SY28-0761-0
File No. S370-36

Systems

OS/VS2
System Logic Library
Volume 1

VS2.03.804
VS2.03.805
VS2.03.807

IBM

Pages numbered as duplicates in this publication must be retained because each of these documents information specific to individual Selectable Units.

This minor revision incorporates the following Selectable Units:

Scheduler Improvements	VS2.03.804
Supervisor Performance #1	VS2.03.805
Supervisor Performance #2	VS2.03.807

The selectable unit to which the information applies, is noted in the upper corner of the page.

First Edition (July, 1976)

This is a reprint of SY28-0713-0 incorporating changes released in the following Selectable Unit Newsletters:

SN28-2682 (dated May 28, 1976)
SN28-2686 (dated May 28, 1976)
SN28-2691 (dated May 28, 1976)

This edition applies to Release 3.7 of OS/VS2 and to all subsequent releases of OS/VS2 until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

System Logic Library comprises seven volumes. Following is the content and order number for each volume.

OS/VS2 System Logic Library,

Volume 1 contents: SY28-0713
- MVS logic introduction
- Abbreviation list
- Index for all volumes

Volume 2 contents: SY28-0714
- Method of Operation diagrams for
- Communications Task
- Command Processing
- Region Control Task (RCT)
- Started Task Control (STC)
- LOGON Scheduling

Volume 3 contents: SY28-0715
- Method of Operation diagrams for
- System Resources Manager (SRM)
- System Activity Measurement Activity (MF/1)
- JOB Scheduling
  - —Subsystem Interface
  - —Master Subsystem
  - —Initiator/Terminator
  - —SWA Create Interface
  - —Converter/Interpreter
  - —SWA Manager
  - —Allocation/Unallocation
  - —System Management Facilities (SMF)
  - —System Log
  - —Checkpoint/Restart

Volume 4 contents: SY28-0716
- Method of Operation diagrams for
- Timer Supervision
- Supervisor Control
- Task Management
- Program Management
- Recovery/Termination Management (R/TM)

Volume 5 contents: SY28-0717
- Method of Operation diagrams for
- Real Storage Management (RSM)
- Virtual Storage Management (VSM)
- Auxiliary Storage Management (ASM)

Volume 6 contents: SY28-0718
- Program Organization

Volume 7 contents: SY28-0719
- Directory
- Data Areas
- Diagnostic Aids

Please note that if you use only one order number, you will only receive that volume. To receive all seven volumes, you must either use all seven form numbers or, simply the following number: SBOF-8210. If you use SBOF-8210, you will receive all seven volumes.

The publication is intended for persons who are debugging or modifying the system. For general information about the use of the MVS system, refer to the publication *Introduction to OS/VS Release 2,* GC28-0661.

How This Publication is Organized

This publication contains six chapters. Following, is a synopsis of the information in each section:

- *Introduction and Master Index* — an overview of each of the functions this publication documents, an abbreviation list of all acronyms used in the publication, and a complete index for all seven volumes.
- *Method of Operation* — a functional approach to each of the subcomponents, using both diagrams and text. Each subcomponent begins with an introduction; all the diagrams and text applying to that subcomponent follow.
- *Program Organization* — a description of module-to-module flow for each subcomponent; a description of each module's function, including entry and exit. The module-to-module flow is ordered by subcomponent. The module descriptions are in alphabetic order without regard to subcomponent.
- *Directory* — a cross-reference from names in the various subcomponents to their place in the source code and in the publication.
- *Data Areas* — a description of the major data areas used by the subcomponents (only those, however, that are not described in *OS/VS Data Areas,* SYB8-0606, which is on microfiche); a data area usage table, showing whether a module reads or updates a data area; a control block overview diagram for each subcomponent, showing the various pointer schemes for the control blocks applicable to each subcomponent; a table detailing data area acronyms, mapping macro instructions, common names, and symbol usage table.

- *Diagnostic Aids* — the messages issued, including the modules that issue, detect, and contain the message; register usage; return codes; wait state codes; and miscellaneous aids.

Corequisite Reading

The following publications are corequisites:

- *OS/VS2 JES2 Logic,* SY28-0622
- *OS/VS Data Areas,* SYB8-0606 (This document is on microfiche.)
- *OS/VS2 System Initialization Logic,* SY28-0623

Contents

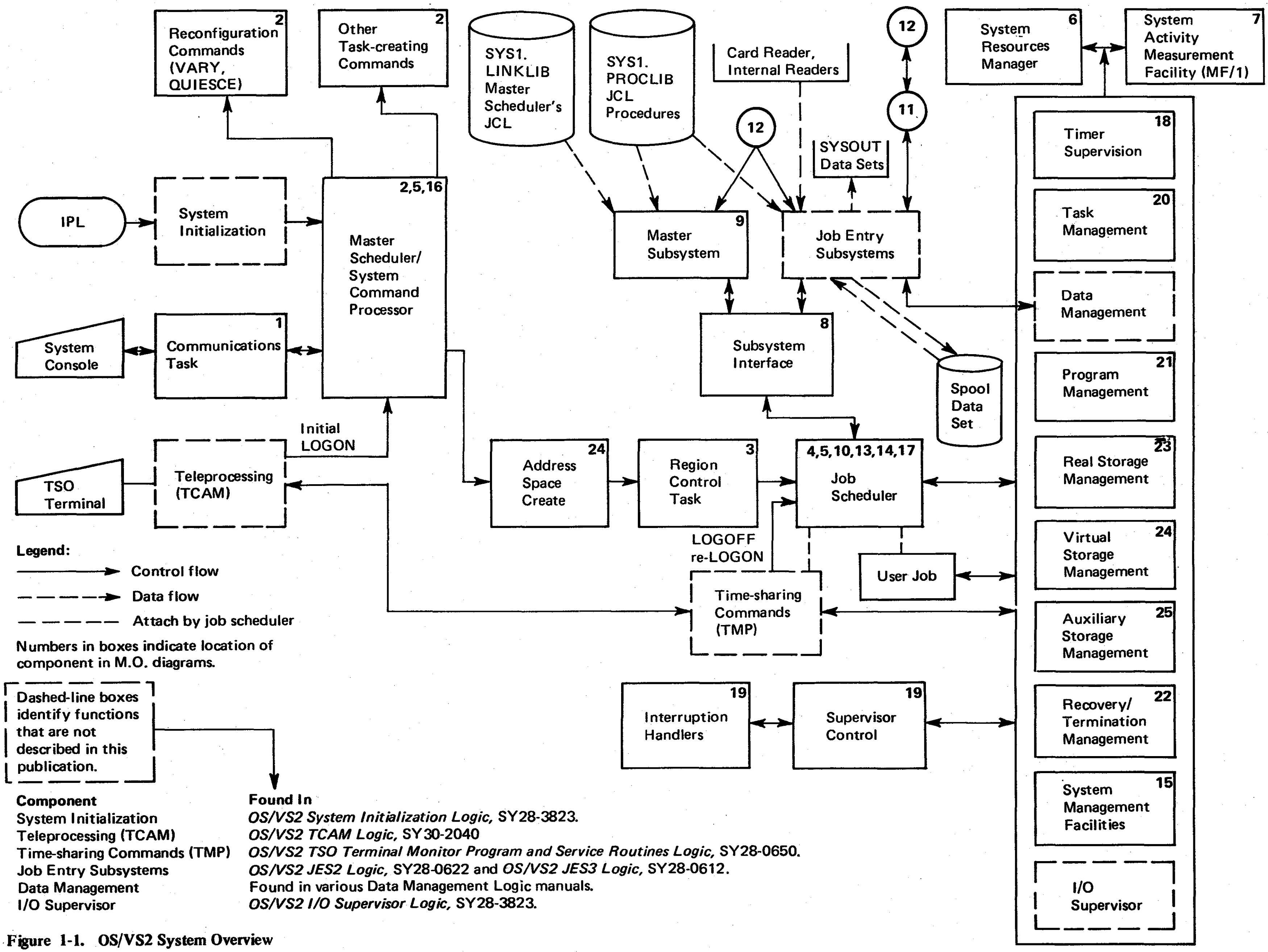
Figures

MVS is a virtual storage operating system with multiprogramming, multiprocessing, time sharing (TSO), and a job entry subsystem. The basic purpose of the system is to improve the use of the system's resources and reliability. The publication *Introduction to OS/VS2 Release 2*, GC28-0661 defines the model support, minimum storage requirements, and minimum configuration for the system.

This publication documents the supervisor and scheduler portions of the system, plus the System Resources Manager, System Activity Measurement Facility, and Auxiliary Storage Manager. For convenience, however, Figure 1-1 shows an overview of the entire system. As the figure's key notes, items in dotted boxes are not documented in this publication.

Synopsis of Coverage

The following list gives a brief synopsis of the subcomponents this publication documents. The list contains the name of the subcomponent and a brief statement of the subcomponent's function. A fuller explanation of each subcomponent follows later in this Introduction.



Component	Found In
System Initialization	*OS/VS2 System Initialization Logic,* SY28-3823.
Teleprocessing (TCAM)	*OS/VS2 TCAM Logic,* SY30-2040
Time-sharing Commands (TMP)	*OS/VS2 TSO Terminal Monitor Program and Service Routines Logic,* SY28-0650.
Job Entry Subsystems	*OS/VS2 JES2 Logic,* SY28-0622 and *OS/VS2 JES3 Logic,* SY28-0612.
Data Management	Found in various Data Management Logic manuals.
I/O Supervisor	*OS/VS2 I/O Supervisor Logic,* SY28-3823.

Figure 1-1. OS/VS2 System Overview

COMPONENT	M O Diagram Reference	FUNCTION
Communications Task	1	Controls data transfer between programs and operator consoles.
Command Processor	2	Handles command scheduling and command execution.
Reconfiguration Commands	2	Allows the operator to alter the system configuration by varying the online/offline status of central processor units, channels, device paths, and real storage locations.
Region Control Task (RCT)	3	Attaches other tasks in its address space when RCT is dispatched. For every address space created, one RCT is created and is the only task associated with it.
Started Task Control (STC)	4	Processes jobs begun by START, LOGON, or MOUNT commands.
LOGON Scheduling	5	Verifies the user's authorization against the UADS (user attribute data set) and schedules the user's session.
System Resources Management (SRM)	6	Maintains information about the status of the system, its resources, and its users. It uses information to evaluate the condition and performance of the system, and to decide whether to change the status of the system either to improve system performance or to better some user requirement.
System Activity Measurement Facility (MF/1)	7	Controls the collection, recording, and reporting of system activity measurements.
Subsystem Interface	8	Provides the means used by the control program and data management to communicate with the job entry subsystem and the master subsystem.
Job Scheduler:		Consists of requesting a job from the queue the JES readers build, building control blocks from that job's JCL, and assigning the system resources each job step requires. Job scheduling routines also perform termination processing for jobs and steps.
Master Subsystem	9	
Initiator/Terminator	10	
SWA Create Interface	11	
Converter/Interpreter	12	
SWA Manager	13	
Allocation/Unallocation	14	
System Management Facilities (SMF)	15	
System Log	16	
Checkpoint/Restart	17	
Timer Supervisor	18	Provides the means to obtain the date and time of day, measure periods of time, schedule activity, or set the interval timer.
Supervisor Control	19	Includes exit processors, task switching routines, the dispatcher, locking, and inter-processor communications. Supervisor control routines also perform the initial processing of interruptions and pass control to the routines that will handle them. The service manager, using dispatching techniques, is designed to improve system response.
Task Management	20	Creates and deletes subtasks, controls the execution of tasks in one or more address spaces, and provides information services for the requestor.
Program Management	21	Searches for and schedules requested modules, sychronizes exit routines, and brings modules into storage.
Recovery/Termination Management (R/TM)	22	Controls software recovery processing and supplies the system services of normal and abnormal task and address space termination.
Real Storage Management (RSM)	23	Assigns real storage page frames from an available pool when a user must have data in real storage. RSM routines repossess frames when real storage is freed, when a user is terminated or swapped out, or when the pool of available frames is depleted. RSM assigned the appropriate frames for V=R regions.
Virtual Storage Management (VSM)	24	Allocates address spaces and virtual storage within address spaces. The VSM routines record what storage is free and what storage is allocated within every address space in the system.
Address Space Create	24	Creates a new user address space.

Auxiliary Storage Management (ASM)	25	Supports the dynamic paging requirements of RSM and Virtual Block Processor (VBP). ASM assigns auxiliary storage for all paging, swapping, and temporary data sets (VIO).

An Overview of Each Subcomponent

The following discussion gives an overview of each subcomponent that this publication documents.

Communications Task

The console communications task (comm task) is the system interface to the operator. It provides the I/O device dependent support that reads data from and writes data to the operator consoles.

User/system programs communicate with the operator through WTO (write-to-operator) and WTOR (Write-to-operator with reply) macro instructions. Additional control over messages sent to the operator, especially over messages displayed on a graphics device, is available through the DOM (delete operator messages) macro instruction.

Each WTO, WTOR, and DOM request causes control to be passed to the job entry system responsible for scheduling the job that issued the request. The subsystem is passed a pointer to the WQE (or each WQE in the case of a multi-line WTO), ORE (operator reply element), or DOM control block. The subsystem may delete the WTO or WTOR thus suppressing the printing at any console. Any suppressed message will appear on the hardcopy device if there is one. If the subsystem suppresses a WTOR, it must generate a reply. The subsystem may change the message routing and the message text.

Command Processor

Command processor modules include modules that perform a common function for all commands, and individual command processors (that can consist of more than one module) to handle one or a group of commands.

The common processing modules:

- Establish an ESTAE environment.
- Handle message processing.
- Translate commands.
- Check command authority.
- Route commands to the proper processors.
- Interface with subsystems for command identification.
- Create control blocks.
- Manipulate control block queues.
- Perform recovery functions in the case of failure.

For many commands, these modules store the command in CSCBs (command scheduling control blocks) which are chained together.

Individual command processors handle individual commands based on the command verbs and their specific keywords. In some cases, the processors perform checking and routing services for commands with multiple keywords and operands: an individual processor can pass control to or signal other subcomponents to perform the final processing. In other cases, the processors perform the final processing.

Reconfiguration Commands

System reconfiguration is the process of changing, physically or logically, the type or quantity of units (CPUs, channels, and real storage pages) available to the system. Physical reconfiguration connects or disconnects units from the system. Logical reconfiguration changes system tables to notify the control program of the number and types of units available to it.

Logical reconfiguration can be performed during IPL. It can also be performed as a result of issuing the VARY command, which changes the status of paths, devices, channels, CPUs, or areas of real storage to online or offline. The six reconfiguration commands are QUIESCE, DISPLAY MATRIX, VARY CPU, VARY CHANNEL, VARY PATH, VARY STORAGE; and there are two subroutines, the IEEVDEV device path dependency analysis subroutine, and the stop and restart subroutine.

The QUIESCE command suspends system processing until a system restart is initiated by the operator.

The DISPLAY MATRIX command displays on the system console (optionally) the online/offline status of all CPUs, channels, and devices belonging to the system, and real storage configuration status.

The VARY CPU command takes offline, or brings online, a central processor unit. The CPU's channels and device paths go offline, or come online, along with the CPU.

The VARY CHANNEL command takes offline, or brings, online, the specified channel of the indicated CPU. The channel's device paths also go offline or come online.

The VARY PATH command takes offline, or brings online, the specified path to the device through and indicated CPU, channel, and control unit.

The VARY STORAGE command takes offline, or brings online, the specified real storage address locations that can be reconfigured. The device path dependencies analysis subroutine determines for the caller if any operational paths exist to a particular device and whether a particular CPU or channel is used or necessary to access the device.

The stop and restart subroutine halts system processing by placing in the manual state all central processing units other than the one that is executing the routine. The CPU executing the stop/restart routine is put into the wait state with a wait state reason code. The routine will field a restart interrupt on any CPU and resume processing on all CPUs.

Reconfiguration Commands:

QUIESCE

1) Command Processor	IEEMPS03
2) Stop and Restart System	IEESTPRS

DISPLAY MATRIX

1) Command Processor	IEEMPDM

VARY CPU

1) Online/Offline Processor	IEEVCPU
2) CPU Online Wakeup Routine	IEEVWKUP
3) Device Path Dependency Analysis Routine	IEEVDEV
4) CPU Offline Stop Routine	IEEVSTOP
5) Online/Offline Cleanup Processor	IEECLEAN

VARY CHANNEL

1) Online/Offline Command Processor	IEEVCPU
2) Device Path Dependency Analysis Routine	IEEVDEV
3) Test Channel Operative Routine	IEEVTCH
4) Online/Offline Cleanup Processor	IEECLEAN

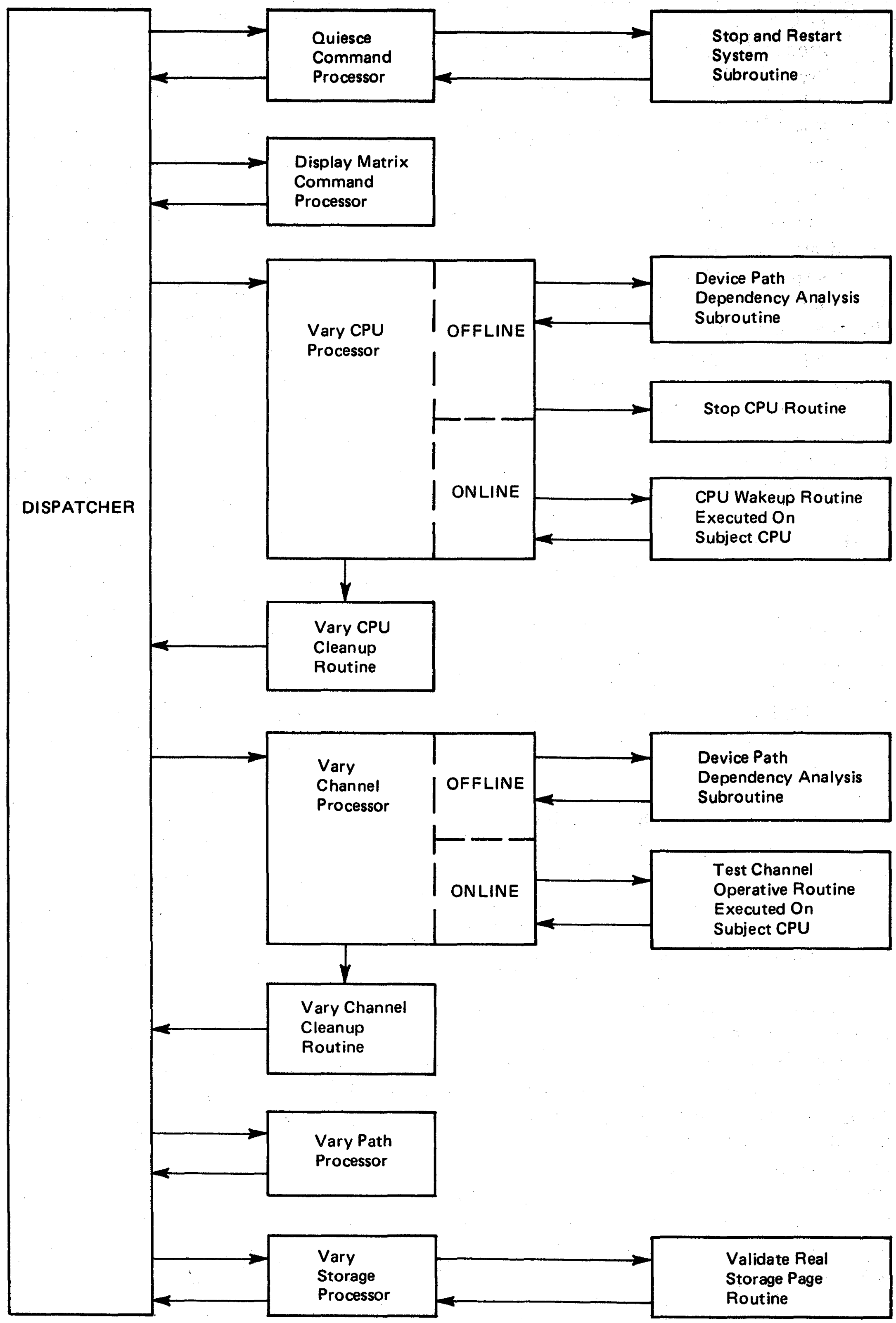
VARY PATH

1) Online/Offline Command Processor	IEEVPTH

VARY STORAGE

1) Online/Offline Command Processor	IEEMPVST
2) Validate Real Storage Page Routine	IEEVALST

Reconfiguration Commands Control Flow



Region Control Task (RCT)

The region control task is the highest priority task in an address space; it is swapped out with the user's task. RCTs:

1. Prepare the address space to be swapped out.
2. Prepare the address space for execution after it has been swapped in.
3. Ensure proper scheduling of an attention exit.

When a new address space is created via a START, LOGON, MOUNT command, RCT's initialization routine receives control from the address space creation routine via the dispatcher. When the system resources manager determines that the address space should be swapped out, RCT's common routine is posted, and it, in turn, routes control to Quiesce. Restore processing prepares the address space so that it can once again execute. If an attention exit is requested, the RCT will ensure its proper scheduling. When the initiator, MOUNT command Processor, or time-sharing session ends, the termination routine receives control, performs housekeeping, and then exits so that the address space can be freed.

Started Task Control (STC)

Started task control processes jobs begun by START, LOGON, or MOUNT commands.

It builds internal JCL for the task associated with the command being processed.

The JCL consists of:

- a JOB statement.
- a PROC statement that contains parameters from the START command not recognized as DD parameters.
- a //IEFPROC.IEFRDER DD statement if the DD parameters were coded on the START command or if this is a MOUNT command.

STC then invokes a special function of the master subsystem to determine if JES is the program being started. If it is JES, much of the processing within STC will be skipped. Equivalent function will be performed by the master subsystem when it is called by the initiator to select the job. If the program is not JES, STC then passes the JCL to JES. JES reads the job, scans and spools the JCL, invokes the converter to transform the JCL to internal text, queues the job, and assigns a job identification, which is returned to started task control.

Started task control invokes an initiator to initiate the task. When the job terminates, control returns to the initiator and then to started task control, which deletes its control blocks.

LOGON Scheduling

When a TSO terminal user enters a LOGON command, Started Task Control (STC), eventually gets control. STC passes control to the LOGON Scheduling Task. Once LOGON Initialization is complete, the LOGON Scheduler invokes the LOGON Prompting Task to parse the LOGON command and validate the LOGON data against the User Attribute Dataset (UADS).

After validity checking successfully completes, the LOGON Prompting Task invokes List Broadcast Dataset (LISTBC) while the LOGON Scheduling Task passes control to the Job Scheduling Subroutine (JSS). JSS will pass control to the Pre-TMP Exit, which will terminate the LOGON Prompting Task after LISTBC has finished and then return back to JSS which will then invoke the TMP (terminal monitor program).

When LOGON or LOGOFF is entered by the terminal user, the TMP will terminate. JSS first passes control to the Post-TMP Exit and then eventually JSS will pass control to the LOGON Scheduling Task. The LOGON Scheduling Task invokes the LOGON Prompting Task who will perform the LOGOFF function and parse the command that was entered at the terminal. For a LOGOFF command, the LOGON Prompting Task will terminate and the LOGON Scheduling Task will perform some cleanup and then pass control to STC. If the command was LOGON, this is referred to as a re-LOGON, the LOGON Prompter will start the validation process all over again.

System Resources Manager (SRM)

The System Resources Manager (SRM) is a component in the MVS control program. The SRM has two principle objectives:

- First, to distribute the system's processing resources among individual address spaces in a way that satisfies the installation's response and turnaround time objectives (as specified in SYS1.PARMLIB member IEAIPSxx).
- Second, to optimize the use of the system's CPU, storage and I/O resources by system users (address spaces). This is primarily a system throughput consideration.

SRM receives control whenever a SYSEVENT (system event SVC) is issued. The various types of SYSEVENTs are enumerated in the SRM Method of Operation section.

The SRM's structure consists of five functional groupings:

- The Interface function is the means through which other system components communicate with the SRM, and through which the SRM requests the services of other system components.
- The SYSEVENT Processor analyzes communications to the SRM and translates them into requests for specific SRM services. It also formulates responses as required by the SYSEVENTs.
- The Workload Manager function is designed to accomplish the first SRM objective. It controls the relative rates of system resource usage (service rates) for address spaces, as specified in the IPS (Installation Performance Specification). It exercises this control by influencing the Control function's swapping decisions.
- The Resource Use Algorithms are designed to accomplish the second SRM objective. They consist of I/O, CPU, and Storage Management functions, which monitor the utilization of these resources, and make address space swapping recommendations that will affect their future use, based on system-wide throughput considerations. Also, a monitoring function exists which samples and adjusts the system multiprogramming level (MPL).
- The Control function performs swapping analyses, obtains swap recommendations from other SRM components, and translates these recommendations into specific swapping decisions. It also requests that previously deferred SRM functions be performed when it is possible to do so.

Incorporated in SRM are functions such as:

- The TSO Driver functions available in OS/MVT.
- Automatic Priority Group control.
- Device Allocation for permanent data sets on mountable volumes.
- Page Replacement algorithms.
- Paging rate control.
- Prevention of real page shortages.
- I/O load balancing.

In addition, two previously supported functions are no longer necessary because of SRM. These are:

- Time slicing as a user option.
- Migration to avoid auxiliary page shortages.

The principle tool used by SRM to meet its objectives is address space swapping. The effectiveness of swapping in meeting the goal of maintaining resource utilization within acceptable levels depends largely on the variety of candidates for swap-in available at the time it is determined to swap out a user - on the advice of the Resource Use Routines. Candidates will be available for swap-in if more address spaces are initiated than can simultaneously fit in storage. Only enough additional initiators should be started to keep the CPU and I/O busy, since starting too many can adversely affect turnaround time.

In addition to address space swapping, the SRM uses three other means to achieve its ends:

- Page stealing (disassociating a page from an address space's working set).
- Address space dispatching priority changes (within the APG).
- Device allocation decisions.

The main control block table for SRM is the Resource Manager Control Table (RMCT) which is contained in IRARMCNS and located by a pointer in the CVT (CVTOPCPT). IRARMCNS also contains the following:

- Control tables used by CPU, I/O, Storage, and Resource management routines.
- Constants used by the Control algorithm to determine the criteria and frequency of swap analysis.
- Entry point descriptor tables for various SRM routines, algorithms, and event initiated actions.

Statistics about each address space are found in its associated user control block (OUCB). Other information is found in the SRM user extension block (OUXB) for swapped-in users. (***Note:*** a mapping of both the OUCB and the OUXB is found in IRARMCNS). If an address space gets swapped out, some of the data in its OUXB is saved in a SRM user swappable block (OUSB) which is built in LSQA and is swapped out with the address space.

More information about some of the constants in IRARMCNS can be found in the SRM section of *OS/VS2 System Library: Initialization and Tuning Guide.*

System Activity Measurement Facility (MF/1)

MF/1, whose execution is optional, can provide information about the CPU, channels, devices, paging, and workload activity of the system.

MF/1 consists of three functional components:

- Measurement facility control (MFC), which includes initialization, control, and termination.
- System activity report generator (SARG), which includes report writers that format and print reports from the information that the measurement gathering routines of SAMG collect.
- System activity measurement gathering (SAMG), whose routines collect information about system activity for routing to the SMF data set and to the report writers.

Job Scheduler

This discussion is divided as follows:

Subsystem Interface
Master Subsystem
Initiator/Terminator
SWA Create
Converter/Interpreter
SWA Manager
Allocation/Unallocation
System Management Facilities (SMF)
System Log
Checkpoint/Restart

Subsystem Interface

The subsystem interface is the means for the control program and data management communicate with the job entry subsystem and the master subsystem. A system routine invokes the interface by issuing the IEFSSREQ macro instruction. Interface control blocks pass control and status information via a SSIB SSOB header and a SSOB functional section from the requestor to a subsystem routine. (The control program uses the interface to inform subsystems about requests for processing.)

Master Subsystem

The master subsystem does not process user jobs. It sets up the environment for starting the master scheduler and subsystems. (In these situations, JES is not available, therefore, the master subsystem performs a subset of the JES functions.)

The subsystem initiation process duplicates the services needed to process JCL input to the converter, to provide converter output — in the form of internal text — to an interpreter, and to provide output — in the form of SWA control blocks — to the initiator.

Initiator/Terminator

JES completely controls job selection. JES selects jobs for initiators and tells initiators when to stop. The initiator builds those control blocks necessary to process a job, obtains a region (when required) for a job, interfaces with device allocation and unallocation, attaches each job step or task in succession, and waits for it to complete processing. When the job step/started task completes, the initiator detaches the job step/started task and frees all associated data areas. When a task or all the steps of a job have completed, the initiator deletes the job from the system and frees all job-associated control blocks.

Converter/Interpreter

The converter operates as a JES subroutine. It reads JCL statements (spooled by JES) and cataloged procedures from SYS1.PROCLIB, and converts the JCL statements into internal text. In addition, converter recognizes and processes all commands in the input stream.

The interpreter processes the internal text data set that the converter creates and uses it to create control blocks in the SWA (scheduler work area) for the job.

SWA Manager

Most of the scheduler control blocks reside on a pageable portion of virtual address space called the scheduler work area (SWA). The scheduler components (and others, including data management) access SWA through a set of routines called the SWA manager.

The SWA manager performs the following functions in both move and locate modes as requested by the calling routine:

- Assign
- Read
- Write
- Delete

Whenever a "write" is performed, the SWA manager invokes the portion of Scheduler Restart that performs job journaling. Job journaling decides if it is to perform in this situation.

Allocation/Unallocation

The major functions device allocation performs are:

1. Locating a requested data set's volume and unit information.
2. Resolving relationships between two or more requests.
3. Creating, via data management, new data sets.
4. Assigning I/O devices to the requests.
5. Instructing the operator to mount necessary volumes.
6. Allowing dynamic concatenation and deconcatenation of data sets.

The major functions device unallocation performs are:

1. Directing the processing of a data set's disposition.
2. Releasing data sets, reserved by an initiator, for use by other job steps.
3. Releasing I/O devices for use by other job steps.

System Management Facilities (SMF)

SMF now provides two more exits than it did in MVT: a job purge exit that can be taken when a job is ready to be purged from the system; and an exit that can receive control before a record is written into the SMF data set.

Checkpoint/Restart (Scheduler Restart)

Checkpoint/restart includes two job scheduler functions that allow a failing job to be restarted or terminated. These functions are the job journal and DSDR (data set descriptor record) data set processing. More specifically, an audit trail of control blocks is maintained for each job so that the job's SWA can be reconstructed for any of the following recovery processes:

- Automatic step restart and checkpoint restart.
- Deferred checkpoint/restart.
- System restart.

Timer Supervision

Timer supervision controls the setting and use of the clocks in the system (CPU timer, clock comparator, and the time-of-day clock).

The timing services routines enable the user to:

- Obtain the date and time of day.
- Measure periods of time.
- Schedule activities for specific times of day or time intervals.

The time-of-day clock provides the system with a means of measuring and maintaining real time.

Timer supervision is also responsible for a portion of job step timing. This function involves the detection of job steps that have violated CPU and wait-time limits.

Supervisor Control

Supervisor Control consists of those routines that:

- Perform the initial processing of interruptions and the routing of control to routines that handle them.
- Dispatch tasks or SRBs to the appropriate address space for execution. (Dispatcher-IEAVEDS0)
- Determine the highest priority address space for which work exists and the processor on which it will be dispatched (Memory Switch - IEAVEMS0 and Dispatcher - IEAVEDS0).
- Provide a common mechanism for communicating with another processor in a tightly-coupled multiprocessing environment. (IPC)
- Provide a common mechanism for serializing serially reusable resources between tasks, SRBs, address spaces, and processors (Lock manager - IEAVELK).

The interrupt handlers routines and their functions are:

- SVC interrupt handler that routes control to the requested SVC routine after acquiring any locks required by that routine. The SVC interrupt handler also acts as the extended SVC router for any *extended SVCs*.
- External interrupt handler that receives control on TOD timer and CPU timer interrupts and routes control to the appropriate timer routines. In a multiprocessing environment, Emergency Signal, Malfunction Alerts, and External Call interrupts are handled by the External interrupt handler prior to routing control to the appropriate routines. Control is routed to a Communication Task routine if an interrupt key external interrupt occurs.
- I/O interrupt handler that routes control to the I/O supervisor for I/O interrupts.
- Program Check interrupt handler that processes program checks, page faults, monitor call, and PER type interrupts and routes control to the appropriate routines.

Note: External, I/O, and Program Check interrupt handlers determine if control is to be returned to the interrupted program or the dispatcher, and routes control appropriately.

- Restart interrupt handler that routes control to RTM or returns to the interrupted program.

The dispatcher has four logical levels of dispatching to service the different dispatchable units of work. These levels are:

—A global SRB (service request block) dispatcher
—A local SRB dispatcher
—A local supervisor dispatcher
—A task (TCB) dispatcher

Dispatching proceeds in the other just indicated; global SRBs are dispatched first and tasks running under a TCB are dispatched last. The service manager uses a dispatching technique that allows internal system functions to run enabled, unserialized, and in parallel on an MP system. (*Note:* SRB routines are interruptable but not pre-emptable. That is, they run enabled but control must pass back to the SRB routine after interrupts.)

The dispatcher also effects the transferring of control to a new memory (memory dispatcher) prior to dispatching any SRBs or tasks in that address space. Although Global SRBs are scheduled to run in a specific address space, those services scheduled at a global priority will be executed first without regard to the priority of the address space. Additionally, the dispatcher saves status of any pre-empted work, including accumulating CPU timing for tasks and SRBs.

Two important concepts to keep in mind are the locking services and IPC (interprocessor communications).

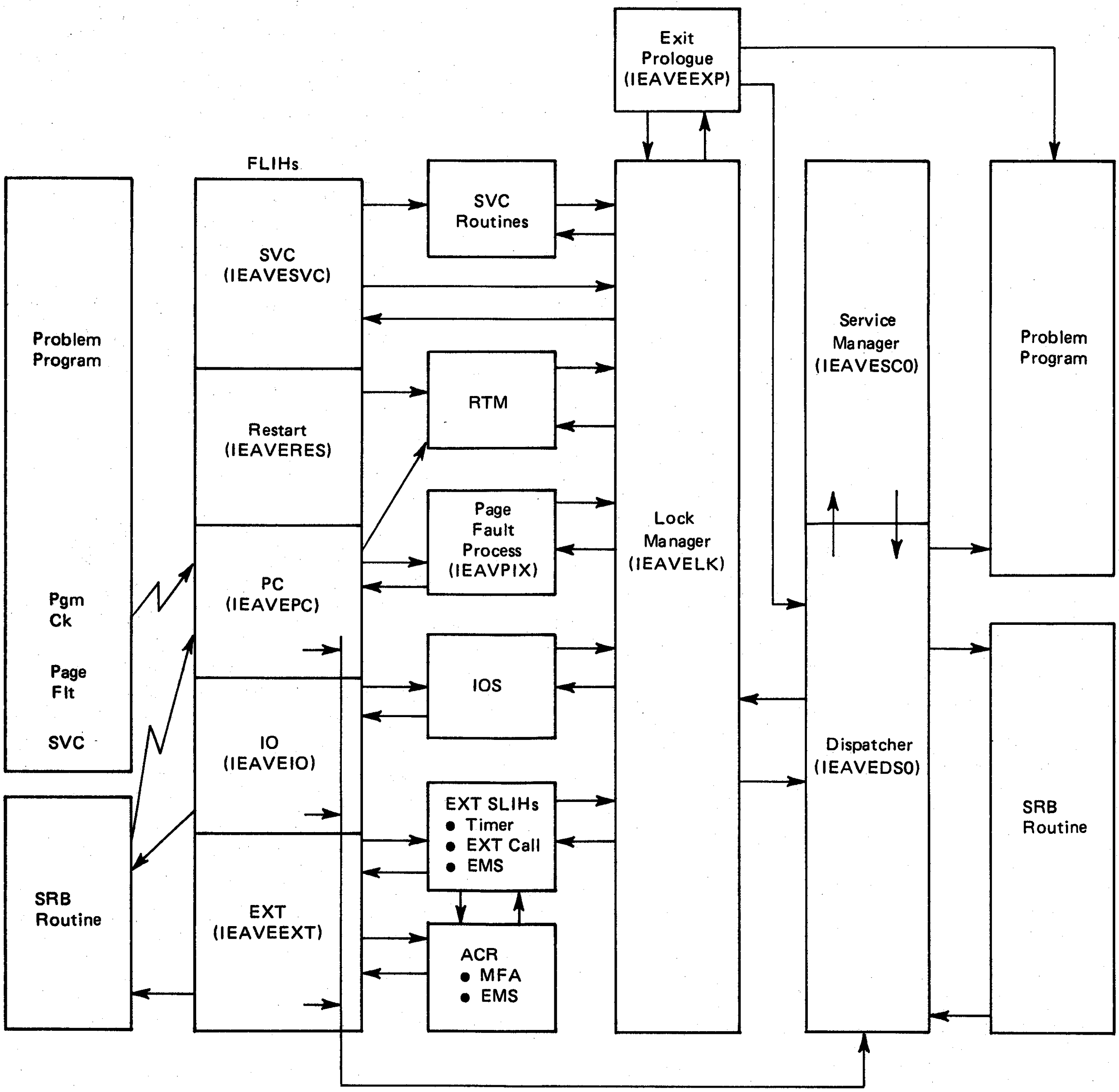
The locking services support the MP and global/local supervisor. These services serialize the use of a resource by multiple tasks and CPUs. (One macro instruction, SETLOCK, is provided.) The locking strategy is to have one local lock per address space for functions that use resources local only to one address space and to have a small set of global locks for discrete function that reference multiple address spaces or resources common multiple address spaces. Local lock functions run physically enabled; global locks run disabled. The CMS is an exception; it is an enabled lock. SETLOCK can be used to get any lock by calling IEAVELK. Type 1 SVCs now run enabled under the local lock, allowing I/O interruptions to be processed when received.

To coordinate MP system activities, such as the reading of data from an I/O device connected to only one CPU, the two CPUs must communicate with one another. This interprocessor communication is accomplished through the use of the IPC routines that use the signal processor (SIGP) instruction. Using a SIGP, one CPU can request the status of the other CPU or it can request the other CPU to execute a program. For example, if one CPU has no channel available, it can request the other CPU to initiate an SIO to a particular device. Some but not all SIGP orders appear as external interruptions.

An example of a user of a SIGP order that appears as an External Interrupt is Memory Switch.

New work may be introduced to the system via the SRB being scheduled or a task being made ready by a function executing under the local lock. Whenever the local lock is released or service manager processes the SRB, memory switch may be called to direct the dispatcher to the highest priority work in the system. Memory switch may in turn call the IPC function to SIGP another processor if it is waiting for ready work to dispatch.

Supervisor Control Flow



Task Management

Task management refers to that collection of supervisor services that create and delete subtasks, control the execution of tasks in one or more address spaces, and provide information services to the requester.

There are several new concepts to keep in mind:

1. Each user is associated with an address space and can address only within that address space. An address space is identified by an ASID (address space identifier) and an ASCB (address space control block).
2. A user's control blocks are kept in the user's address space.
3. POST has the cross-memory option. This service allows the posting of an ECB (event control block) in an address space that the user could not otherwise address.
4. Post exit processing results in control being routed to an authorized user routine when an extended ECB is posted.

Program Management

The program manager creates and maintains the control block queues that define the load modules in virtual storage. The program manager loads modules into the system link pack area (LPA) or into the requestor's job pack area (JPA).

The program manager has both common and specialized functions. The common function (LINK, LOAD, XCTL, and DELETE service routines) satisfy macro instruction requests for linkage to a module, or requests to fetch or delete a module. The specialized service routines are for the IDENTIFY and SYNCH macro instructions. Through IDENTIFY, the program manager is informed of an embedded entry point within a module. The SYNCH macro instruction allows the control program to take synchronous exits to a user-written processing program.

The entry point address of each subroutine is contained in the CVT. These routines are used to search for a module in a region's JPA, active LPA, or in the paged LPA directory.

Recovery/Termination Management (R/TM)

The recovery/termination supervisor subcomponents controls software recovery processing and supplies the system services for normal and abnormal task and address space termination. It also provides recording, SNAP and SDUMP (SVC DUMP) functions.

The recovery objective provides the function and environment to allow recovery processing from abnormal events for control program components, utilities and processors, and all customer applications. Recovery is designed to operate at different levels of control and to satisfy a range of performance objectives. The recovery process is invoked under:

- Program check
- Machine check
- ABEND macro
- CALLRTM macro
- Invalid issuance of an SVC
- I/O error on page-in request
- Restart key

When one of these events occurs, if a recovery exit has been specified, it is invoked causing recovery for the process currently in control. Should that level of control be unable to recover from the incident, should it request termination, or should it fail, the error will be passed (percolated) to the previous level recovery exit. If all recovery routines are unable to recover from the incident, the process is terminated.

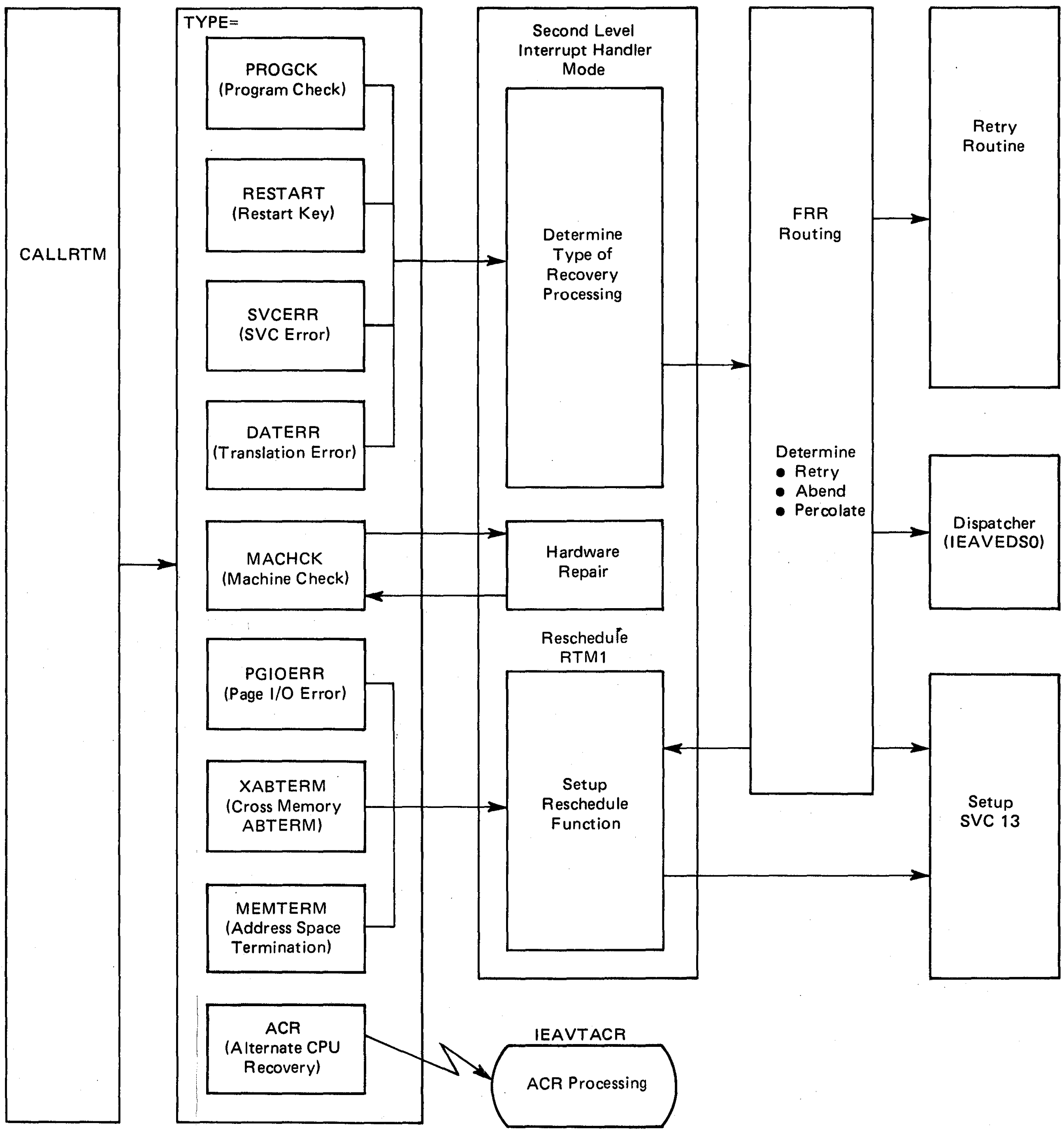
Termination is the process of removing tasks or address spaces from the system and returning allocated resources to an allocatable status. Termination of tasks or address spaces can be either normal or abnormal. Resources are returned to the system by utilizing a resource manager concept.

There are three types of recovery routines:

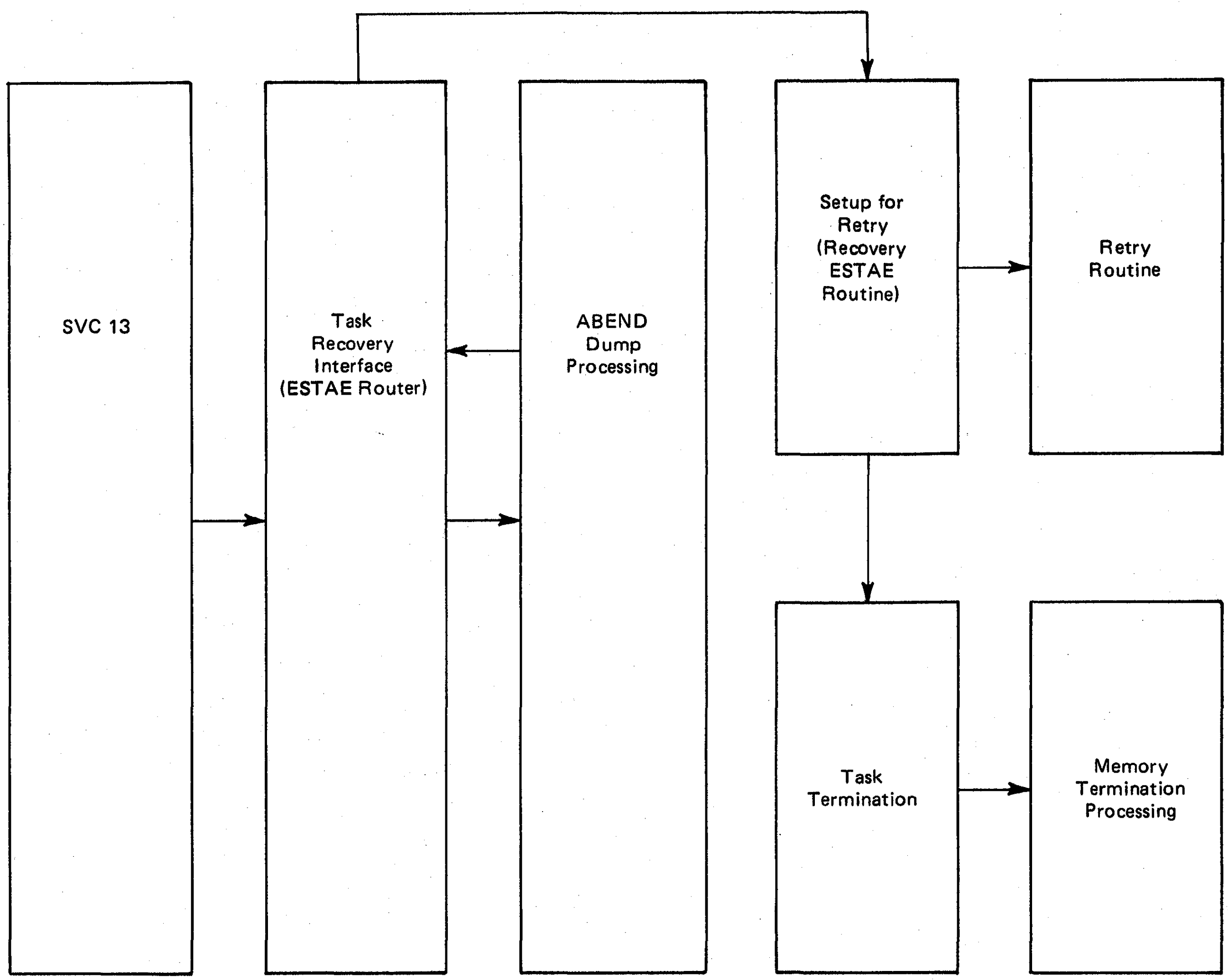
- *Supervisor control program recovery*. These routines handle recovery for critical supervisor routines such as the dispatcher, the lock manager, exit effectors, the first level interrupt handlers.
- *Task recovery routines (STAE/STAI/ESTAE/ESTAI)*. Issuing the STAE or ESTAE macro instructions or the ATTACH macro instruction with the STAI or ESTAI parameters allows the user to intercept a scheduled ABEND. Control is given to a user-specified exit routine where the user can diagnose the cause of the ABEND and specify pre-termination processing or specify a retry address if he wants to prevent termination.

- *Functional Recovery Routines (FRRs)*. These are routines established to protect locked, disabled, or SRB mode routines. The SETFRR macro defines an FRR to RTM. Each FRR is placed in a stack in the system, each stack represents a single path through the control program. When a functional recovery routine is invoked, it will run in the state of the system (enabled or disabled) and with the locks that were held at the time of the error or as modified by previous FRRs.

RTM1


CALLRTM
TYPE=
PROGCK
(Program Check)
RESTART
(Restart Key)
SVCERR
(SVC Error)
DATERR
(Translation Error)
MACHCK
(Machine Check)
PGIOERR
(Page I/O Error)
XABTERM
(Cross Memory
ABTERM)
MEMTERM
(Address Space
Termination)
ACR
(Alternate CPU
Recovery)
Second Level
Interrupt Handler
Mode
Determine
Type of
Recovery
Processing
Hardware
Repair
Reschedule
RTM1
Setup
Reschedule
Function
FRR
Routing
Determine
• Retry
• Abend
• Percolate
Retry
Routine
Dispatcher
(IEAVEDS0)
Setup
SVC 13
IEAVTACR
ACR Processing

RTM2 – Task Mode Processing



Real Storage Management (RSM)

The real storage manager administers the use of a real storage and directs the movement of virtual pages between auxiliary storage and real storage in page size (4096 bytes) blocks. RSM employs the auxiliary storage manager (ASM) to perform the actual paging I/O necessary to transfer pages in and out of real storage.

RSM assigns real storage page frames on request, from a pool of available frames, associating virtual addresses with real storage addresses. Frames are repossessed upon termination of use, when freed by a user, when a user is swapped out, and when needed to replenish the available pool. A page in real storage is considered pageable unless specified otherwise through the PGFIX function or unless the page is used as a system page that must be resident in real storage. RSM also allocates the appropriate frames for V=R regions.

RSM determines the working set size for the SWAP function and maintains the necessary information to remove the virtual pages of an address space from real storage on swap-out and to reestablish those pages on swap-in. ASM provides the real I/O for this operation.

Virtual Storage Management (VSM)

Virtual storage management keeps track of free and allocated storage for both the system and the user. In addition, VSM performs initialization and clean-up functions for the resources it manages. The system interfaces are CREATE/FREE ADDRESS SPACE to initialize and free address spaces; GET/FREE PART to obtain and free regions; CHANGKEY to alter the protection key of virtual storage; VSM initialization to set up storage at IPL time; and VSM task termination to clean up virtual storage at end of task. Non-region storage is provided for in the system queue area (SQA), common service area (CSA), local SQA (LSQA), and the scheduler work area (SWA). Only system functions request non-region storage, but it can be assigned in behalf of either the system or a specific user. Generally, space assigned in the user's behalf resides in LSQA or SWA, and space assigned for the system resides in SQA. Virtual storage management controls the allocation of storage in each area as directed by requesting GETMAIN and FREEMAIN macro instructions.

VS2 uses the subpool interface to indicate the area (CSA, SQA, SWA, LSQA, or region) for which the virtual storage management service is requested.

All functions except GETMAIN or FREEMAIN for SQA and CSA, and GETPART or FREEPART for V=R regions are processed independently and in parallel by addressing space; that is, a GETMAIN can be in progress in more than one address space at a time.

Address Space Create

This subcomponent consists of command processing routines, address space creation and deletion routines, and the system task control routines.

When a new address space is required, a command processing routine invokes the address space creation routine, who notifies the system resources manager (SRM). Based on factors such as priorities and the number of address spaces already existing, SRM decides whether a new address space is desirable. If not, the address space is not created until SRM finds (or makes) conditions suitable. If a new address space is acceptable, the address space creation routine invokes virtual storage management (VSM) to assign the storage. The address space creation routine builds an LSQA (local system queue area) in the address space. The LSQA will contain the page tables and control blocks needed to operate the region control task (RCT) of this address space.

After the address space is dispatched, the RCT attaches an STC (started task control) task. STC, in turn, attaches the task that processes the command specified; for example, for a LOGON command, this is the LOGON Scheduler, and for a MOUNT command, the MOUNT processor.

Storage Protection

Storage protection is a feature that prevents unauthorized access to virtual storage by all except the intended users. The VS2 system uses the 16 standard protection keys allowed by the PSW format and the hardware. A key-0 program has access to all allocated areas of virtual storage. A non-key-0 program has read-only access to the shared areas of the system (for example, the nucleus, the SQA, and the LSQA) and full access to storage in its own address space.

In Release 1, all pageable system programs were key-0. For MVS, protection keys 0-7 are reserved for system programs (leaving keys 8-15 for user programs). The job scheduler and certain logical parts of data management are assigned non-zero protection keys to isolate and protect them from other system functions and from user programs.

The storage protection key assignments for system programs are as follows:

key 1	-	job scheduler (including JES2/JES3)
keys 2-4	-	reserved
key 5	-	data management (including IOS, auxiliary storage management, block processor, and OPEN/CLOSE/End-of-Volume)
key 6	-	TCAM and VSAM
key 7	-	reserved

Note: Where possible, the control blocks and work areas for a non-key-0 system program have a protection key that matches that of the program.

When system functions require that a non-key-0 program modify key-0 data or that this program branch to key-0 programs, the MODESET macro instruction is used to change the non-key-0 program to key-0. All read-only programs reside in key-0 storage, although some of these programs execute with a non-zero key. Any non-refreshable code residing in the MLPA alters its key to zero before modifying itself.

All user programs occupying virtual=virtual (V=V) storage are assigned a storage protection key of 8. V=V user programs, each occupying a different address space, are totally isolated from each other. An address space has access to only its own segment and page tables for translating virtual address to real. Therefore, a user program cannot reference the virtual pages allocated to another user's address space.

User programs occupying virtual=real (V=R) storage (that is, storage where real addresses are the same as virtual addresses) are not protected by the address translation feature and must have unique protection keys. Keys 9-15 are available to these V=R users.

Auxiliary Storage Management (ASM)

Auxiliary Storage Management (ASM) controls all system direct access storage that is set aside for virtual memory paging and for temporary data sets. In MVS, ASM supports the dynamic paging requirements of Real Storage Management (RSM) and the data set storage and retrieval requirements of the Virtual Block Processor (VBP). ASM permits the addressing of storage that is external to memory in a manner that is independent of the physical location of the storage. It does this by a set of device-independent functions on auxiliary storage that separate the management of ASM's physical storage from the management of the data that may be placed into that storage by other components. ASM is the sole interface in MVS for paging I/O.

In addition to its logical I/O abilities, ASM's device-independent interface allows a variety of space-management operations to take place in a way that is transparent to data considerations and the user of the host system. For the collection of auxiliary storage under its control, ASM provides dynamic, centralized allocation of space. ASM does this by: (1) using the fastest storage to keep track of allocated space rather than spreading out allocation information among slower auxiliary storage and (2) allocating auxiliary storage only as and when it is actually needed for data, thus making the most efficient use of its storage.

ASM divides its storage into slots, and a slot or more than one slot contains logical groups (LG). Each logical group has its own identification, the logical group number (LGN). A logical group can be written twice to physically separate slots (duplexed) or written only once to however many slots it takes to contain the logical group (simplexed). The logical groups are further subdivided into logical pages (LP) and each logical page has its own identification (LPID). A set of 16 contiguous logical pages is a logical segment.

For its initialization and operation, ASM depends on the services of NIP and VSAM.

ASM Definitions and Formats

The following definitions and formats are used by ASM.

External Storage

The auxiliary storage managed by ASM is a system pool of storage that is external to virtual and real memory. In MVS, this pool consists entirely and exclusively of direct access space that is made up of VSAM data sets. ASM storage is defined by VSAM and the VSAM catalog keeps track of the storage, but ASM doesn't care what makes up the content of the direct access space.

Permanent Storage Locator Symbol (S)

Because the LGN value is only associated with a logical group while that logical group is known to ASM to be in use or active, the LGN's association with a particular logical group is temporary. When a logical group is saved, the saved version becomes permanent and must be identifiable from the active version which still exists until it is released or the job or system fails. This saved version must have an identifier that is separate from the LGN of the active version. The identifier for the saved version

Real Storage Management (RSM)

The real storage manager administers the use of a real storage and directs the movement of virtual pages between auxiliary storage and real storage in page size (4096 bytes) blocks. RSM employs the auxiliary storage manager (ASM) to perform the actual paging I/O necessary to transfer pages in and out of real storage.

RSM assigns real storage page frames on request, from a pool of available frames, associating virtual addresses with real storage addresses. Frames are repossessed upon termination of use, when freed by a user, when a user is swapped out, and when needed to replenish the available pool. A page in real storage is considered pageable unless specified otherwise through the PGFIX function or unless the page is used as a system page that must be resident in real storage. RSM also allocates the appropriate frames for V=R regions.

RSM determines the working set size for the SWAP function and maintains the necessary information to remove the virtual pages of an address space from real storage on swap-out and to reestablish those pages on swap-in. ASM provides the real I/O for this operation.

Virtual Storage Management (VSM)

Virtual storage management keeps track of free and allocated storage for both the system and the user. In addition, VSM performs initialization and clean-up functions for the resources it manages. The system interfaces are CREATE/FREE ADDRESS SPACE to initialize and free address spaces; GET/FREE PART to obtain and free regions; VSM initialization to set up storage at IPL time; and VSM task termination to clean up virtual storage at end of task. Non-region storage is provided for in the system queue area (SQA), common service area (CSA), local SQA (LSQA), and the scheduler work area (SWA). Only system functions request non-region storage, but it can be assigned in behalf of either the system or a specific user. Generally, space assigned in the user's behalf resides in LSQA or SWA, and space assigned for the system resides in SQA. Virtual storage management controls the allocation of storage in each area as directed by requesting GETMAIN and FREEMAIN macro instructions.

VS2 uses the subpool interface to indicate the area (CSA, SQA, SWA, LSQA, or region) for which the virtual storage management service is requested.

All functions except GETMAIN or FREEMAIN for SQA and CSA, and GETPART or FREEPART for V=R regions are processed independently and in parallel by addressing space; that is, a GETMAIN can be in progress in more than one address space at a time.

Address Space Create

This subcomponent consists of command processing routines, address space creation and deletion routines, and the system task control routines.

When a new address space is required, a command processing routine invokes the address space creation routine, who notifies the system resources manager (SRM). Based on factors such as priorities and the number of address spaces already existing, SRM decides whether a new address space is desirable. If not, the address space is not created until SRM finds (or makes) conditions suitable. If a new address space is acceptable, the address space creation routine invokes virtual storage management (VSM) to assign the storage. The address space creation routine builds an LSQA (local system queue area) in the address space. The LSQA will contain the page tables and control blocks needed to operate the region control task (RCT) of this address space.

After the address space is dispatched, the RCT attaches an STC (started task control) task. STC, in turn, attaches the task that processes the command specified; for example, for a LOGON command, this is the LOGON Scheduler, and for a MOUNT command, the MOUNT processor.

Storage Protection

Storage protection is a feature that prevents unauthorized access to virtual storage by all except the intended users. The VS2 system uses the 16 standard protection keys allowed by the PSW format and the hardware. A key-0 program has access to all allocated areas of virtual storage. A non-key-0 program has read-only access to the shared areas of the system (for example, the nucleus, the SQA, and the LSQA) and full access to storage in its own address space.

In Release 1, all pageable system programs were key-0. For MVS, protection keys 0-7 are reserved for system programs (leaving keys 8-15 for user programs). The job scheduler and certain logical parts of data management are assigned non-zero protection keys to isolate and protect them from other system functions and from user programs.

The storage protection key assignments for system programs are as follows:

key 1	-	job scheduler (including JES2/JES3)
keys 2-4	-	reserved
key 5	-	data management (including IOS, auxiliary storage management, block processor, and OPEN/CLOSE/End-of-Volume)
key 6	-	TCAM and VSAM
key 7	-	reserved

Note: Where possible, the control blocks and work areas for a non-key-0 system program have a protection key that matches that of the program.

When system functions require that a non-key-0 program modify key-0 data or that this program branch to key-0 programs, the MODESET macro instruction is used to change the non-key-0 program to key-0. All read-only programs reside in key-0 storage, although some of these programs execute with a non-zero key. Any non-refreshable code residing in the MLPA alters its key to zero before modifying itself.

All user programs occupying virtual=virtual (V=V) storage are assigned a storage protection key of 8. V=V user programs, each occupying a different address space, are totally isolated from each other. An address space has access to only its own segment and page tables for translating virtual address to real. Therefore, a user program cannot reference the virtual pages allocated to another user's address space.

User programs occupying virtual=real (V=R) storage (that is, storage where real addresses are the same as virtual addresses) are not protected by the address translation feature and must have unique protection keys. Keys 9-15 are available to these V=R users.

Auxiliary Storage Management (ASM)

ASM (Auxiliary Storage Management) keeps track of the auxiliary storage locations of all virtual pages, controls the paging and swapping of virtual pages between real and auxiliary storage, and maintains the necessary copies of VIO data set pages.

There are two callers of ASM: RSM (Real Storage Management) and VBP (Virtual Block Processor). RSM calls ASM to initiate paging and swapping requests; VBP calls ASM to request updates to the information about VIO data sets. The new operator command PAGEADD enables an installation to dynamically add page or swap data sets to the system without having to do another IPL. PAGEADD initializes control blocks so that the new page or swap data set is available to ASM for processing. Figure 1-2 illustrates who communicates with ASM.

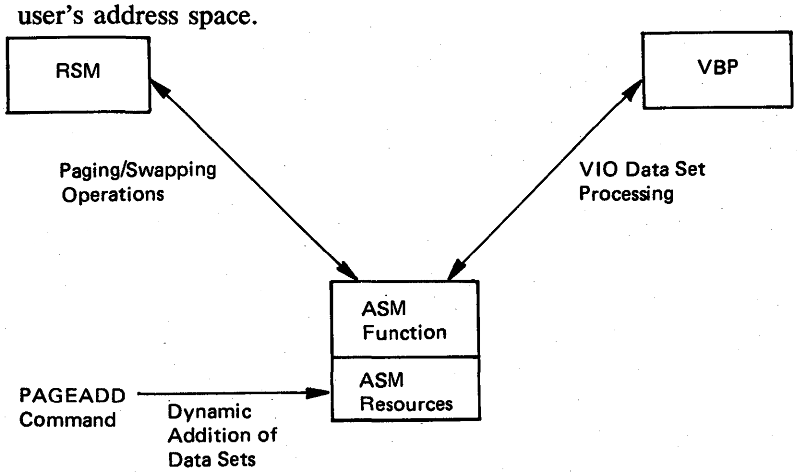


Figure 1-2. ASM Communications

ASM Functions

ASM processing is divided into four major functional sections: I/O Control, I/O Subsystem, VIO Control, and the VIO Group Operators. ASM also includes three other sections: Page Expansion, Recovery, and Service Routines.

I/O Control

I/O Control is the communication link between RSM and the I/O Subsystem for paging requests and between RSM and IOS (I/O Supervisor) for swapping requests. RSM initiates all paging and swapping I/O; the I/O Subystem and IOS execute this I/O. I/O Control accepts the paging/swapping requests from RSM, determines the type of I/O to be done and when it can be started, and notifies RSM when the I/O is complete. I/O Control also records the auxiliary storage locations of all virtual pages.

I/O Subsystem

I/O Subsystem communicates with IOS to cause the physical transfer of data between real and auxiliary storage. It allocates auxiliary storage slots, builds paging channel programs, passes them to IOS for execution, processes I/O completion, and manages the page data sets.

VIO Control

VIO Control coordinates all the ASM processing required to support VIO data sets (called logical groups by ASM). Operations on a logical group are classified as group operations and page operations. VBP initiates group-related operations, and VIO Control passes them to the VIO Group Operators to be processed. RSM initiates page-related operations and I/O Control and VIO Control jointly process them.

VIO Group Operators

The VIO Group Operators maintain the logical group information VBP requires. The VIO Group Operators perform all processing necessary to create, save, restore, and delete a logical group. These operators are invoked only by VIO Control as the result of requests from VBP.

Figure 1-3 illustrates the major functional sections of ASM processing.

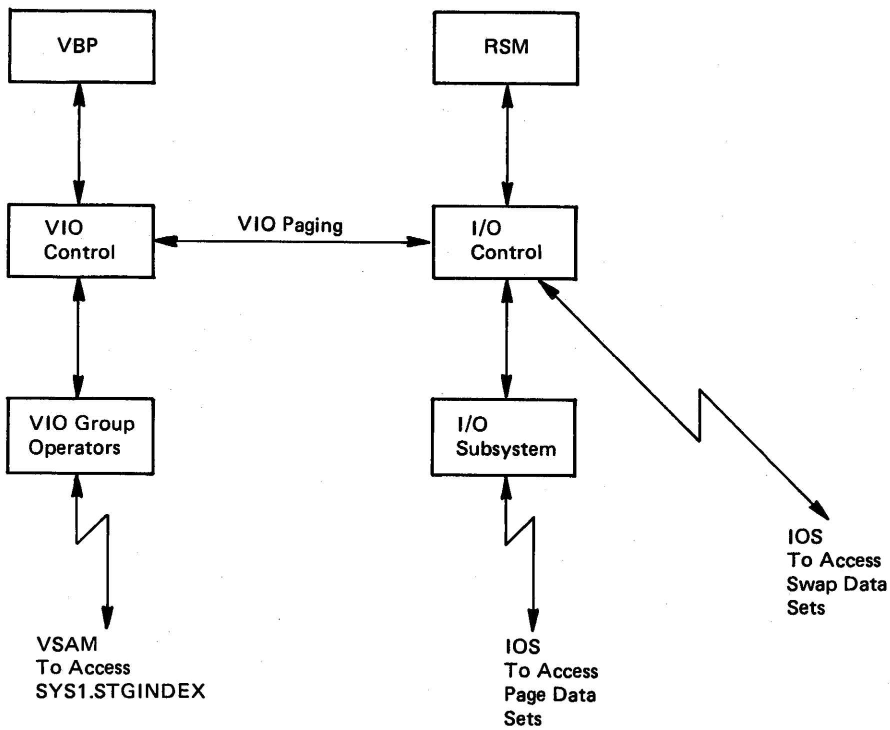


Figure 1-3. Overview of ASM Processing

Recovery

Recovery provides the mechanism to handle any errors that occur during normal ASM processing. Recovery classifies errors into two groups: errors detected during mainline processing, called determinate errors; and errors detected by recovery itself, called indeterminate errors. A determinate error does not prevent continuation of ASM processing. It is recorded in SYS1.LOGREC and processing is resumed. For indeterminate errors, recovery assesses the severity of the error in terms of how much damage it could do to ASM control blocks, to the code, and to the process in progress, and then takes appropriate action. Possible recovery actions include recording the error with module id and ASM status information, clean-up of ASM resources where possible, conversion of the error to a failure indication such as a return code to the caller of ASM, and termination of a task or address space if necessary.

Page Expansion
Page Expansion gives the user the ability to add page or swap data sets to the system without having to do another IPL. This function is available to the installation through the PAGEADD operator command. This command causes the specified page or swap data sets to be opened and the control blocks necessary for ASM processing to be initialized.

Service Routines
Service Routines include: an ASM control block formatting facility, which is invoked by the system dump-printing facility; an address space termination resource manager, whose main function is to reclaim auxiliary storage resources from an address space that is terminating; and a pool extender routine for adding storage to a virtual storage pool.

ASM Serialization
For a description of the serialization of ASM processing, see the Diagnostic Aids section in Volume 7.

ASM Control Blocks

Following is a description of some of the most important ASM control blocks.

ASMVT — Auxiliary Storage Management Vector Table
The ASMVT is ASM's extension to the CVT. It contains all global flags, slot counts, and save areas used by ASM routines. It also contains addresses of the main control blocks (PART, SART, LGVT), the cell pools, the message buffers, and the error record.

PART — Page Activity Reference Table
The PART contains information relating to the page data sets in use. It is managed and used mainly by the I/O Subsystem. The PART consists of a header and an entry (PARTE) for each page data set. The header contains the write request queue. It also contains the addresses of two circular queues of PARTEs, one for page data sets on moveable-head devices and one for page data sets on fixed-head devices. Each PARTE contains flags, a read queue, the address of the appropriate write queue, and addresses of control blocks (including the PAT) related to that page data set.

PAT — Page Allocation Table
The PAT describes the 4096-byte slots on the page data set with which it is associated. The PAT is built when a page data set is opened. It contains a header followed by cylinder maps of the entire data set.

SART — Swap Activity Reference Table
The SART contains information relating to the swap data sets in use. It is used by the swap routines of I/O Control. The SART consists of a header section and entry (SARTE) for each swap data set. The header contains the addresses of two circular queues of SARTEs, one for swap data sets on fixed-head devices and one for swap data sets on movable-head devices. The SARTE contains flags and addresses of control blocks associated with that swap data set.

SAT — Swap Allocation Table
The SAT describes the swap sets (which are multiples of 4096-byte slots) on the swap data set with which it is associated. It contains a header followed by cylinder maps of the entire data set.

IORB — I/O Request Block
The IORB is the control block used by ASM to track I/O requests. It contains the addresses of other control blocks (IOSB, SRB), the address of an 18-word save area for use by IOS, and flags. When a page or swap data set is opened, the IORB is built and chained from the PARTE or SARTE associated with the data set.

LGVT — Logical Group Vector Table
The LGVT is a table of addresses to in-use LGEs (Logical Group Elements). It contains a header and entries (LGVTEs). The header contains the LGVT size and the address of the first available LGVTE. An in-use LGVTE contains the address of the LGE for that logical group and the address space owning the logical group.

ASMHD — Auxiliary Storage Management Header
The ASMHD is address-space related. It is built in SQA immediately following the RSMHD (RSM header). ASMHD is used by I/O Control to manage paging and swapping I/O, and VIO data set operations for each private address space. It contains flags, counts, swap queues, and the queue of LGEs for this address space.

LGE — Logical Group Element
The LGE is the focal point for controlling all operations on a logical group. VIO Control is the principal user of this control block. An LGE is built for each logical group created for VIO. It contains a process queue, flags, counts, and the address of the associated ASPCT.

ASPCT — Auxiliary Storage Page Correspondence Table

The ASPCT is used to record the auxiliary storage locations (called LSIDs, Logical Slot Identifiers) of VIO data set pages. One ASPCT is created for each logical group assigned by ASM. The ASPCT address is maintained in the LGE for the logical group.

Figure 1-4 on the next page depicts an overview of ASM control blocks.

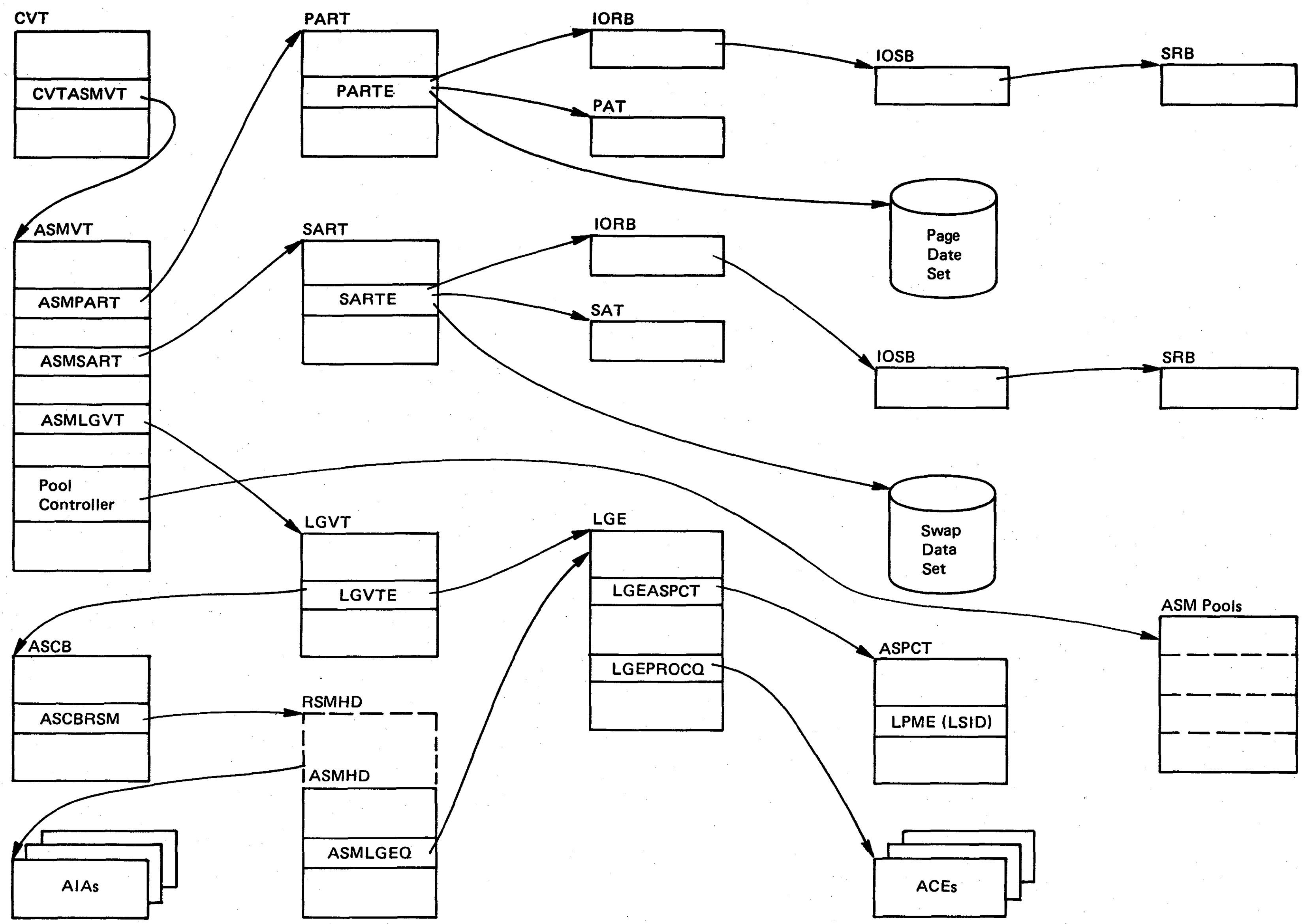


Figure 1-4. Auxiliary Storage/Management Control Block Overview

ASM Data Sets

ASM has three types of direct access data sets that are required for maximum function and performance. These types are: page data sets, swap data sets, and SYS1.STGINDEX (ASM's VIO journaling data set).

Page Data Sets

Page data sets are single VSAM data sets that make up the page space portion of auxiliary storage. They are defined, formatted, and cataloged as VSAM PAGESPACEs. (Further discussion of page data set definition can be found in the *OS/VS2 MVS System Programming Library: System Generation Reference*. A discussion of how to determine the size of page data sets can be found in the *OS/VS2 Initialization and Tuning Guide*.) ASM uses this page space to store the contents of pageable virtual pages and VIO data set pages. Each data set is formatted in 4096-byte records called slots. A slot is allocated dynamically by ASM whenever a page must be moved out of real storage.

ASM has four page data set classifications, based on data set content and use. The four types of page data sets are:

- PLPA (Pageable Link Pack Area) Page Data Set.
- Common Page Data Set.
- Duplex Page Data Set (optional).
- Local Page Data Sets.

There is only one PLPA data set, one Common data set, and, if specified, one Duplex data set. One page data set of each type (except the Duplex data set) must be supplied to ASM at IPL. All page data sets are named by the installation. A maximum of 64 page data sets is allowed.

PLPA, Common, and Duplex Page Data Sets The PLPA page data set contains the pageable LPA pages of the system, the TPARTBLE, and the Quick Start Record (QSR). This page data set is filled during a cold start IPL and becomes effectively a read-only data set once PLPA has been built, the only exception being overflow from the Common page data set described later in this section.

The Common page data set provides space for the non-PLPA virtual pages in the system common area.

The Duplex page data set is an optional data set supplied by the user when a secondary copy of the common area is desired.

ASM provides for overflow from the PLPA data set to the Common data set if the PLPA data set becomes full during IPL. If the Common data set should ever become full and PLPA is not full, it will overflow to the PLPA data set. If both data sets should become full, ASM will not overflow to Local data sets. If duplexing is active, system operation continues as long as the Duplex data set is operational. If duplexing is not active or if the Duplex data set is not operational, system operation is terminated. (For more information, see the Error Action Matrix in Diagnostics Aids in Volume 7.)

Local Page Data Sets The Local page data sets provide space for each private address space's unique pages, VIO data sets, and LSQA pages if there are no Swap data sets available.

To ensure that there are enough pages for the current system workload, ASM reserves a number of slots on the Local page data sets for each address space and each VIO data set as they are created. This "reserve" is not an actual allocation of slots, but an anticpation of the number of auxiliary slots an address space or VIO data set will require at some future time. This is done to prevent the system from becoming overloaded. When there are no more page data set slots available to be reserved the system will not allow new address spaces or new VIO data sets to be created.

In order to monitor page data set slot usage, ASM maintains a series of slot counts for its own use and for use by SRM (System Resources Manager) and MF/1 (Measurement Facility 1). For each data set, ASM maintains in its related PART a count of slots available for allocation (PARESLTA) and a count of the bad slots (PARERRCT). Additionally, the following system-wide counts are maintained in the ASMVT:

ASMSLOTS	—	The total number of slots in all open Local page data sets.
ASMERRS	—	The total number of bad slots detected by ASM on Local page data sets.
ASMVSC	—	The total number of slots allocated to VIO data set pages that are paged out to Local page data sets.
ASMNVSC	—	The total number of slots allocated to private area (non-VIO) pages that are paged out to Local page data sets.

The following two counts are subsets of the above global counts. They are maintained on an address space basis in each ASCB.

ASCBVSC	—	The total number of slots allocated to VIO data set pages for each address space.

ASCBNVSC — The total number of slots allocated to private area (non-VIO) pages for each address space.

Swap Data Sets

Swap data sets are single VSAM data sets that make up the swap space portion of auxiliary storage. They are defined, formatted, and cataloged as VSAM PAGESPACEs. (Further discussion of swap data set definition can be found in the *OS/VS2 MVS System Programming Library: System Generation Reference*. A discussion of how to determine the size of swap data sets can be found in the *OS/VS2 Initialization and Tuning Guide*,.) The swap data sets are formatted in 4096-byte slots, but ASM utilitizes them in groups of 12 slots. These groups are referred to as swap sets.

Swap space is used by ASM to store and retrieve the set of LSQA pages belonging to an address space. LSQA pages are critical to an address space since they must be in real storage before any other processing can occur. The swap data sets are designed to process these pages quickly. Thus they are written and read in groups (swap sets) instead of as individual pages.

Swap data sets are optional. If none are supplied swapping is done to Local page data sets, which may degrade swap performance. A maximum of 25 swap data sets is allowed.

SYS1.STGINDEX

The SYS1.STGINDEX data set is a key-sequenced VSAM data set used to store information (ASPCTs) for journaled VIO data sets. It is made up of fixed length control intervals (in the data portion), each containing one keyed record. The key consists of 12 bytes, the first eight of which are assigned by ASM for each ASPCT. The next four bytes of the key are used if ASPCT extensions have been created and must be written to SYS1.STGINDEX. ASM assigns keys in such a fashion as to take advantage of VSAM techniques for reclaiming space freed by deleted records.

ASM Initialization makes use of the REUSE option of VSAM to clear SYS1.STGINDEX when the IPL type is other than warm start. SYS1.STGINDEX must be defined and cataloged before an IPL is performed. However, if ASM initialization is unable to open the data set, system initialization is not cancelled. A message is written to the console informaing the installation that SYS1.STGINDEX is not operational. VIO journaling-related request will be rejected. If a warm start is in progress, ASM informs the operator of any problem and does not continue processing in quick start (CVIO) mode until so directed by the operator. In taking these actions, ASM's objective is to allow the system to become operational in order that the installation may diagnose and correct the problem with the data set before doing another IPL.

Data Set Usage

Each page and swap data set must be defined and cataloged on a direct access storage device (DASD) before being supplied to ASM via the PAGE and SWAP system parameters. It is recommended that page and swap data sets be allocated from clean DASD volumes or a volume containing the necessary amount of contiguous free space. All page and swap data sets opened by ASM Initialization remain open for the life of the IPL. They may not be closed or varied off-line. Additional page and/or swap data sets can be added to the system via the PAGEADD console command without having to re-IPL the system.

Abbreviation List

ABDA (ABDUMP work area)
ABDPL (ABDUMP parameter list)
ABDUMP (snap dump)
ABDUMP parameter list (ABDPL)
ABDUMP work area (ABDA)
ABEND (abnormal end)
ABP (actual block processor)
ACA (auxiliary storage management control area)
ACB (access method control block)
ACE (auxiliary storage management control element)
ACR (alternate CPU recovery)
ACT (account tables)
ACR (alternate CPU recovery)
AFC (available frame count)
AFM (allocation function map)
AFMP (allocation function map)
AFQ (available frame queue)
AIA (auxiliary storage management I/O request area)
AIE (auxiliary storage management I/O control element)
ALCWA (allocation work area)
AMB (access method block)
AMBL (access method block list)
AMWA (access method work area)
APF (authorized program facility)
APG (automatic priority group)
AQE (allocated queue element)
AQMRB (allocation queue manager request block)
ASCB (address space control block)
ASID (address space identifier)
ASM (auxiliary storage manager)
ASMI (auxiliary storage management interface routine)
ASMM (auxiliary storage management monitor routine)
ASMRL (auxiliary storage management request list – main work queue)
ASMVT (auxiliary storage management vector table)
ASPCT (address space page correspondence table)
ASVT (address space vector table)
ASWA (allocation ESTAE work area)
ASXB (address space extension block)
ATB (attention table entries)
ATCOM (allocation/termination communications area)

BASEA (master scheduler resident data area)
BCMSG (SYS1.BRODCAST data set)
BRKELEM (break element)
BUFC (buffer control block for VSAM)

CAT (channel availability table)
cathode ray tube (CRT)
CCA (configuration control array)
CCH (channel check handler)
CCT (CPU control table)
CCW (channel command word)
CDE (contents directory entry)
CIB (command input buffer)
CIWA (common internal work area)
CMS (cross memory services)
CPA (channel program area for VSAM)
CPAB (cell pool anchor block)
CPID (cell pool identifier)
CPU (central processing unit)
CPU control table (CCT)
CQE (console queue element)
cross memory services (CMS)
CRT (cathode ray tube)
CSA (common service area)
CSCB (command scheduling control block)
CSD (common system data area)
CSECT (control section)
CSOA (command scan output area)
CSPL (command scan parameter list)
CSW (channel status word)
CVT (communications vector table)
CXSA (comm task SVRB extended save area)

DASD (direct access storage device)
DAT (dynamic address translation)
DCB (data control block)
DCM (display control module)
DD (data definition)
DD name (data definition name)
DDR common control block (DDRCOM)
DDR (dynamic device reconfiguration)
DDRCOM (DDR common control block)
DEB (data extent block)
DECB (data event control block)
DEVNAMT (device name table)
DEVTAB (device table)
DIDOCS (device independent display operator console support)
DOM (delete operator message)
DOMC (delete operator message control block)
DPL (delete operator message parameter list)
DQE (descriptor queue element)
DSAB (data set association block)
DSCB (data set control block)
DSDR (data set descriptor record)
DSDRP (data set descriptor record processing)
DSENQ (data set enqueue table)
DSNT (data set name table)
DSP (device support processor)
DSS (Dynamic Support System)

EBCDIC (extended binary coded decimal interchange code)
EC (extended control)
ECB (event control block)
ECT (environment control table)
EDB (extent definition block)
EDL (eligible device list)
EDT (eligible device table)
EED (extended error descriptor)
EIL (event indication list)
EP (entry point)
EPA (external parameter area)
EPAL (external parameter area, locate mode)
EPAM (extermal parameter area, move mode)
EPFP (extended precision floating point)
ERP (error recovery program)
ESD (external symbol dictionary)
ESDID (external symbol dictionary ID)
ESR (extended SVC router)
ESSLIH (emergency signal SLIH)
ESTA (extended STAE parameter list, ESTAE-ESTAI-ESTAR services)
ESTAE-ESTAI-ESTAR services (ESTA)

FBQE (free block queue element)
FLIH (first level interrupt handler)
FOE (fix ownership element)
FQE (free queue element)
FRR (functional recovery routine)

FRRS (functional recovery routine stack)

GDA (global data area)
GFA (general frame allocation)
GMT (Greenwich Mean Time)
GSDA (global system duplex area)
GTF (Generalized Trace Facility)

HC (hardcopy)

ICB (interruption control block)
ICT (input/output control table)
ID (identifier)
IEDQTCX (TCAM CVT extension)
IEFPARAM (initiator parameter list)
INVT (initial NIP vector table)
I/O (input/output)
IOB (input/output block)
IOC (input/output complete)
IOCX (input/output supervisor communication extension table)
IOMB (input/output management block)
IOS (input/output supervisor)
IOSB (input/output supervisor block)
IPL (initial program loader)
IPC (interprocessor communication)
IPS (installation performance specification(s))
IQE (interruption queue element)
IRB (interruption request block)

JACT (job account table)
JCL (job control language)
JCLS (job control language string)
JCT (job control table)
JES (job entry subsystem)
JESCT (job entry subsystem control table)
JFCB (job file control block)
JSCB (job step control block)
JSEL (job scheduling entrance list)
JSOL (job scheduling options list)
JSR (journal service routine)
JSTCB (job step TCB)
JSWA (job scheduling work area)
JSXL (job scheduling exit list)

K (1024 bytes)

LCA (log control area LCE (LOGON communication element)
LCCA (logical configuration communications area)
LCT (linkage control table)
LDA (local data area)
LG (logical group)
LGID (logical group identifier for ASM)
LGN (logical group number)
LGVT (logical group vector table)
LPA (link pack area)
LPDE (link pack directory entry)
LPID (logical page identifier)
LRB (LOGREC record block)
LSID (logical slot identifier for ASM)
LSPL (local service priority list)
LSQA (local system queue area)
LWA (LOGON work area)

MAXPN (maximum RPN value declared or implied for a logical group)
MC (monitor call)
MCH (machine check handler)
MCHEAD (monitor call tables head)
MCRWSA (monitor call routing work/save area)
MCS (multiple console support)
MCT (main storage control table)
MEL (merge entrance list)
MFA (malfunction alert)
MIH (missing interruption handler)
MLPA (modified link pack area)
MLWTO (multiple line write-to-operator)
MP (multiprocessing)
MPL (monitor parameter list)
MQE (monitor queue element)
MSRDA (master scheduler resident data area)
MSS (mass storage system)
MSSC (mass storage system communicator)
MUG (multi-unit generic)

NEL (interpreter entrance list)
NIP (nucleus initialization program)
NIPM (nucleus initialization program mainline function or routine)
NIP parameter address table (PARMTAB)
NIP parameter area (NIPPAREA)
NIP vector table (NVT)
NIPPAREA (NIP parameter area)
NIP parameter area (NIPPAREA)
NVT (NIP vector table)

OLTEP (online test executive program)
OPSVT (system resources manager performance specification vector table)
ORE (operator reply element)
OS/VS2 (Operating System with Virtual Storage 2)
OUCB (system resources manager user control block)
OUXB (system resources manager user extension block)

PARMTAB (NIP parameter address table)
PART (paging activity reference table)
PAT (page assignment table)
PCB (page control block)
PCCA (physical communications configuration area)
PCCAVT (PCCA vector table)
PCCB (private catalog control block)
PCI (program-controlled interrupt)
PCT (performance characteristics table)
PDI (passed data set information)
PDS (partitioned data set)
PER (program event recording)
PFK (program function key)
PFT (page frame table)
PFTE (page frame table entry)
PGT (page table)
PGTE (page table entry)
PICA (program interruption communications area)
PIE (program interruption element)
PIOP (page I/O post)
PIRL (purge I/O request list)
PLC (program level control)
PLH (placeholder)
PLPA (pageable link pack area)
PPT (program properties table)
PRB (program request block)
PRLIST (permanently resident reserved list)
PSA (prefixed storage area)
PSLIST (public/storage list)

PSW (program status word)
PVT (page vector table)
PWA (processor work area)

QCB (queue control block)
QDB (queue descriptor block)
QEL (queue element)
QMPA (queue manager parameter area)
QSR (quick start record)

RB (request block)
RBA (relative byte address or relative block address)
RBN (real block number)
RCA (RSM recovery communication area)
RCB (recording control block)
RCBSRB (recording task SRB)
RCT (region control task, routing control table)
RCTD (region control task data area)
RDCM (resident display control module)
RIM (resource initialization module)
RLCT (system resources manager logical Channel table)
RLD (relocation dictionary)
RMCT (system resources manager control table)
RMEP (system resources manager algorithm entry point block)
RMPL (system resources manager parameter list)
RMPT (system resources manager parameter table)
RMS (recovery management support)
RMTR (resources manager termination routine)
RPL (request parameter list)
RPN (relative page number)
RRPL (recovery routine parameter list)
RSM (real storage management)
RSMHD (real storage management header)
RSN (relative slot number)
RSVT (recovery stack vector table)
RTCT (recovery/termination control table, recovery/termination management control table)
RTCTECB (recording task ECB)
R/TM (recovery/termination management)
RRPL (recovery routine parameter list)
RTCA (recovery/termination communication area)
RTMCB (recovery/termination management control block)
RTMCT (recovery/termination management control table, recovery/termination control table)
RMT1 work area (RTM1WA)
RTM1WA (RMT1 work area)
RMT2 work area (RTM2WA)
RTM2WA (RMT2 work area)
RVT (recovery management support secondary communications vector table)

S/370 (System/370)
SALLOC (storage allocation)
SCA (SPIE control area)
SCB (STAE control block)
SCT (step control table)
SCVT (secondary commmunications vector table)
SDT (start descriptor table)
SDWA (STAE diagnostic work area)
SGT (segment table)
SGTE (segment table entry)
SIC (system-initiated cancel)
SIRB (system interrupt request block)
SLIH (second level interrupt handler)
SLIP (serviceability level indication process)
SLOT (scheduler look-up table)
SMCA (system management control area)
SMF (system management facilities)
SPCT (swap control table)
SPE (system parameter element)
SPIE (specify program interruption element)
SPIE control area (SCA)
SPL (service priority list)
SPQE (subpool queue element)
SQA (system queue area)
SQS (system queue space)
SRB (service request block)
SRM (system resources manager)
SSCVT (subsystem communications vector table)
SSIB (subsystem identification block)
SSOB (subsystem options block)
SSVT (subsystem vector table)
STAE (set task asynchronous exit)
STAE control block (SCB)
STAE exit parameter list (STEPL)
STAX (set terminal attention exit)
STC (started task control)
STCINRDR (started task control internal reader)
STEPL (initiator/terminator STAE exit parameter list)
SVA (SWA virtual address)
SVC (supervisor call)
SVRB (supervisor request block)
SWA (scheduler work area)
SWA virtual address (SVA)
SWAP control table (SPCT)
SYSGEN (system generation)

TAIE (terminal attention interrupt element)
TAXE (terminal attention exit element)
TCAM (telecommunications access method)
TCAM CVT extension (IEDQTCX)
TCB (task control block)
TCTIOT (timing control table I/O table)
TCWA (TOD clock work area)
TDCM (pageable display control module)
TFRRPARM (timer functional recovery routine parameter list)
TIOC (terminal input-output coordinator)
TIOC reference pointer table (TIOCRPT)
TIOCRPT (TIOC reference pointer table)
TIOT (task input-output table)
TLB (translation lookaside buffer)
TOD (time-of-day)
TOD clock work area (TCWA)
TPC (timer work area)
TQE (timer queue element)
TRE (track request element)
TSB (terminal status block)
TSO (time sharing option)
TSOINRDR (time sharing option internal reader)

UADS (user attribute data set)
UCB (unit control block)
UCM (unit control module)
UCME (unit control module entry)
UCMI (unit control module identifer)
UIC (unreferenced interval count)
UPT (user profile table)

V=R (virtual equals real)
VAT (virtual address table)
VBN (virtual block number)
VBP (virtual block processor)
VIO (virtual input/output)
VM&V (volume mount and verify)
VSAM (virtual storage access method)
VSL (virtual subarea list)

VSM (virtual storage management)
VS2 (see OS/VS2)
VTAM (virtual teleprocessing access method)
VTOC (volume table of contents)
VUT (volume unload table)

WMPGV (performance group vector table)
WMST (workload manager specification table)
WPL (WTO, WTOR, MLWTO, WTP parameter list definition block, write parameter list)
WQE (write queue element)
WSAG (global work/save vector table)
WSAVT (work/save area vector table)
WSCT (workload manager control table)
WTO (write-to-operator)
WTOR (write-to-operator with reply)
WTP (write-to-programmer)
WWB (write wait block)

XCSLIH (external call second level interrupt handler)
XPT (external page table)
XPTE (external page table entry)
XSA (extended save area)

Index

INDEX

Master Index

OS/VS2
System Logic Library
Volume 1
SY28-0761-0

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ______________________________

Number of latest Technical Newsletter (if any) concerning this publication: ______________

Please indicate your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut or Fold Along Line

Fold | Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold | Fold

OS/VS2 System Logic Library Volume 1 (S370-36) Printed in U.S.A. SY28-0761-0

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)