENTRANT SUBROUTINES

The user can write a reentrant subroutine and add it to the VORTEX nucleus. RTE service requests ALOC and DEALOC interface between a task and a reentrant subroutine.

A task calls a reentrant subroutine via an ALOC request that allocates a variable-length push-down reentrant stack with the external name V$CRS. The reentrant subroutine address is specified in the ALOC calling sequence. The first word of the reentrant subroutine contains the number of words to be allocated.

A reentrant stack generated by the ALOC request has the format:



CODING AN I/O DRIVER

The IOC (section 3) activates I/O drivers. When a user task makes an I/O request, it executes a JSR V$IOC,X instruction with V$IOC containing the IOC entry address. IOC then makes validity checks on the parameters specified in the request block (RQBLK) that immediately follows the JSR instruction. IOC queues RQBLK to the I/O driver controller table (CTBL), and activates the corresponding controller-table TIDB. The TIDB contains the entry address for the I/O driver. To determine the proper CTBL and corresponding TIDB, IOC obtains the logical-unit number from RQBLK. By referring to the logical-unit table (LUT), IOC then finds the device assigned to that logical unit. Each device has a device specification table (DST) associated with it, and each DST has a corresponding controller table.

I/O Driver System Functions

Each I/O driver under IOC performs certain system pre- and post-processing functions.

Pre-interrupt processing: If the I/O driver uses a BIC, the driver calls V$BIC with the X and A registers set to the initial and final buffer addresses respectively to build and execute the initial BIC transfer instruction. If the BIC is shared, the interrupt line handler is modified to the proper interrupt event word setting (TBEVNT) and TIDB address. V$BIC performs this modification if the word immediately following the call (JSR V$BIC,B) is nonzero, since this is assumed to be the interrupt event word setting. If,It is zero, no line handler modification is performed. The I/O driver clears the interrupt event word (TBEVNT) in the controller TIDB immediately preceding a DELAY (type 2) call. To wait for an interrupt, the I/O driver executes a DELAY (type 2) call with a time-out. The return to the driver, either from a time-out or interrupt is to the address immediately following the call. The contents of the X register is not restored following a DELAY call but the A and B registers are. Executing a TXA immediately preceding and a TAX following the DELAY call X restores the value in the X register.

Interrupt processing: The driver clears the time-delay flag (TBST bit 6) set by the DELAY call, and checks TBEVNT to determine if an interrupt occurred (TBEVNT = 0 indicates a time-out). Following the interrupt processing, the driver clears TBEVNT and calls DELAY (type 2) for the next instruction.

SECTION 13
SYSTEM GENERATION

The VORTEX **system-generation component (SGEN)** tailors the VORTEX operating system to specific user requirements. SGEN is a collection of programs on magnetic tape, punched cards, or disc pack. It includes all programs (except the key-in loader) for generating an operating VORTEX system on an RMD.

The block diagram below shows the data flow through SGEN.

ORGANIZATION

SGEN is a four-phase component comprising:

- I/O interrogation
- SGEN directive processing
- Building the VORTEX nucleus
- Building the library and the resident-task configuration

I/O interrogation specifies the peripherals to:

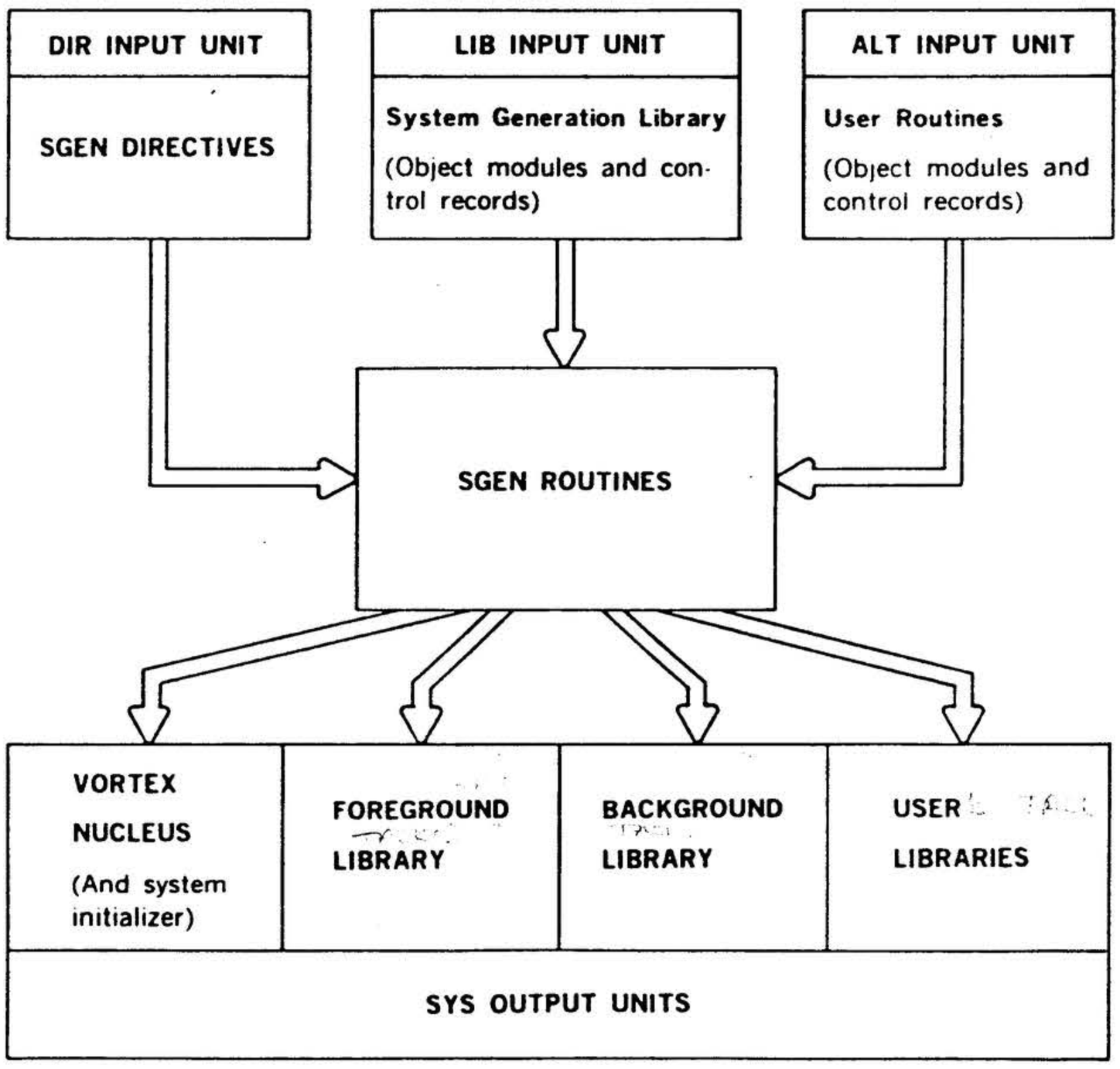
a. Input VORTEX system routines (LIB unit)

b. Input user routines (ALT unit)

c. Input SGEN directives (DIR unit)

d. Output the VORTEX system generation (SYS unit)

e. List special information and input user messages (LIS unit)



SGEN Data Flow

SECTION 14
SYSTEM MAINTENANCE

The VORTEX **system-maintenance component (SMAIN)** is a background task that maintains the **system-generation library (SGL).** The SGL comprises all object modules and their related control records required to generate a generalized VORTEX operating system (see block diagram below).

SYSTEM-MAINTENANCE DIRECTIVES

This section describes the SMAIN directives:

•	IN	Specify input logical unit
•	OUT	Specify output logical unit
•	ALT	Specify output logical unit for new SGL items
•	ADD	Add items to the SGL
•	REP	Replace SGL items
•	DEL	Delete items from the SGL
•	LIST	List the old SGL
•	END	End input of SMAIN directives

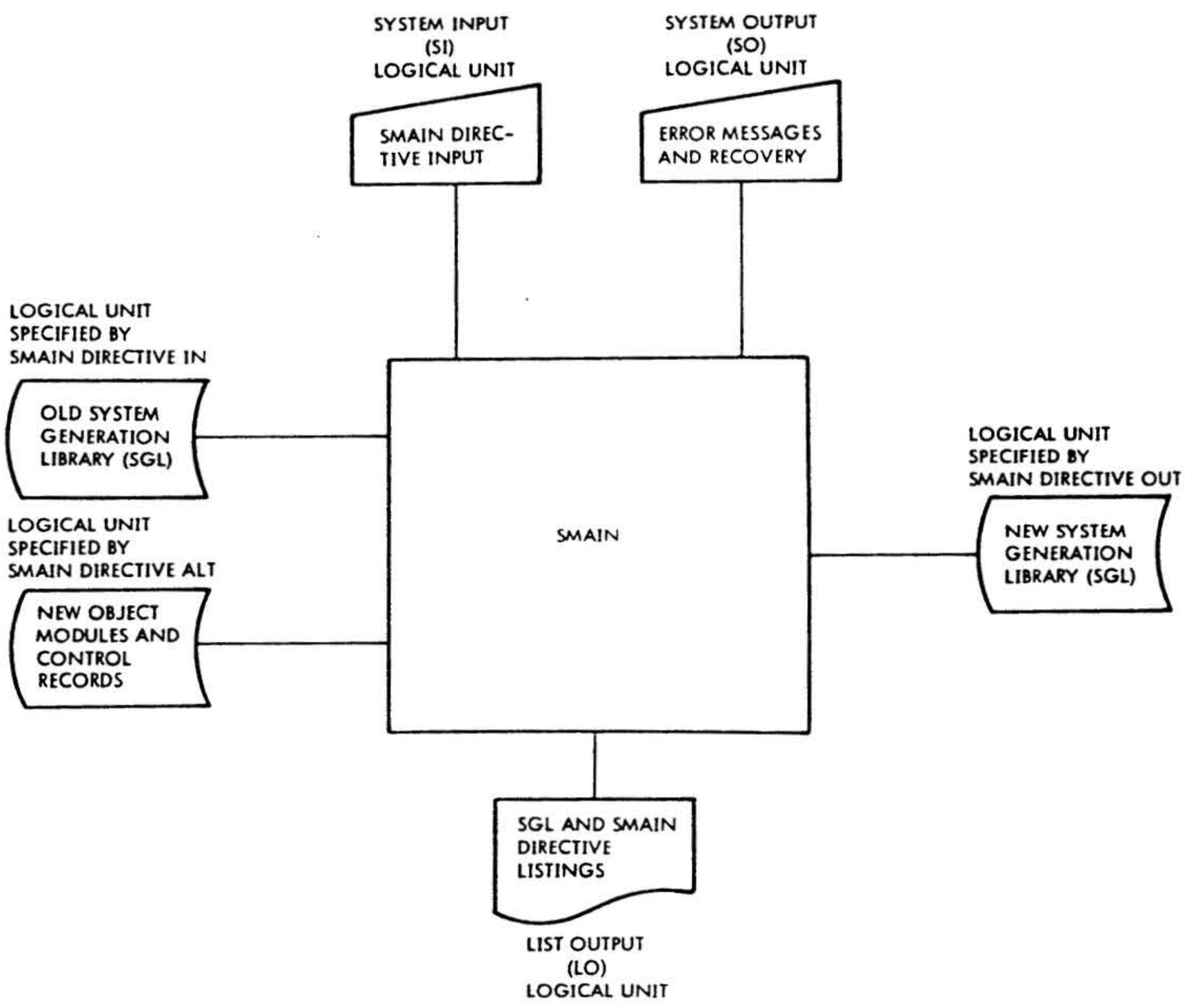
SMAIN directives begin in column 1 and comprise sequences of character strings having no embedded blanks. The character strings are separated by commas (,) or by equal signs (=). The directives are free-form and blanks are permitted between the individual character strings of the directive, i.e., before or after commas (or equal signs). Although not required, a period (.) is a line terminator. Comments can be inserted after the period.

The general form of an SMAIN directive is

name,*p(1)*,*p(2)*,...,*p(n)*

where

name	is one of the directive names given above (any other character string produces an error)
each *p(n)*	is a parameter defined below under the descriptions of the individual directives



SMAIN Block Diagram

SECTION 15
OPERATOR COMMUNICATION

The operator communicates with the VORTEX system through the **operator communication component** by means of *operator key-in requests* input through the *operator communication (OC) logical unit.*

DEFINITIONS

An **operator key-in request** is a string of up to 80 characters beginning with a semicolon. The request is initiated by the operator and is input through the OC unit. An operator key-in request is independent of I/O requests via the IOC (section 3) and, hence, is known as an *unsolicited request.*

The **operator communication (OC) logical unit** is the logical unit through which the operator inputs key-in requests. There is only one OC unit in the VORTEX system. Initially, the OC unit is the first Teletype, but this assignment can be changed by use of the ;ASSIGN key-in request.

OPERATOR KEY-IN REQUESTS

This section describes the operator key-in requests:

- ;SCHED Schedule foreground task
- ;TSCHED Time-schedule foreground task
- ;ATTACH Attach foreground task to PIM line
- ;RESUME Resume task
- ;TIME Enter or display time-of-day
- ;DATE Enter date
- ;ABORT Abort task
- ;TSTAT Test task status
- ;ASSIGN Assign logical unit(s)
- ;DEVDN Device down
- ;DEVUP Device up
- ;IOLIST List logical-unit assignments

The general form of an operator key-in request is

;request,*p(1)*,*p(2)*,,...,*p(n)*cr

where

request	is one of the key-in requests listed above in capital letters
each *p(n)*	is a parameter defined under the descriptions of the individual key-in requests below
cr	is the carriage return, which terminates all operator key-in requests

Each operator key-in request begins with a semicolon (;) and ends with a carriage return. Parameters are separated by commas. A backarrow (←) deletes the preceding character. A backslash (\) deletes the entire present key-in request.

The table below shows the system names of physical I/O devices as used in operator key-in requests.

Physical I/O Devices

System Name	Physical Device
DUM	Dummy
CPcu	Card punch
CRcu	Card reader
CTcu	Cathode ray tube (CRT) device
Dcup	Rotating-memory device (RMD) (disc/drum)
LPcu	Line printer or Statos-31
MTcu	Magnetic tape unit
PTcu	High-speed paper tape reader/
TYcu	Teletype printer/keyboard
CIma, COma	Process I/O

NOTES

c = Controller number. For each type of device, controllers are numbered from 0 as required.

u = Unit number. For each controller, units are numbered from 0 as required (within the capacity of the controller).

cu can be omitted to specify unit 0 controller 0, e.g., CR00 or CR.

p = Partition letter. RMD partitions are lettered from A to T as required to refer to a partition on the specified device, e.g., D00A.

SECTION 16
OPERATION OF THE VORTEX SYSTEM

This section explains the operation of devices in the VORTEX system, the loading of the system bootstrap and procedures for changing and initializing the disc pack during VORTEX operation.

SECTION 17
VORTEX PROCESS INPUT/OUTPUT

INTRODUCTION

VORTEX supports a number of VDM devices which are used in industrial applications for a wide range of monitor and control purposes. These devices are called 'Process Input/ Output' devices and are listed below:

VDM Model	Description
70-8310 and -8311	Digital Output Module
70-8410 and -8411	Digital Input Module
70-800x and -801x	Analog to Digital
70-8020 and -8021 70-8022 and -8023	Converter/Multiplexor
70-821x, -8220, and -8221	Digital to Analog Module

The VORTEX Support Library includes a number of subroutines with FORTRAN calling sequences defined by the Instrument Society of America (ISA), which are useful for input, output, and manipulation of process data.

SECTION 18
WRITABLE CONTROL STORE

The Writable Control Store (WCS) option extends the Varian 73 processor's read-only control store to permit the addition of new instructions, development of microdiagnostics and optimal tailoring of the computer system to its application. Unlike the read-only control store, which contains the Varian 73 standard instruction set and cannot be altered, the WCS can be loaded from main memory under control of certain I/O instructions. The capabilities of WCS give the user more complete access to the resources of the Varian 73 computer system.

Supporting software for the WCS includes the following:

- Microprogram assembler MIDAS
- Microprogram simulator MICSIM
- Microprogram utility loader and diagnostic MIUTIL

All software for microprogram development operates under VORTEX.

SECTION 19
ERROR MESSAGES

This section comprises a directory of VORTEX operating system error messages, arranged by VORTEX component.

ERROR MESSAGE INDEX

Except for the language processors (section 5), VORTEX error messages each begin with two letters that indicate the corresponding component:

Messages beginning with:	Are from component:
CM	Concordance program
DG	Debugging program
EX	Real-time executive
FM	File maintenance
IO	I/O control
IU	I/O utility
JC	Job-control processor
LG	Load-module generator
MS	Microprogram simulator
MU	Microprogram utility
NC	VTAM Network control
OC	Operator communication
RP	RPG IV Compiler
RT	RPG IV Runtime/Loader
SE	Source editor
SG	System generator
SM	System maintenance
*	DAS MR assembler

The entire VORTEX system provides many services of which only some are initiated by specific requests from the user. Other services are automatic without the user's intervention. Automatically, VORTEX gives efficient management of memory, handling of errors such as power failure, and scheduling the use of the processor. To a user, such as a programmer writing in FORTRAN, these activities are transparent. Applications written in higher level languages have their storage assigned by VORTEX, so actual physical addresses are not the programmer's concern. For the user of VORTEX, this means simplified programming and operating of the system. The complexity of VORTEX underlies the smooth operation of many concurrent tasks and efficient use of the system's resources.

Additional information about using VORTEX for your computer applications can be obtained from sales offices of Varian Data Machines.