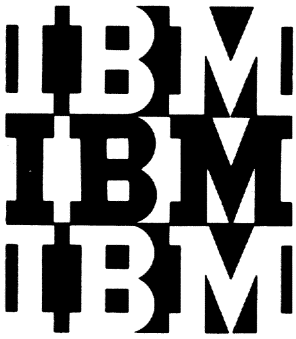


File No. S360-48
GY28-2022-2

Program Logic

Version 8.1

IBM System/360 Time Sharing System

Time Sharing Support System

This publication describes the internal logic of the IBM System/360 Time Sharing Support System (TSSS), a system-recovery tool for the IBM System/360 Time Sharing System (TSS/360). The intended audience is those personnel involved in TSSS maintenance and those system programmers involved in altering the design of TSSS.

TSSS services allow the system programmer to gather data for analyzing system software errors and to correct these errors dynamically. TSSS also provides services for monitoring and testing TSS/360.

This publication begins with an Introduction. The main body of the text provides module descriptions for Environment (interface with TSS/360), Language (processing TSSS commands), and I/O (input and output) routines. Flowcharts give greater detail of the program logic than provided in the body of the publication. Appendixes contain additional detailed material for reference.

Prerequisite Publications

The reader should be familiar with the information contained in:

IBM System/360 Time Sharing System: Time Sharing Support System, GC28-2006.

Third Edition (September 1971)

This is a major revision of, and makes obsolete, GY28-2022-1 and Technical Newsletters GN28-3154 and GN28-3122. The flowcharts and module directory have been updated and reorganized and a number of editorial changes have been incorporated.

Changes to pages are indicated by a bar (|) in the margin at the left of the text. The bar indicates that the adjacent text contains a change.

This edition is current with Version 8, Modification 1, of the IBM System/360 Time Sharing System (TSS/360), and remains in effect for all subsequent versions or modifications of TSS/360 unless otherwise noted. Significant changes or additions to this publication will be provided in new editions or Technical Newsletters. Before using this publication, refer to the latest edition of IBM System/360 Time Sharing System: Addendum, GC28-2043, which may contain information pertinent to the topics covered in this edition. The Addendum also lists the editions of all TSS/360 publications that are applicable and current.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 printer using a special print chain.

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to the IBM Corporation, System/360 Time Sharing System Programming Publications, Department 643, Neighborhood Road, Kingston, New York 12401.

PREFACE

This publication describes the internal organization and operation of the System/360 Time Sharing Support System (TSSS). The information in this document is directed to those service personnel and system programmers responsible for maintaining TSSS. These individuals are assumed to be thoroughly familiar with the System/360 Time Sharing System (TSS/360).

This publication consists of four major parts:

- An Introduction, which describes the function and internal structure of TSSS as a whole, including its interface relationships with TSS/360. This section introduces the concept of dual structuring between a resident support system and a virtual support system, as well as describing the qualifications of a TSSS user. In order to be acceptable to the system, the user requires system programmer authority (authority codes O and P).

- A Body, which is divided into three functional groups: Environment, Language, and I/O, each consisting of introductions and module descriptions. Each division is discussed in terms of the functions it performs. This section also discusses common data, such as tables, control blocks, and work areas, but only to the extent required to understand the logic of the modules themselves. The Environment (RSS and VSS) describes the program logic involved in interfacing with TSS/360, with particular regard to interruption handling, and in maintaining internal control for the entire component. Language, in describing the processing of the TSSS commands, shows the means of interfacing with the TSSS user. TSSS I/O is described in the third group. This publication, in discussing each module that implements a specific division's function, provides a frame of reference for the comments and coding supplied in the program listing.

- The Flowcharts, which provide a logic description of greater detail than that found within the second section. Because this publication represents a complete system, the flowcharts are presented as a comprehensive unit, and, accordingly, are drawn to present the system logic as a single entity.

- The Appendixes, which contain additional material for reference. In particular, the common data is described here in greater detail than within the body of the manual.

If more detail about the programming techniques used is required, the reader should refer to the comments, remarks, and coding in the TSSS program listings.

PREREQUISITE PUBLICATIONS

The external characteristics of TSSS and the syntax and use of its language are defined in the Systems Reference Library publication IBM System/360 Time Sharing System: Time Sharing Support System, GC28-2006. The reader is assumed to be throughly familiar with the contents of that publication as the two TSSS publications are directed to the same audience.

Knowledge of the information in the following publications is required for a full understanding of this manual:

IBM System/360 Principles of Operation, GA22-6821
IBM System/360 Time Sharing System: Concepts and Facilities, GC28-2003
IBM System/360 Time Sharing System: System Logic Summary, Program Logic Manual, GY28-2009

In addition, the following publications are essential to a complete understanding of the interrelationships described in this manual, and are available for reference:

IBM System/360 Time Sharing System: Resident Supervisor, Program Logic Manual, GY28-2012
IBM System/360 Time Sharing System: Task Monitor, Program Logic Manual, GY28-2041
IBM System/360 Time Sharing System: System Control Blocks, Program Logic Manual, GY28-2011
IBM System/360 Time Sharing System: System Programmer's Guide, GC28-2008
IBM System/360 Model 67 Functional Characteristics, GA27-2719

CONTENTS

ILLUSTRATIONS

The Time Sharing Support System (TSSS) is a system-recovery tool for the IBM System/360 Time Sharing System (TSS/360); it allows the system programmer to gather data for analyzing system software errors and to dynamically correct those errors. TSSS also provides services that the system programmer can use to monitor and test TSS/360.

TSSS resides within TSS/360, but TSSS operation is nearly independent of TSS/360. Conversely, when TSSS is not in use, TSS/360 executes without TSSS participation. Points of interaction between TSS/360 and TSSS are pointed out in this publication.

A DUAL SYSTEM: RSS AND VSS

TSSS comprises two systems. As Figure 1 shows, TSSS consists of the Resident Support System (RSS) and the Virtual Support System (VSS), which share a control nucleus. An overview of TSSS is shown in Figure 1.

The TSSS control nucleus is loaded with the TSS/360 Resident Supervisor by TSS/360 Startup. The control nucleus processes interruptions and activates either RSS or VSS, as requested. When RSS is activated, TSS/360 execution is suspended. VSS, which is part of each task's initial virtual storage (also called Initial Virtual Memory), is activated within a specified task, and only the execution of that task is suspended during VSS execution.

RSS has independent language-processing and input/output routines that enable it to operate as a stand-alone, non-time-shared program. VSS executes in virtual storage with similar language processing and I/O routines.

RSS is dependent only on hardware and on a minimal interface with TSS/360, while VSS is dependent on the TSS/360 Resident Supervisor and a part of the Task Monitor. VSS is independent of other virtual storage programs, both privileged and non-privileged. RSS is not time-sliced, whereas VSS executes within a task that is time-sliced. However, VSS can call upon RSS to perform certain functions that it cannot perform for itself; RSS is activated and TSS/360 execution is suspended while the function is performed.

RSS includes the TSSS control nucleus, which comprises the routines that are permanently resident in real storage. The remainder of RSS is transient (resident on the TSS/360 auxiliary paging volume) and is dynamically paged when needed through use of the paging exception program interruption.

Interruption handling by the control nucleus is usually conducted with dynamic address translation (DAT) inactive (bit 5 of the extended PSW is 0). DAT is inactive for the VSS activation and deactivation performed by the control nucleus. However, when RSS is being activated, DAT is made active, and the remainder of RSS executes with DAT active. The transient RSS modules are regarded as "virtual" in order to maintain the addressability of RSS. When a paging exception occurs, causing a page of RSS to be paged in, and during subsequent processing, DAT must be active in order to translate these virtual or transient addresses into their real main storage counterparts.

RSS executes in supervisor state, employing only one CPU. (In a duplex system the other CPU is placed in wait state.) VSS executes in privileged mode.

TSSS USERS

Only system programmers (authority codes O and P) may use TSSS. For convenience here, the TSSS user is referred to as a system programmer, which is sometimes abbreviated SP. A user of RSS is called a master system programmer (MSP). He is connected to the system via the external interruption key on a CPU control panel (2150 console). There can be only one MSP connected at a given time. A user of VSS is called a task system programmer (TSP). He is connected to the system via the VSS command or via MSP intervention. There can be only one TSP connected and associated with a given task at a given time; the total number of TSPs may be as large as the number of current conversational tasks.

TSSS SERVICES

The TSSS command language provides a variety of services that the system programmer may use. Whether he is the MSP or a TSP makes little difference in regard to the service and how it is requested through the command language.

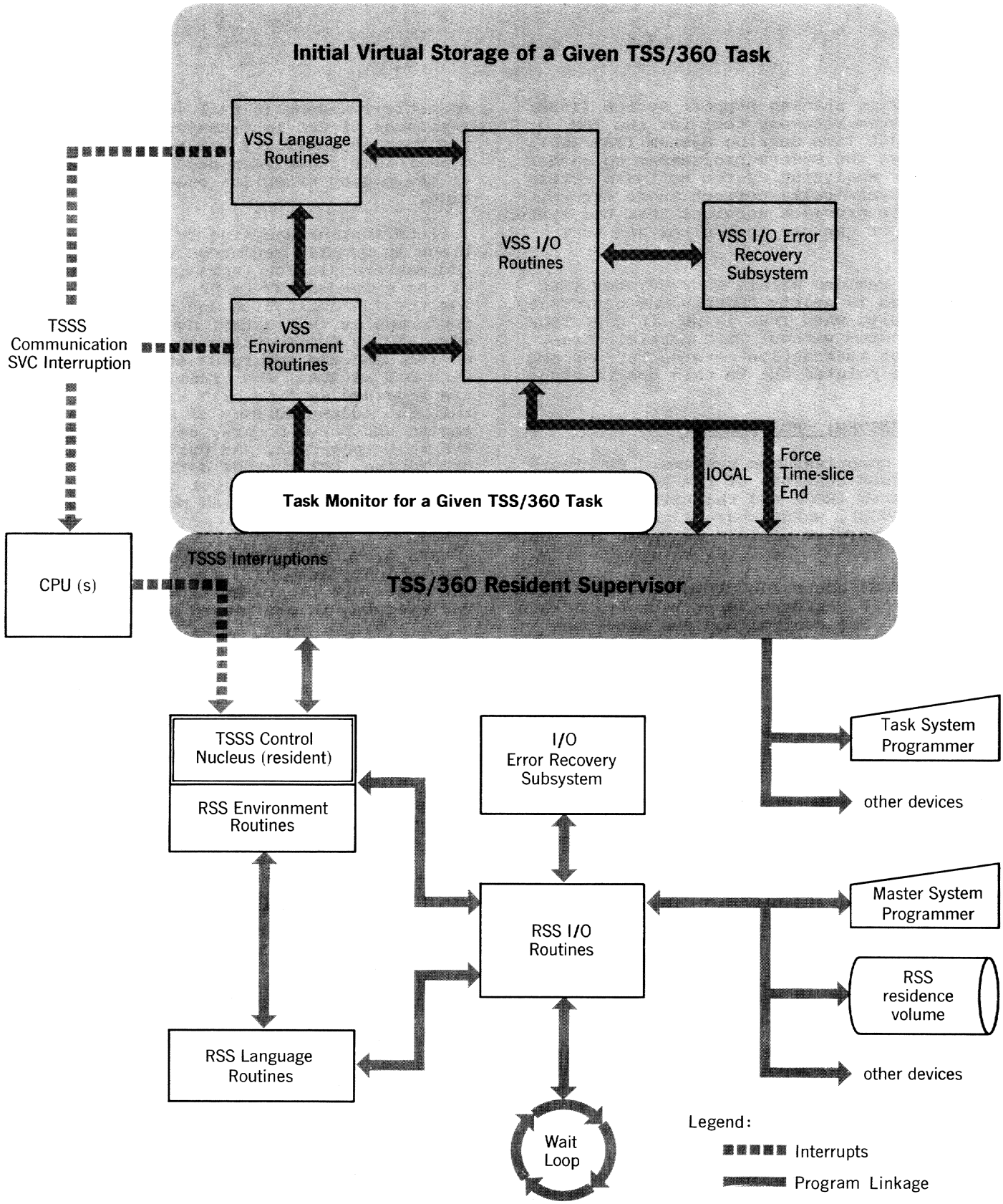


Figure 1. TSSS overview

Command	Function
AT	Designates a dynamic statement and when it is to be executed.
CALL	Initiates the execution of a prestored set of command statements.
COLLECT	Moves data from one area to another.
CONNECT	Causes a TSP to be connected to VSS at a task's terminal. (Valid for an MSP only.)
DEFINE	Enables the SP to define temporary symbols and allocates storage when necessary.
DISCONNECT	Removes the SP capability from the terminal, restores TSS/360 (except for patches), and permanently transfers control to TSS/360.
DISPLAY	Writes on his terminal data requested by an SP.
DUMP	Writes on a specified output device data requested by an SP.
END	Terminates reading of prestored statement sets.
IF	Designates a conditional statement, whereby execution of the statement is dependent on the predetermined condition.
PATCH	Alters the contents of a data field and keeps a record of the patch.
QUALIFY	Establishes implicit RM (real memory), VM (virtual memory), or global qualification for subsequent operands.
REMOVE	Deletes ATs and their associated dynamic statements, or deletes patches.
RUN	Causes control to revert to TSS/360; AT SVCs can then be executed.
SET	Alters the contents of a data field.
STOP	Causes TSS/360 or a specific task to halt and control to be given to the issuing SP.

Figure 2. TSSS commands and their functions

The TSSS commands (sometimes called "keywords" when language processing is under discussion) request the services of TSSS; these are summarized in Figure 2. (The word "command" may mean only the keyword, or it may include the operands. The term "command statement" may imply more than one command, and is used synonymously with "input string.")

The TSSS services can be requested from a terminal or by an AT command. The TSSS AT command designates that the remainder of an input statement is a dynamic statement and indicates when it is to be executed. Many of the TSSS internal functions result from the use of the AT command. Since several sections of this publication are involved in the description of AT processing, an overview is provided here.

PROCESSING THE AT COMMAND.

Figure 3 shows the steps in processing an AT command. The following paragraphs supplement the figure. After being translated into an SVC, the AT is implanted by either RSS or VSS in (1) the real storage of the TSS/360 Supervisor, (2) a task's virtual storage, or (3) shared virtual storage. Implanting overlays an existing instruction, and this original instruction is saved. The instruction that is overlaid by an AT SVC must not be altered by TSS/360. An AT control block (ACB) is built to keep a record of the implanted AT, and it is stored, with the dynamic statement, in an AT table. A given AT SVC may represent a number of stored statements.

The SVC code for a particular AT command is determined by (1) the type of storage,

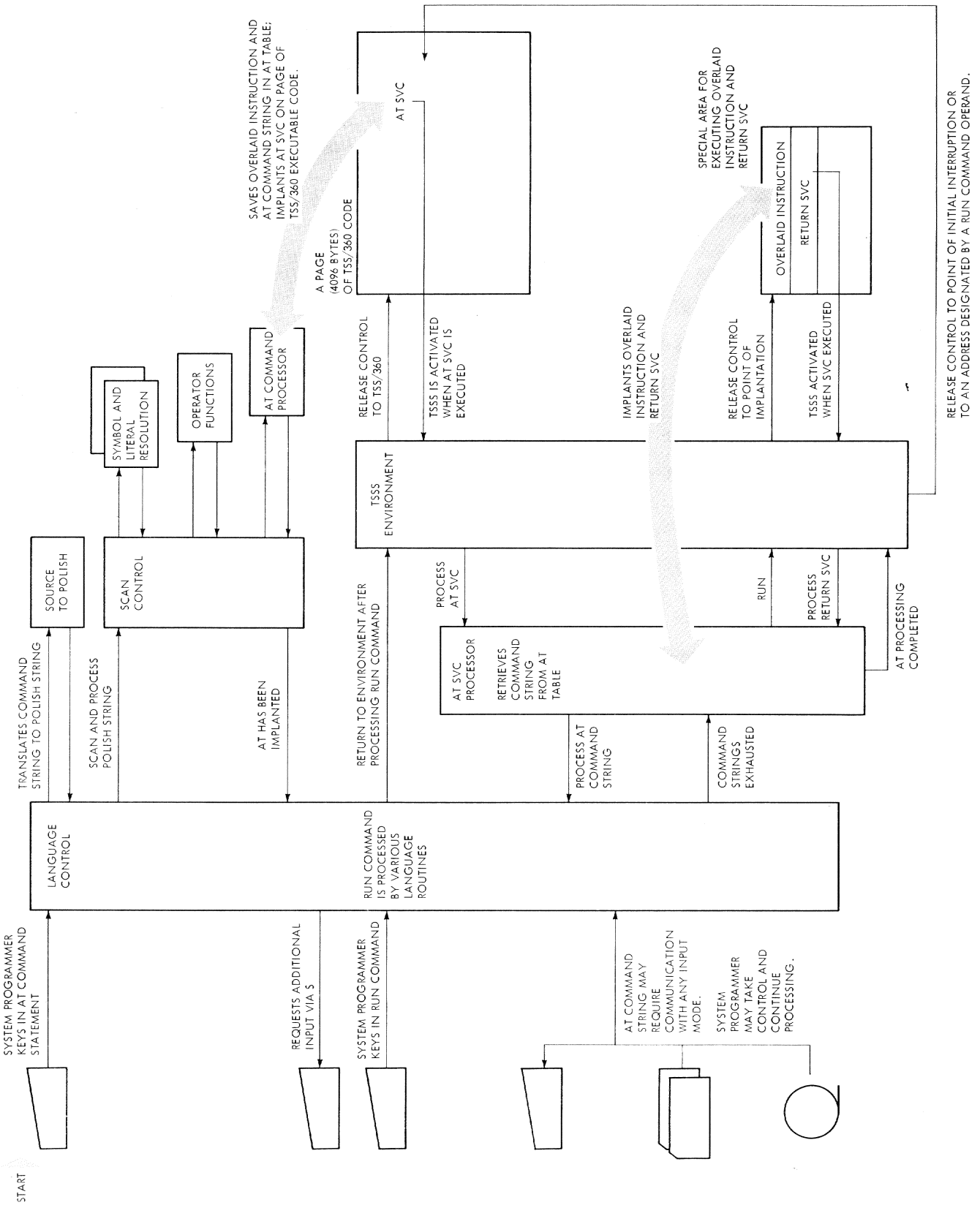


Figure 3. A conceptual approach to AT processing for TSSS

real or virtual, in which it is implanted, and (2) whether RSS or VSS implants it. When TSSS is deactivated, and TSS/360 or the task resumes execution, the implanted AT SVC may be encountered and executed. As a result, either RSS or VSS is activated, depending on the AT SVC code.

The dynamic statement is executed following the execution of the AT SVC, unless the AT SVC was implanted in shared virtual storage by a task other than the currently executing task, and it represents a dynamic statement that is designated by the SP as not applicable to the executing task.

If the AT SVC represents more than one dynamic statement, each dynamic statement is checked for applicability. After all applicable dynamic statements have been executed, TSSS is deactivated, and control is returned to TSS/360 in the following manner.

A return SVC is implanted immediately following the instruction that was overlaid by the original AT SVC in a block of storage. The instruction counter of the PSW that will be loaded to resume TSS/360 execution is set to point to the overlaid instruction. TSSS is deactivated and control is returned to TSS/360, which executes the original instruction and the return SVC. Either RSS or VSS is reactivated and restores the PSW instruction counter to the original next sequential instruction. Processing of the AT command is completed when TSSS is deactivated.

TSSS MACHINE CONFIGURATION

TSSS is usable with the same configurations of CPUs, main storage units, and control units as TSS/360. The I/O devices supported are:

- 1050 Data Communication System -- 1052 Printer Keyboard and 1056 Card Reader only, attached via 2702 Transmission Control Unit
- 2741 Communication Terminal, attached via 2702
- 1052-7 Printer Keyboard
- 1403-2 Printer
- 1403-N1 Printer
- 2401 Magnetic Tape Unit, Models 1, 2, and 3
- 2311 Disk Storage Drive
- 2301 Parallel Drum
- 2314 Direct Access Storage Facility
- 2540 Card Read Punch -- Reader Only
- Tel 33 or etype Model 35[1] KSR teletypewriter (a product of the Teletype Corporation) attached via 2702
- The 2702 Transmission Control

TSSS STRUCTURE

The components (modules) of TSSS make up three logical units:

1. RSS and VSS environment (including the control nucleus).
2. RSS and VSS language processing.
3. RSS and VSS I/O (including the I/O error subsystem).

Figure 1, shown earlier, depicts this structure.

The environment unit handles control and service functions, the language unit processes input from the TSSS user, and the I/O unit performs I/O.

In the language and I/O units most RSS functions and VSS functions are identical, and thus many pairs of modules are nearly duplicates, except for external names; they are sufficiently similar to be described together. Within the environment, however, most of the TSSS modules are unique in function and structure. The introductions to the separate units provide greater detail about the structure of RSS and VSS, as described in those sections.

CONVENTIONS AND GENERAL CONSIDERATIONS

Certain functions are performed in the same way for all three logical units of TSSS. Consequently, they are described here and not discussed separately for each unit.

Message Handling: When an error condition is detected by a TSSS routine, the parameters for requesting output of an error message are placed by the encountering routine in register 0. The originating module is

---

[1]Terminals that are equivalent to those explicitly supported may also function satisfactorily. The customer is responsible for establishing equivalence. IBM assumes no responsibility for the impact that any changes to the IBM-supplied programs or products may have on such terminals.

identified by these parameters, as well as in the message text. Unless the routine that detected the error calls the message routine directly, the presence of an error condition is indicated by passing a return code in register 15 to the calling routine. This return code, as well as the original message parameters, may be passed back through a number of routines. The message parameters are defined in the module descriptions for the RSS and VSS message routines (CEHCM and CZHNM, respectively). The routines that actually call these message routines are Language Control (CEHLC/CZHXC), Scan Control (CEHLS/CZHXS), the Loader (CEHBL) the AT SVC Processor (CEHJA/CZHZA), and, in the case of "intervention required," Error Scan and Recovery (CEHGE/CZHUE).

Note that all TSSS messages are diagnostic; no messages require SP responses. Appendix I lists message numbers and the modules that request output of each message. The messages themselves are shown in Appendix B of the Systems Reference Library publication IBM System/360 Time Sharing System, Time Sharing Support System, GC28-2006.

Register Usage: TSSS employs the following conventions in assigning general purpose registers. TSSS does not use the floating-point registers.

0 - 1	Parameter registers
2 - 12	Work and base registers
13	Save area address register; usually the PSECT register
14	Return register
15	Link register; return code register

Naming and Notation: The following conventions regarding the naming of modules and notation apply throughout TSSS:

- All return codes are given in decimal.
- TSSS module identifiers (IDs) are module names in the format CEHxy for RSS and CZHxy for VSS, where "x" represents a letter in the range A through M for RSS and N through Z for VSS. The value of "y" is any alphabetic character, except that the letters I and O are not used. Note that these conventions do not apply to the TSS/360 module identifiers referred to in this publication.
- Two module IDs separated by a slash (for example, CEHKP/CZHYP) designate the RSS and VSS versions, respectively, of a logical module for which there is a single module description (though not always a single flowchart). Two modules with similar functions, one in RSS and the other in VSS, for which there are separate module descriptions, may be referred to jointly with a comma between the module IDs (for example, CEHCM, CZHNM).
- Entry point and CSECT names are standardized throughout TSSS. The main entry point is the module identifier followed by the letter A; all subsidiary entry points are represented by the module identifier followed by one of the letters B through M. If an entry point is referred to but not specifically called out in this publication, the entry point in question is the primary entry point. A primary entry point has the form: CEHxyA or CZHxyA. All subsidiary entry points are specifically noted.
- Table identifiers immediately following the table names in the form "VSS Status Save Area (CHAEVS)" refer only to the DSECTs. If the CSECT is to be referred to, the identifier is as follows: VSS Status Save Area (CSECT:CHBEVS). For further detail see the introduction to the Appendixes:

Module Attributes: Groups of modules often have identical attributes. Whenever the module descriptions for such a group are preceded by introductory text, the common attributes are listed in that introduction and are not necessarily repeated within each module description.

TSSS Environment modules acquire and release system control through activation and deactivation procedures (including interruption processing) and maintenance of status indicators. At any time after TSS/360 Startup, the MSP may or may not be connected, and one or more TSPs may be connected. Connection and connected are used throughout this publication to denote MSP or TSP capability at a terminal. The implication is that RSS or VSS was successfully invoked and that the disconnect function has not been performed for the connected user. Connection is the result of initial activation, and disconnection is a result of final deactivation.

Activation and deactivation consist of those procedures that make RSS or VSS available and unavailable to the SP. These terms imply entry to and exit from RSS execution mode or VSS execution mode, which may occur repeatedly during a SP's connected period (or terminal session). During this period RSS or VSS may be activated and deactivated any number of times. TSS/360 (for RSS) or the task (for VSS) is made correspondingly inactive or active. The basic purpose of the TSSS environment is controlling these functions. (RSS and VSS communication is also a function of the TSSS environment.)

TSSS LOGON Interface: When an SP activates TSSS by logging on at an idle terminal, a LOGON task is created for that terminal alone. It provides the input parameters for TSSS to determine whether the LOGON request is RSS or VSS. The LOGON task exits to TSSS by remotely executing an SVC 81 instruction, which results in an attempt to activate RSS or VSS by the LOGON RSS/VSS SVC Processor (CEHDL). In VSS, or if activation fails, the TSSS LOGON Processor returns to the LOGON task via LPSW (old SVC PSW) with a return code. Figure 4 illustrates this occurrence. If a RUN command is executed during a given terminal session with RSS, the return is also to the LOGON task with a return code of zero. This situation is called an "intervening run." For all other cases (the return code is not zero), including, for RSS, a Disconnect situation without an intervening run, the LOGON task exits to LOGOFF.

However, if the return code is zero, indicating in VSS a successful LOGON and in RSS an intervening run, the LOGON task puts itself into a wait state by issuing the TSS/360 TWAIT macro instruction. TWAIT also has the advantage of releasing any

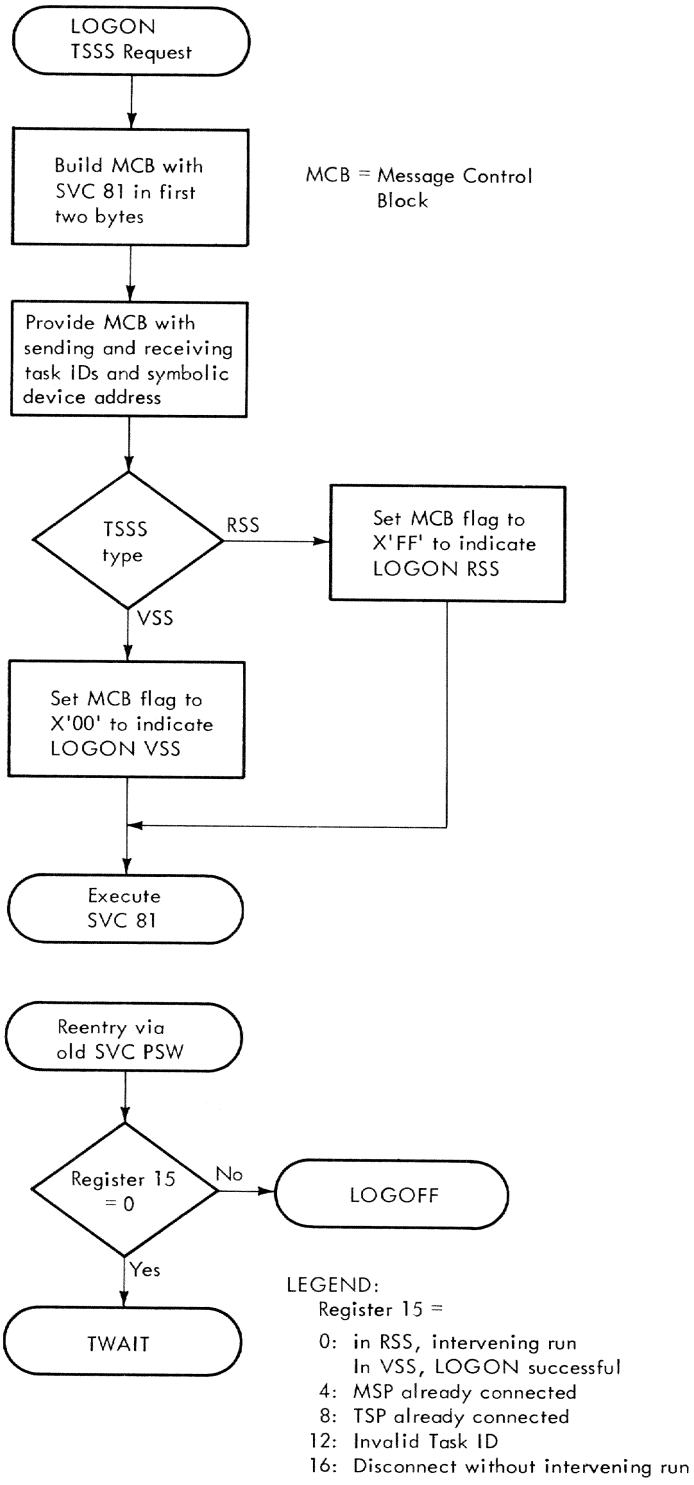


Figure 4. TSS/360 LOGON task interface with TSSS

auxiliary storage on drums by moving pages to the auxiliary disk storage. When, following this action, the SP signals the end of his terminal session through the DISCONNECT command, TSSS causes the TWAIT to be terminated by queuing an external interruption for the LOGON task.

Environment Functions: Aside from interruption handling, which involves several functions, TSSS Environment has four general functions:

- Activation and deactivation of RSS.
- Activation and deactivation of VSS at the control nucleus (real storage) level.
- Completion of VSS activation, and initiation of deactivation, at the task (virtual storage) level.
- Performing services upon request from RSS or VSS.

The description of TSSS Environment that follows is divided into three sections; the groups of modules that perform each of the environment functions are described under separate headings. For convenience the interruption processors, the service functions, and the RSS message routine are included with the first group, "RSS Environment." Certain of their functions are repeated as necessary. The VSS Message Writer is included with the third group, "VSS Environment, Virtual Memory."

RSS ENVIRONMENT

Entry to the RSS Environment is gained via TSS/360 Supervisor-loaded PSWs. After determining that a hardware interruption belongs to RSS, the TSS/360 Resident Supervisor loads a PSW from the SYSRSS field of the TSS/360 System Table (CHASYS). This PSW contains the primary entry point of the appropriate RSS interruption processor. (For a description of SYSRSS, see Appendix H.)

The TSS/360 Resident Supervisor performs a short save into the Prefix Storage Area (PSA) before loading the PSW. Thus, the status saved by RSS into the TSS/RSS Status Save Area (CHAESV) from the PSA is the status of the system at the time of the hardware interruption.

The VSS Environment routines that reside in virtual storage process virtual (simulated) interruptions; the part of the control nucleus having a VSS environment function processes a number of SVC interruptions.

The full range of TSSS interruption handling is effective only after RSS has been activated. Consequently, RSS activation is discussed first, followed by a generalized description of interruption handling.

RSS Activation and Deactivation

When the RSS Environment modules activate RSS, they:

1. Save the TSS/360 status in the TSS/RSS Status Save Area (CHAESV).
2. Force the other CPU into wait state, if in a duplex configuration.
3. Link to the language routines to process input, except for a special case (see "VSS Service Request").

This activation sequence is shown as a part of RSS Environment in Figure 5.

The transient RSS modules are paged as they are referred to. Paging requires space in real storage, which is obtained dynamically by writing one TSS/360 Resident Supervisor page onto the residence device for each transient RSS page that is brought into real storage. Deactivation of RSS reverses this process, although all transient RSS is paged out at one time.

There are three circumstances under which RSS is activated:

- The MSP has initiated activation (initial connection) or has signaled RSS to reactivate and accept input from his terminal.
- An AT (actually an SVC) implanted by RSS or by VSS in real storage has been executed, which requires RSS to be active in order to process a command statement.
- VSS has signaled that it requires service from RSS; the third step in the RSS activation sequence is omitted in some cases, as described under "VSS Service Request."

MSP Intervention: The MSP connects to RSS by pressing a CPU interruption key, which preempts the operator's terminal and temporarily dedicates it to RSS. The manual key external interruption is delivered to and processed by the RSS External Interrupt Processor (CEHAE). Only one MSP can be connected at a given time.

ATs Implanted by RSS: The AT SVCs implanted in real or virtual storage by RSS, cause activation of RSS each time an AT SVC is executed. The execution of an AT

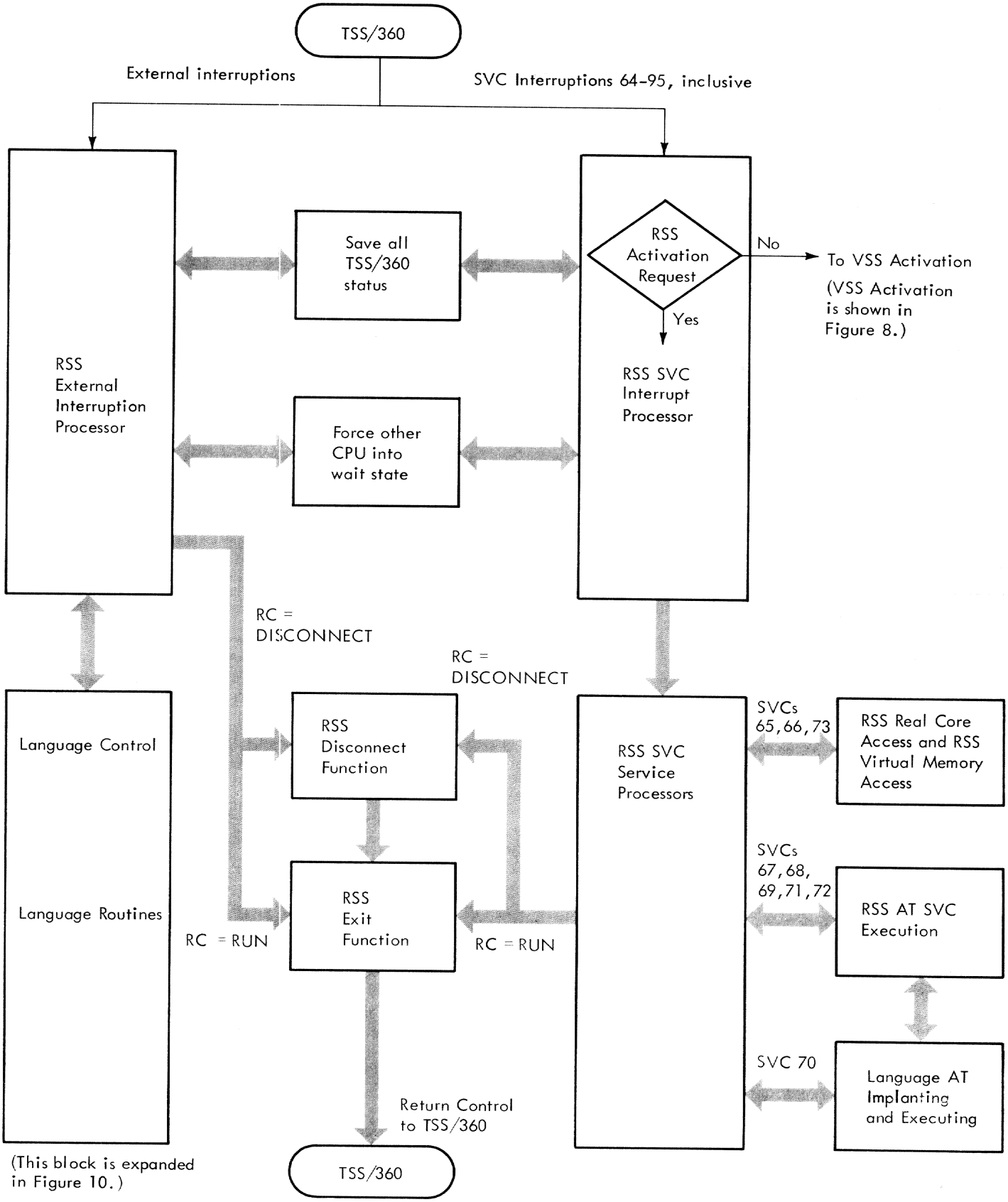


Figure 5. Overview of RSS environment

causes execution of a stored dynamic statement to take place, after which deactivation of RSS normally is automatic. A STOP command in the dynamic statement causes RSS to remain active.

VSS Service Request: VSS cannot access real storage; it must call upon RSS to move data into a buffer (a page at a time) or back to the original location. VSS also calls upon RSS to implant an AT in real storage, using the RSS language area rather than moving the designated page to and from the buffer. In either case, the request is delivered to RSS by execution of an SVC instruction, and RSS is deactivated upon completion of the requested operation.

Deactivation of RSS: The deactivation of RSS occurs automatically, as described above, or is the result of the processing of a RUN or DISCONNECT command by the language routines. The RSS Exit (CEHBE) and RSS Disconnect (CEHBD) routines provide for unloading the transient RSS routines, reloading the saved supervisor pages, restarting the CPU that was halted in a duplex configuration, and transferring control to TSS/360.

Interruption Handling

All SVC interruptions in the range 64-95 and all manual key interruptions are delivered to the TSSS control nucleus for processing. In addition, after RSS has been activated, RSS I/O Control and all program interruptions are directed to the control nucleus.

The TSS/360 Interrupt Stacker recognizes the above interruptions as belonging to TSSS and delivers them by loading a new PSW. As a result of this procedure, there actually are two sets of new PSWs. (A reference to a "new PSW" in a module description must be considered in context; the reference may be to either a hardware-loaded or a supervisor-loaded new PSW.)

INTERRUPTION PROCESSORS: The following discussion introduces the interruption processors of the TSSS control nucleus.

I/O Interruptions: Two types of I/O interruptions may be received after RSS has been flagged active; they are directed to the RSS I/O Interrupt Processor (CEHAD):

1. With each initiation of an I/O operation, RSS waits for its completion and processes the resulting synchronous (expected) I/O interruption before proceeding.

2. An asynchronous interruption that is caused by an Attention from the MSP terminal, with RSS executing, is recorded but processed later by a non-environment routine. (Any other asynchronous I/O interruption belongs to TSS/360 and is queued by the TSS/360 Interrupt Stacker.)

The status of a interruption -- expected or not expected -- is recorded in the TSSS Active Device Table (SADT). SADT is a part of the TSS/360 System Table (CHASYS) for RSS, and a part of TSS/VSS Status Save Area (CHAEVS) for VSS.

SVC Interruptions: All TSSS SVC interruptions are initially directed to the SVC Interrupt Processor (CEHAS), which recognizes two types: those that activate RSS and those that do not.

The execution of AT SVCs in real storage and the service requests from VSS to RSS (via SVCs) are described under "RSS Activation and Deactivation."

Program Interruptions: Program interruptions occurring when RSS is active are delivered to the RSS Program Interruption Processor (CEHAP). Code 17 program interruptions indicate that an addressed page (with dynamic address translation active) is not in main storage; the RSS Loader is called to read the excepted page. Code 5 program interruptions, if they occur while the RSS Real Core Access routine is in control, indicate a recoverable addressing exception. To correct this error, this information is transmitted to the routine that called RSS Real Core Access. All other program interruptions are treated as major error conditions (see Chart 06).

Message Procedures

All error conditions for which a message is sent to the system programmer's terminal are handled identically within this group of modules. An error message word is created in register 0 (as described under "RSS Message Writer"), and return code 4 is passed back to the calling routine in register 15.

Attributes

The interruption handling and RSS activation and deactivation modules described individually below are nonrecursive, serially reusable, and they execute in supervisor state. The attributes listed within each module description are: (1) residency and (2) the state (active or inactive) of dynamic address translation (DAT).

RSS External Interrupt Processor (CEHAE)

Chart 01

The RSS External Interrupt Processor directs the activation, and, subsequently, the deactivation of RSS for the MSP. The interruptions which cause direct or indirect entry to this module are as follows:

1. Main operater's terminal: An external interruption from a CPU manual interruption key.

2. Remote terminal: An asynchronous I/O interruption (attention) from the MSP.

ATTRIBUTES: Resident. DAT inactive upon initial entry, until control is returned from the call to the Status Save routine (CEHCH).

ENTRIES: In normal processing, entry is at CEHAEA via the TSS/360 Supervisor-loaded interruption key PSW from the system table (CHASYS). If an external interruption is pending when RSS Disconnect (CEHBD) or RSS Exit (CEHBE) receives control, this module is entered at CEHAEB to handle the external interruption.

MODULES CALLED: Under normal conditions:

Module Name and ID	Reason	Chart ID
RSS Inter-CPU Communications (CEHCC)	Force other CPU into halt and transfer	04
RSS Status Save (CEHCH)	Save TSS/360 status	02
RSS Language Control (CEHLC)	Invite input and direct the processing of it.	38

Under an error Condition:

RSS Message Writer (CEHCM)	Indicate restart in progress or storage print failure.	15
RSS Unloader (CEHBU)	Unload RSS in an activation retry attempt.	17
RSS Loader (CEHBL)	Load all transient RSS.	05

EXITS: Under normal conditions exit from this routine is either to RSS Exit (CEHBE) or to RSS Disconnect (CEHBD), depending on the return code from Language Control. Language Control returns control to the External Interrupt Processor upon encountering and processing a RUN or DISCONNECT command, indicating which it was with a return code of zero or four, respectively.

OPERATION: The External Interrupt Processor checks lock byte CEHEAK to determine whether RSS is active in the other CPU. If it is, a loop is entered to await the Halt and Transfer operation.

If CEHAEK is off, this routine checks whether RSS is active. If RSS is not active, it is loaded and activated. The system table is set to show RSS active, the SYSERR lock byte CEAIS15 is saved, and the system is checked for duplex operation. If more than one CPU is in operation, the interruption processor calls the inter-CPU communications module (CEHCC) to initiate a Halt and Transfer operation in the other CPU by means of a Write Direct.

After this is done, or if only one CPU is in operation, the RSS Status Save module (CEHCH) is invoked to save CPU status.

When Status Save returns control the cause of the interruption is checked. An area in the TSS/RSS Status Save Area (CHAESV) records terminal information if a remote MSP is connected to the system. If this area contains only zeros, the terminal information must be filled in for the main operator terminal. If a remote MSP is connected, the input device table entry is filled in with the remote terminal information.

CEHLC is then invoked to direct the processing of input. Normal termination occurs when the MSP requests either a run (CEHBE) or a disconnect (CEHBD).

If RSS is active at the time of the interruption, the origin of the interruption must be determined. If it comes from a CPU other than the one executing RSS, it is ignored (the external old PSW is loaded). If the interruption comes from the CPU executing RSS, and was received while this CPU was in the error wait state, it indicates that a Halt and Transfer operation attempted upon the other CPU was unsuccessful, and control is passed to CEHCH to save TSS status. When this module returns control, CEHBL is invoked to resume normal processing.

If the error wait flag is not on, and an external interruption is pending, TSSS is restarted. If no external interruption is pending, a flag is set to show this one as pending, and processing is resumed by loading the external old PSW.

RSS Status Save Routine (CEHCH)
Charts 02,03

This module saves TSS/360 status information in a predefined RSS save area (DSECT CHAESV) when RSS is activated, and loads any pages needed to activate RSS. The status information will be restored at RSS exit to reestablish TSS/360.

ATTRIBUTES: This module is resident, and executes with DAT active.

ENTRIES: This module is entered at CEHCHA by the RSS SVC Interrupt Processor (CEHAS) and the RSS External Interrupt Processor (CEHAE) upon activation of TSS.

MODULES CALLED: TSS Write Shared Pages (CEAMW) is called to free shared pages for RSS use.

EXITS: Exit is to the calling routine.

OPERATION: Some of the TSS status information that must be saved by this routine is contained within the Prefix Storage Area (PSA) of the CPU on which RSS is activated. This information includes:

1. Old PSWs (PSAEOP, PSASOP, PSAPOP, PSAMOP, PSAIOP)
2. New PSWs (PSAENP, PSASNP, PSAPNP, PSAMNP, PSAINP)
3. Current TSI Pointer (PSATPT)
4. Channel Address Word (PSACAW)
5. Channel Status Word (PSACSW)
6. Interrupt Codes (PSAEIC, PSASIC, PSAPIC, PSAMIC, PSAIIC)
7. Registers 15 through 4 as they were saved by the Interrupt Stacker at the time of interruption (PSAISS)

This routine saves additional machine data which was present at the time of interrupt:

1. Current PSW (that is, the old PSW stored at the time of interrupt).
2. Remaining registers. (These registers are not used by any routine between the time of interruption and being saved by CEHCH.)
3. Floating point registers.
4. Control registers.

This module initializes the Loader SIORCB (an SIORCB set aside for Loader usage) with V-type address constants which it copies from the primary SIORCB (used by the I/O area for normal processing).

After the save operations, this module builds the page tables for segment two (transient portion of RSS to be paged in) and segment three (symbol dictionary for use by RSS).

If the transient portion of RSS is greater than the pageable portion of the Resident Supervisor, then RSS, when activated, will determine the number of pages still needed and dynamically build page tables for them in segment two and segment three. This is done in three steps. First the storage block table is used to get pages from the unassigned chain. If more pages are needed, this module calls TSS Write Shared Pages (CEAMW) in order to free shared pages for RSS use. If still more pages are required, the task status indexes are searched to find available private pages for RSS use. If still more pages are needed, a major system error is declared.

RSS Inter-CPU Communications (CEHCC)

Chart 04

The function of this routine is to cause all but the primary CPU (that is, the CPU which executed an RSS activation SVC or received a manual key external interrupt) to be forced into a halt and transfer, thereby giving RSS complete system control.

ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: Inter-CPU has the following three entry points;

CEHCCA: Initial entry. This entry point is used by SYSERR.

CEHCCB: The RSS External Interrupt Processor (CEHAE) and the RSS SVC Interrupt Processor use this entry point.

CEHCCC: This is the transfer point for the Halt and Transfer write direct option. The subject CPU executes a wait loop in this portion of the module.

MODULES CALLED: This module calls the TSS Inter-CPU Communication routine (CEAIC) to issue a write direct against the subject CPU.

EXITS: If the routine was entered at CEHCCA or CEHCCB to initiate a Halt and Transfer operation, and if the operation was successful, a return code of zero is placed in register 15, and control is

passed to the calling routine. If the Halt and Transfer operation was unsuccessful, the error wait state is entered.

If the routine was entered at CEHCCC as the transfer address, return at the end of RSS activation is to the calling routine.

OPERATION: When this module is entered at CEHCCA by SYSERR, it first checks the lock byte CEHAEK. If this is on, a loop is entered to await the halt and transfer operation.

If CEHAEK is off, the module tests and sets the SYSERR lock byte (CEAIS15), and it is here (CEHCCB) that the RSS modules CEHAE and CEHAS enter the module. If the SYSERR lock byte is on, indicating that the other CPU is in halt and transfer, control is returned to the calling routine.

If the SYSERR lock is off, the number of active CPUs in the system is loaded. The ID of the object CPU is obtained for reference. If the first CPU ID referred to is that of the CPU currently executing RSS, then the next one is referred to. If the CPU being inspected is not the object CPU, a flag is checked to determine whether or not it is available. If it is, its ID is loaded, the transfer point within this module (CEHCCC) is passed, and control is given to the TSS Inter-CPU Communications Module (CEAIC) for the Write Direct. When CEAIC returns control, a test is made to determine whether the Halt and Transfer was successful. If it was, and the RSS lock byte (RSSLCK) is on, a code of zero is placed in register 15, and control is returned to the calling routine. If it was not, a 1-second loop is entered to await the completion of the operation. If the Halt and Transfer is not successfully completed within one second, the Error Wait PSW is loaded, and the system is placed in the error wait state. Otherwise, the RSS lock (RSSLCK) is set off, flags are set to indicate that the write direct was received and that the subject CPU is halted, and a code of zero is returned to the calling routine.

If the CPU is not the object CPU and it is not available, the RSS lock is tested. If it is on, the CPU is considered to be halted. If it is off, again the 1-second wait loop is entered, and processing continues as above.

A CPU that is being halted passes control to CEHCC at CEHCCC. This module then saves the CPU's general and control registers, turns on the RSS lock byte (RSSLCK), and enters a loop by testing the SYSERR lock and branching back to that test until the lock goes off. This means that RSS has been deactivated and the subject CPU may restore its registers and resume normal processing.

RSS Loader (CEHBL)

Chart 05

Under normal conditions this routine reads in one non-resident RSS page or TSS/360 Symbol Dictionary page and writes a TSS/360 Supervisor pageable page. The Loader is normally called as the result of a paging exception, at which time it reads in the excepted page. If called as the result of a restart attempt, this module pages in all the transient RSS modules, writing out as many supervisor pages as necessary.

ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: This module is entered at CEHBLA by the Program Interrupt Processor (CEHAP) if a paging exception occurs. The following parameters are passed to it:

Reg. 0	A page table entry pointer
Reg. 1	External work area table entry pointer
Reg. 2	An external page table entry pointer
Reg. 3	TSS external page table entry pointer

During a restart attempt, this module is called by the RSS External Interrupt Processor (CEHAE). In this case no parameters are passed.

MODULES CALLED: The loader calls:

Module Name and ID	Reason	Chart ID
RSS External Page Location Address Translator (CEHBT)	To locate the physical location of the input address.	06
RSS I/O Control (CEHEA)	To write out a page of TSS/360 or read in a page of RSS.	09
RSS Message Writer (CEHCM)	To indicate a load failure.	15

Input to I/O Control is the Loader SIORCB with the designated input or output device, the external device page number, and the storage location specified.

EXITS: Exit is to the calling routine.

OPERATION: The paging process involves writing one page of the TSS/360 Supervisor onto an external device, and then reading a

transient RSS page into the vacated space. For normal dynamic paging the Loader gets the addresses of the load function tables from the RSS Program Interupt Processor as input parameters. These entry pointers indicate the next available slots in the load tables into which the Loader may record the page transaction.

If the Loader is called during restart it gets the addresses of the load function tables from the RSS Segment Table. The address of the RSS Segment Table is placed in control register 0 by the RSS Status Save routine (CEHCH).

TSS Startup is responsible for building all but one of the tables used by the Loader (that is, the RSS Segment Table, the TSS Pageable Table, the RSS External Page Table, a Segment Three Page Table and External Page Table for the supervisor symbol dictionary, and the External Work Area Table). Figure 6 shows the content and function of each table. The Loader builds and maintains the TSS External Page Table (CHAEXT). (See Appendix B for a detailed description of the Loader Tables.)

Load Function Execution: After initialization, the Loader calls I/O Control to write out the supervisor page. The current entry in the External Work Area Table provides the output device identification and target address, while the current entry in the TSS Pageable Table designates the current storage location of the supervisor page. Following the Write, this module builds an entry for the supervisor page in the TSS External Page Table.

This module then links to I/O Control, which reads the requested page of transient RSS. The current entry in the RSS External Page Table contains the input device identification and the page's current storage location on an external device, while the current entry in the TSS Pageable Table is

Table Name	Contents	Function
RSS Segment Table	One entry for each segment number, totaling five.	To provide entries for segments two, three, and four point to the TSS Page Table, Segment Three Page Table, and the External Work Area Table, respectively.
TSS Pageable Table	A list of addresses of the TSS/360 Supervisor pages that can be written out. An "in storage" indicator.	To determine which pages can be written out, and into which locations in real storage RSS pages can be read.
RSS External Page Table	A list of auxiliary storage addresses of the direct access device location for each RSS page (where it resides).	To locate each page of transient RSS and to cause it to be read. (The Loader determines the origin of this table by manipulating the pointer to the TSS Pageable Table.)
Segment Three Page Table	A list of addresses of the symbol dictionary pages.	To provide target addresses for a page of the symbol dictionary.
Segment Three External Page Table (Symbol Dictionary)	A list of addresses of the external device location of each page of the symbol dictionary.	To allow access to the symbol dictionary pages to be read.
External Work Area Table	A list of addresses of available direct access storage.	To provide target addresses for each TSS/360 page list in the TSS Pageable Table.
TSS External Page Table (created by the Loader) (CHAEXT)	The external and the internal (main storage) addresses of each page that is written out, the name of the direct access device on which it resides, and the page's protection keys.	To enable the Unloader to replace each page of TSS/360 when RSS is deactivated.

Figure 6. The Loader tables

the target storage address (the address of the supervisor page just written out).

Following a successful read operation, the Loader flags the corresponding entry in the TSS Pageable Table as "in storage" (bit 12 is set to zero), to indicate that the supervisor page has been replaced by an RSS page. If the Loader was called as the result of an attempted restart, it decrements its loop control count, updates the table pointers to point to the next entries in each table, and loops through the same logic until all the transient RSS pages have been read into main storage. Otherwise the paging operation is finished.

(In the restart paging procedures, after the non-resident RSS load is completed, this module pages in the supervisor symbol dictionary. The loop count is governed by a Startup-supplied number. The loop logic is the same as the RSS load, except that the symbol dictionary PT and the symbol dictionary XPT replace the TSS Pageable Table and the RSS External Page Table. The Loader continues using the External Work Area Table and the TSS External Page Table, serially.)

After the paging operation, this module saves the pointers to the last-used entries in the External Work Area Table and the TSS External Page Table in the TSS/RSS Status Save Area (ESVEWPGE and ESVTEPE, respectively). The pointers are available to the Program Interrupt Processor (CEHAP) and the RSS Unloader (CEHBU).

The table entries represented by these pointers indicate (1) the next available address on the direct access device used to save the supervisor pages, and (2) the next available slot in which a record of the write-out of a supervisor page can be created. The Program Interrupt Processor calculates the other two pointers needed as input to this module if a paging exception occurs.

If an error indication follows an attempt to write or read a page, this module sets the error parameters, restores the table pointers to the origins of the tables, and calls RSS Message Writer to indicate a paging failure. When control returns from RSS Message Writer, the Loader exits to the TSS/360 System Error Processor (CEAIS).

External Page Location Address Translator (CEHBT/CZHRT)

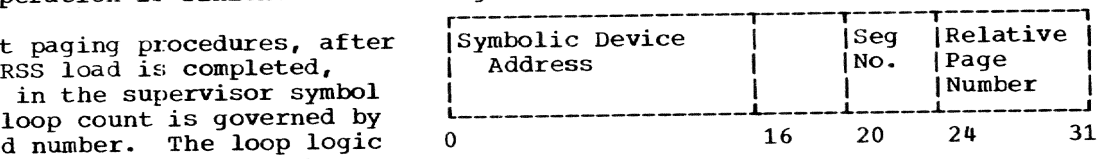
Chart 06

This module translates the two-byte relative page number of an External Page Address Word received as input into a physical I/O data location.

ATTRIBUTES: This routine is resident and executes with DAT active.

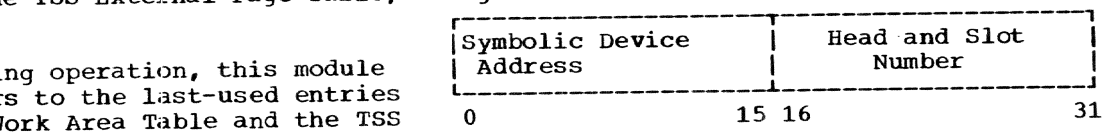
ENTRIES: This module is called by the Loader (CEHBL), the Unloader (CEHBU), RSS Real Core Access (CEHCA), the Operator Functions routine (CEHLA/CZHXA), and RSS VM Access (CEHCB). It expects an External Page Address Word as an input parameter (register 0) if the device specified is a disk.

Register 0:



If the device specified is a drum (2301), the input required is the symbolic device address and a "head and slot" number.

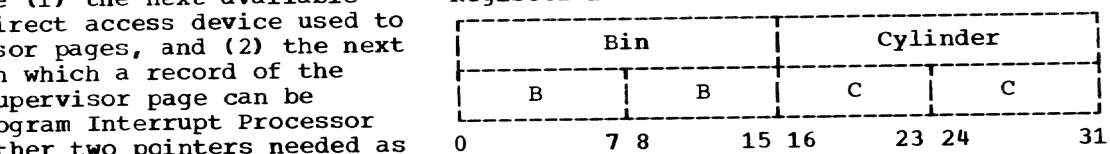
Register 0:



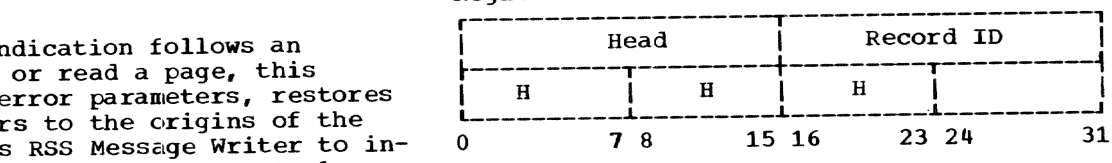
MODULES CALLED: None.

EXITS: Exit is to the calling routine. If the operation is successful, this module passes the following return parameters:

Register 1:



Register 2:



If the device specified is a drum (2301), the output is the head and record ID only, as shown in register 2.

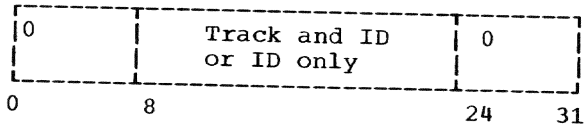
OPERATION: If RSS is being loaded and unloaded, this routine uses the symbolic device address in the input External Page Address Word as a search argument to find the corresponding entry in the resident portion of the TSSS Device Allocation Table (SSDAT). If RSS is not being loaded or unloaded, this routine uses the same search

argument to find the corresponding entry in the non-resident portion of the TSSS Device Allocation Table (SSDAT). It copies the device type for later use.

This module checks the input relative page number to ensure that it does not exceed the address range for the specified device:

Device Name	Device Type	Range
Drum Storage	2301	0-899
Disk Storage Drive	2311	0-1623
Direct Access Storage Facility	2314	0-6495

If the relative page number is acceptable, and if the device is a 2311 or 2314, this module uses the device type to determine which subroutine it will use to find the cylinder number. The selected subroutine finds the cylinder number by dividing the input relative page number by the number of pages that can be placed on a cylinder. The remainder of this division is used as a search argument to locate a track and a record ID for the designated address. This search takes place on a separate table for each device. These tables are maintained by this module in the following format (four bytes per entry):



Output from the division and search procedures is a physical data address of the form CHHR, corresponding to the input symbolic device address. On completion of the requested operation, this module returns control to its calling routine (see "Exits").

For a 2301, input becomes output without the search procedures described above and without additional translation. The input is first checked for validity.

If this module recognizes one of the following error conditions, it executes the error return procedures, passing the listed return code. Such an error is recognized as a TSSS system error.

Error Condition	Hexadecimal Error Numbers
Invalid paging device type	cc0902
Invalid 2301 symbolic address	cc0A02
Invalid 2311 symbolic address	cc0B02
Invalid 2314 symbolic address	cc0C02
Device not in the SSDAT	cc0202

where cc = the last two characters of the module identifier.

The error numbers are passed in register 0; the first digit is the class code and the second digit is the message number.

RSS Program Interrupt Processor (CEHAP)

Chart 07

The Program Interrupt Processor receives and processes program interruptions that occur while RSS is active.

ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: TSS/360 delivers the interruption to this module at CEHAPA when RSS is active, via the Supervisor-loaded RSS program PSW (SYSRS1 in TSS System Table).

When the RSS SVC Interrupt Processor (CEHAS) or the RSS SVC Service Processors module (CEHDR) encounters a TSSS system logic error, it exits to the Program Interrupt Processor at CEHAPB. CEHAPB is the entry point that defines the TSSS System Logic Error Processor subroutine. The input parameter for this subroutine is a two-character module identifier in bytes 2 and 3 of Register 1.

MODULES CALLED: If entry to this module resulted from a paging exception program interrupt, this module calls the RSS Loader (CEHBL) to read in the excepted page. It passes the following parameters:

Reg. 0 A page table entry pointer

Reg. 1 An external work page table entry pointer

Reg. 2 An external page table pointer

Reg. 3 A TSS external page table entry pointer

The page table and external page table may be for either segment 2 or segment 3 addresses.

If this module is entered as the result of an internal system logic error it calls RSS Message Writer (CEHCM, Chart 15).

EXITS: (1) If entry to this module resulted from a paging exception, and the requested load is successful, this module exits by loading the old program PSW (location X'28'). (2) If entry to this module resulted from an addressing check while RSS Real Core Access (CEHCA) was executing, this module exits to the routine that called Real Core Access. (3) Under any other condition this module recognizes a major error and exits to RSS Exit.

OPERATION: This module distinguishes between interruptions caused by either a paging exception (interruption code 17) or an addressing exception (code 5), and all other interruptions.

Code 17: "Paging exception" means that an addressed page is not in main storage. This program interruption is delivered to RSS whenever RSS is active. This module links to the RSS Loader, which reads the excepted page into main storage. This capability allows RSS to execute with only those RSS pages that are necessary for execution in main storage.

The Program Interrupt Processor checks for a valid address by testing the relocation address register (control register 2). A valid address in this case is a segment two or three address. If the address is invalid this module exits to the TSS/360 System Error Processor.

This module uses the address of the excepted page as an index to search the appropriate page table (that is, Segment 2 Page Table or Segment 3 Page Table). The origin of the page table is an entry in the RSS Segment Table. When it finds the entry for the excepted page, this module puts the address in register zero. Using the page table entry's displacement from the page table origin, this module locates the corresponding entry in the corresponding external page table (called here the RSS External Page Table and the Symbol Dictionary External Page Table). The page table entry indicates the location into which the excepted page is to be loaded, while the external page table entry indicates the current location of the excepted page on an external device.

If this is the first time in a given terminal session that a paging exception has occurred, this module initializes the load tables pointers from the RSS Segment Table (address in control register 0). For any time after the first, this module updates the pointers to the External Work Area Table (Register 1) and the TSS External Page Table (Register 3) by referring to the RSS Status Save Area (ESVEWPGE and ESVTEPE, respectively). These pointers indicate the next available entries in each table, as maintained and saved by the RSS Loader, at the end of the load function. The two entries indicate the location on a direct access device into which a TSS/360 Supervisor page may be written (ESVEWPGE) and a location in which a record of this Supervisor page's location on the external device may be kept (ESVTEPE).

This module calls the RSS Loader to load the page that caused the exception. When the Loader returns control, this module restores the input registers and loads the old program PSW.

Code 5: If an addressing exception occurs this module tests the "real core access in control" flag in the status save area (ESVRCAP). If the flag is on, this module determines which routine called RSS Real Core Access (CEHCA) and returns to this calling routine, passing back error parameters to indicate the addressing error. This permits cancellation of the operation that called upon Real Core Access, with notification to the System Programmer and without a major system error.

If the flag is not on, or if any condition other than the ones already discussed exists, this module branches to its internal subroutine, the TSSS System Logic Error Processor, which is entered at CEHAPB.

System Logic Error: This subroutine calls the RSS Message Writer module (CEHCM) to print out a system logic error message. Input to the message routine is a message control word.

On return from the RSS Message Writer, this subroutine modifies the exit PSW to allow external and I/O interruptions and tests if the program is in problem state. If it is not, this subroutine sets the wait bit on before exiting to CEHBE. It then exits to RSS Exit (CEHBE) to unload RSS and restore control to TSS/360. This internal subroutine may also be called by the RSS SVC Service Processors module (CEHDR).

RSS I/O Interrupt Processor (CEHAD)

Chart 03

This routine processes all RSS I/O interruptions. It (1) determines whether the interruption was synchronous or asynchronous, (2) sets the appropriate indicator, and (3) stores the attendant program and channel status information.

ATTRIBUTES: This routine is resident and executes with DAT active.

ENTRIES: This routine is entered from the TSS/360 Interrupt Stacker at CEHADA via the TSS/360 Supervisor-loaded RSS I/O PSW (SYSRS3 in TSS/360 System Table) when an RSS I/O interruption occurs. It expects no input parameters.

MODULES CALLED: None.

EXITS: This routine exits by loading the I/O old PSW (location X'38').

OPERATION: This module checks both entries in the TSSS Active Device Table (SADT), which is a portion of the TSS/360 System Table (CHASYS) to determine if the I/O interruption is synchronous (expected). If it is synchronous, this module sets the "synchronous interruption received" flag (SYSIRM) in the corresponding SADT entry. It stores the I/O old PSW and the CSW in the SADT, masks I/O interruptions, and exits via the I/O old PSW to the point of interruption.

If the interruption is from the Attention key, this routine sets the "attention received" flag (SYSARM) in the corresponding entry of the SADT, stores the PSW and the CSW, and exits. If the interruption is an asynchronous interruption other than an Attention, it is ignored.

RSS Channel Interrupt Processor (CEHAC)

Chart 09

This module initiates RSS activation as a result of an MSP asynchronous interruption (attention). (See Chart 20.)

ATTRIBUTES: This module is resident and executes with DAT inactive.

ENTRIES: This module is entered at CEHACA from the TSS/360 Channel Interrupt Processor or from the LOGON RSS SVC Processor (CEHDL). (The TSS/360 routine loads the RSS Channel Interrupt PSW, SYSRS5, from the TSS/360 System Table.)

MODULES CALLED: None.

EXITS: This module exits to TSS/360 by loading the new external PSW, which simulates a manual key external interruption. TSS/360 then delivers control to the RSS External Interrupt Processor (CEHAE).

OPERATION: This routine disables all but machine check interruptions. It sets the "MSP attention received" flag in the System Table (CHASYS) to indicate an asynchronous interruption.

This routine modifies the external old PSW (location X'18') to contain the address of the TSS/360 Queue Scanner. As a result, when the RSS Exit routine (CEHBE) loads the external old PSW at the end of RSS operations to return control to TSS/360, the Queue Scanner gets control instead of the TSS/360 routine that was in control at the time of the interruption.

This module then sets the manual key interruption code in the new external PSW and loads it. The new external PSW contains the address of the TSS/360 Recovery Nucleus to which all external interruptions are normally delivered. On recognizing the manual key interruption code, the TSS/360 Recovery Nucleus passes control to the RSS External Interrupt Processor (CEHAE) by loading a field (SYSRS4) from the TSS/360 System Table (CHASYS). The RSS External Interrupt Processor continues the RSS activation procedures.

RSS SVC Interrupt Processor (CEHAS)

Chart 10

This module is called when TSS/360 encounters a TSSS SVC interruption (SVC codes 65-95, inclusive). This module may initiate and direct RSS activation. SVCs 65-79 force RSS activation, halting TSS/360; SVCs 80-95 do not. This module verifies the validity of the SVC issued, determines which module supplies the requested service and links to it.

ATTRIBUTES: This module is resident and executes with DAT inactive.

ENTRIES: This routine is entered from TSS/360 at CEHASA via the RSS SVC PSW (SYSRS2 in the TSS System Table). The SVC interruption code is stored in the Prefix Storage Area (PSA).

MODULES CALLED: If RSS is to be activated, this routine calls the following modules, expecting a return from each.

Module Name and ID	Requested Function	Chart ID
RSS Inter-CPU Communications (CEHCC)	Force a wait state on the subject CPU	04
RSS Status Save (CEHCH)	Save all TSS Status	02,03

If a nonprivileged user issues a privileged SVC, this module calls the Queue VSS Interrupt routine (CEHCQ) and exits to the TSS Queue Scanner (CEAJQ).

EXITS: Depending on the SVC code it encounters, and whether the SVC has been correctly issued, this module exits to one of the following modules:

Module Name and ID	SVC Code	Chart ID
RSS SVC Service Processors (CEHDR)	65-73	11
RSS/VSS LOGON Processor (CEHDL)	81	19
VM AT Execution SVC Processor (CEHDA)	80,84,85	24
VSS Command SVC Processor (CEHDV)	83	23
VSS Exit (CEHDE)	82	26
TSS Queue Scanner (CEAJQ)	86	

The routines given control by SVCs 80,83, 84, and 85 return to the queue scanner.

OPERATION: Because TSSS SVCs can be executed in either CPU in a duplex configuration, TSSS must guard against an attempt to execute an SVC in both CPUs at one time. Accordingly, this module sets an activation indicator before honoring requests to activate RSS (SVCs 65-79). Upon entry, the module tests the indicator and, if it is on, enters a loop to await the Halt and Transfer operation.

(Note: The CPU executing the wait loop will momentarily be forced into wait state by the RSS activation procedures executed by the other CPU. When the waiting CPU is restarted, the activation indicator will be off, and this routine's wait loop will then be broken.)

If RSS is not being activated, and the SVC interruption code exceeds 79, this module calls one of the resident SVC processors listed under "Exits".

If the RSS activation flag (ESVTASP) is on, a loop is entered until it goes off. It is then turned on, and the validity checks are made. If the supervisor call old PSW or the service routine priority flag indicates that the user is privileged, all validity checks are bypassed. If the user has issued an SVC that is available to a nonprivileged user, this module branches to the appropriate processing routines to activate RSS. If he has issued any other SVC, the Queue VSS Interrupt routine and the TSS Queue Scanner receive control.

SVCs 65, 66, and 70 require that a Load Real Address be performed on the contents of register 1 as they were at the time of the SVC. If this operation fails, or if SVCs 87-95 (unassigned) have been issued, this module exits to the Error Processor (CEHAP).

RSS SVC Service Processors (CEHDR)

Chart 11

Each of the routines that make up this module performs a service for RSS or VSS. Each recognizes one of the following SVC codes as a request for service.

SVC 65	Get real storage for VSS
SVC 66	Put real storage for VSS
SVC 70	Submit a VSS-supplied command string to RSS Language Control (CEHLC) for processing.
SVC 67	Resume processing after AT execution in RM
SVC 68	Resume processing after RSS AT execution in VM
SVC 69	Execute AT in real storage for RSS or VSS
SVC 71	Execute AT in virtual storage for RSS
SVC 72	Execute AT in shared virtual storage for RSS
SVC 73	Determine whether input virtual storage page is shared

ATTRIBUTES: This module is nonresident and executes with DAT active.

ENTRIES: This module is called by the RSS SVC Interrupt Processor (CEHAS) at entry point CEHDRA. A branching table uses the input SVC code as an index to the specific service routine. The routines require the following input parameters:

Input SVC code	Input Parameters for SVC Service Routine	
SVC 65	Reg. 0 --	real address of the requested page
	Reg. 1 --	real address of the virtual paging buffer
	Reg. 2 --	a Qualify Table entry
SVC 66	Reg. 0 --	real address of the destination page
	Reg. 1 --	real address of the virtual paging buffer
	Reg. 2 --	a Qualify Table entry

	Reg. 3 --	in bytes two and three, the piece of data overlaid by the SVC
SVC 70	Reg. 1 --	real address of the first byte of the command string
SVC 73	Reg. 1 --	virtual address of requested page
	Reg. 2 --	X'CC' in byte 0
SVCs 67,68, 69,71,72		No parameters required.

MODULES CALLED: Each routine calls specific modules to perform its requested function.

Module Name and ID	Requested Function	Responsible SVC code(s)
RSS Real Core Access (CEHCA)	Perform the requested get/put operation on a page of real storage.	65,66
RSS AT SVC processor (CEHJA)	Process AT-implanted command string or return SVC.	67,68,69 71,72
RSS Language Control (CEHLC)	Process VSS-supplied command string.	70
RSS Virtual Storage Access (CEHCB)	Determine if input page is shared	73

EXITS: Under normal conditions exit is either to RSS Exit (CEHBE) or RSS Disconnect (CEHBD), as requested. If this routine encounters an invalid SVC code, it exits to the RSS Program Interrupt Processor (CEHAP) at CEHAPB. CEHAPB is the entry point of the TSSS System Logic Error Processor subroutine.

OPERATION: These routines operate on the parameters saved in the TSS/RSS Status Save Area by the RSS SVC Interrupt Processor in order to perform a service for either RSS or VSS. The SVC processors do not perform the actual work for the requested operation. Instead they supply the correct parameters to the operating routines -- Real Core Access, RSS VM Access, and the AT SVC Processor.

The routine that processes SVC 70 -- submitting a VSS-supplied command string to RSS Language Control (CEHLC) for processing -- moves 260 bytes into the RSS Language buffer, simulating terminal input. This includes a 4-byte field containing the actual length of the command string and 256 bytes of input. This routine turns on the "input in storage" flag in the input Device Table (CHALCR) and the AT execution flag (ESVATXM) in the RSS Status Save Area (CHAESV). This module then calls Language Control to process the input string, insuring that Language Control returns to it when SVC 70 processing is completed, rather than inviting additional input from the terminal. By using SVC 70, VSS may implant and remove specified AT SVCs in real storage for a TSP.

The routine that processes SVC 73--determining if an input VM page is shared--passes the 'CC' flag in the high order byte of register 2 when it calls RSS VM Access (CEHCB). This flag causes RSS VM Access to supply all the information that it normally does for a get request, without actually performing the get function. Upon return, this module substitutes the registers 1 and 15 supplied by RSS VM Access for the task's registers 1 and 15, as saved in the RSS Status Save Area (CHAESV).

After completing its requested function, each SVC processor exits as requested. If one of these processors encounters an error, it passes error return parameters back to VSS by altering the stored register contents.

RSS Real Core Access (CEHCA)

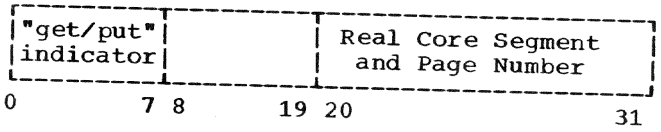
Chart 12

This routine makes available, upon request, a designated page of TSS/360 real storage. The page may be an RSS page, a saved page that was overlaid by RSS, or a Supervisor page currently in main storage.

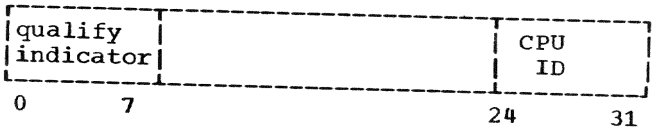
ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: This module is called by the RSS SVC Service Processors (CEHDR), DUMP/DISPLAY Commands Processor (CEHKD), the AT Command Processor (CEHKA), and the SET Command Processor (CEHKS). Registers 1 and 2 contain the input, as follows:

Register 1:



Register 2:



where qualify indicator may be either RM unqualified or RM qualified (see Appendix E) real storage, and CPU ID may be either 1, 2, 3, or 4.

Note: This input parameter is only relevant if the input address in register 1 is zero, thus referring to a Prefixed Storage Area in any CPU.

MODULES CALLED: In order to read or write a page of storage, Real Core Access calls the External Page Location Address Translator (CEHBT) and RSS I/O Control (CEHEA).

EXITS: Exit is to the calling routine.

OPERATION: This routine determines if the request is for an RSS or TSS/360 page.

If the request is for an RSS page and if the input address is a valid RSS address, this routine checks the input get/put flag. If a get function is requested, this routine moves the contents of the designated page into the paging buffer; if put, it moves the contents of the paging buffer into the designated location.

If the request is not for an RSS page and if the address is a valid real storage address, this routine sets pointers to the top of the TSS External Page Table (CHAEXT) and searches for an address matching the input address. If no match is found, the requested page is in storage. This routine moves either the contents of the page into the paging buffer (get), or the contents of the paging buffer into the input address (put).

If the input segment and page numbers are zero, this module tests register 2 for RM qualified or unqualified.

If RM is qualified and a valid CPU ID has been specified (one that exists in the current configuration) this module uses either the primary or alternate prefix of the CPU specified to address the requested page. The primary and alternate prefixes of each CPU are listed in the CPU Status Table (CHACST), which resides within the PSA of each CPU. If neither prefix is available, an error is recognized.

If a match is found, the TSS/360 page has been saved on an external device. This routine calls the External Page Location Address Translator routine to find the physical location of the data. On return, this module initializes an SIORCB and calls RSS I/O Control (CEHEA). RSS I/O Control performs the external get/put function as requested.

Note: The get is not in lieu of the paging performed by the Loader. An RSS page not in main storage will cause a paging exception when the move is first attempted.

If the operation is completed successfully, this module returns to the calling routine. For the use of the RSS Program Interrupt Processor (CEHAP), this routine turns the "real core access active" flag (ESVRCAP in Status Save Area) on at entry and off at exit.

If the input address is not valid or if this routine receives an error return from either of the called modules, it executes the error return procedures.

RSS VM Access (CEHCB)

Charts 13,14

This module enables an MSP to refer to a virtual storage page. The functions of this module are:

1. To develop the real location of the input virtual storage address (VMA).
2. To indicate to the calling routine whether the VMA is in shared storage.
3. To load the specified page into an RSS buffer or move a page from the buffer back to its prior location, as requested.

ATTRIBUTES: This module is nonresident and executes with DAT active.

ENTRIES: This routine is called by the RSS SVC Service Processors (CEHDR), the AT SVC Processor (CEHJA), the AT Command Processor (CEHKA), the DUMP/DISPLAY Commands Processor (CEHKD), and the SET Command Processor (CEHKS) at entry point CEHCBA. This module expects the following input parameters:

Reg. 1 -- a virtual storage address

Reg. 2 -- Byte 0 indicates the operation to be performed: X'AA'=get, X'BB'=put, X'CC'=shared page determination, X'DD'=bring in XTSI.

Byte 1 is unused.

Bytes 2 and 3 contains task ID compatible with the TSS task ID field in the TSI (TSITID).

If Byte 0 = X'CC', bytes 2 and 3 will contain a task ID of zero.

MODULES CALLED: If the task ID is not zero, this routine calls the Find TSI routine (CEHCF). To read or write external pages of virtual storage, this routine calls the External Page Location Address Translator (CEHBT) and I/O Control (CEHEA). Real Core Access (CEHCA) is also called if the VM page resides in real storage.

EXITS: Exit is to the calling routine.

OPERATION: This routine uses the input task ID to find the TSI by calling the Find TSI routine. If the TID field of register 2 is zeros, this routine extracts the currently active TSI pointer from the Prefix Storage Area (PSA). A flag in the TSI indicates if the XTSI is in real storage.

If the XTSI is not in real storage, this routine gets the external address of the first XTSI page from the TSI and causes it to be read into real storage. It then checks for valid input segment and page numbers. If they are valid, this routine determines if the Segment Table (ST) and the Auxiliary Segment Table (AST) are contained within the page of the XTSI currently in real storage. If either is not, the missing table is read into real storage.

This routine then computes the appropriate ST entry from the ST origin and the input segment number. The ST entry contains a pointer to the origin of the appropriate Page Table (PT), and a count of the number of pages in the PT.

This routine determines if the required segment is in shared storage by testing a flag in the AST entry. If the segment is not shared, this routine uses the PT pointer in the ST to reference the appropriate PT. If it is shared this routine uses the Shared Page Table (SPT) number in the AST to search the Resident Shared Page Index (RSPI) for a match. The matching entry in the RSPI contains a pointer to the origin of the PT for the shared segment.

The page tables may be in the XTSI page if there was sufficient space on the XTSI page after the ST and AST were built. If the page tables are not in the XTSI, a Page Table Page (PTP) exists for the input segment. This routine determines the external location of the PTP from the AST entry and causes the PTP to be read into real storage. Each PTP is composed of a number of entries, each comprising a header, the PT, and the External Page Table (XPT). This routine searches the PTP headers for an entry matching the input segment number. (From this entry it later determines the appropriate XPT entry to have it read into real storage.)

If the input virtual address in register 1 is valid, this routine tests register 2 to determine if the input condition is 'CC'. If it is, this module bypasses the get/put function. If the entry condition is not 'CC' this routine tests a flag in the PT to determine if the corresponding page is in real storage. If the requested page is already in real storage, this routine calls Real Core Access (CEHCA) to perform the internal get/put function as requested. If it is not, this module calculates the External Page Table (XPT) entry from the PT entry. Using this entry as an input parameter, this module calls the External Page Location Address Translator to determine the physical page location. When control returns, this module links to I/O Control to complete the get/put operation with a physical read or write.

On successful completion of the requested operation, this module returns to its calling routine. If the segment was shared, this routine returns an indication to this effect and the SPT number.

ERRORS: The following error conditions result in a return to the calling routine with a return code of four in register 15 and message parameters in register 0:

1. Error return from any called routine.
2. The segment number of the input VMA is greater than the Segment Table length (invalid VMA).
3. The page number in the input VMA is greater than the Page Table length (invalid VMA).
4. The page is unavailable (in transit, TWAIT, or unprocessed by loader). A page of zeros is returned to the calling routine.

RSS Message Writer Routine (CEHCM)

Chart 15

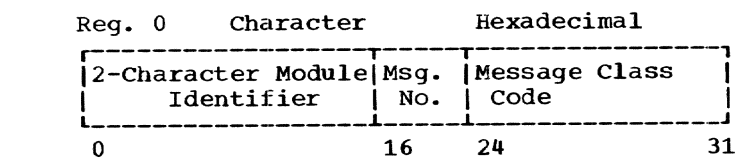
This routine is called when an error condition is detected within RSS for which a diagnostic message is to be written to the system programmer. The messages are divided into four classes:

Class 0	I/O-generated messages
Class 1	User (SP) errors
Class 2	TSSS system errors
Class 3	Loader/Unloader failures

The messages normally are written on the SP's terminal.

ATTRIBUTES: This module, including Class 3 message texts, is resident. It executes with DAT active.

ENTRIES: This module is called by those routines responsible for posting error conditions (generally Language Control and Scan Control), at CEHCMA. Register 0 contains the identification code of the module that encountered the error, a message number, and class code for the requested message.



CEHCMB is used by Error Scan and Recovery (CEHGE/CZHUE) when it is impossible to write a message because of a total I/O failure. This module then exits to the TSS/360 System Error Processor.

MODULES CALLED: This module calls I/O Control (CEHEA) to write the message at the terminal.

EXITS: Exit is to the calling routine.

OPERATION: RSS Message Writer uses the message class code and the number as an index to get the desired message text. Class 0, 1, or 2 message texts reside in a transient RSS module. This routine places the module identifier in the message test, builds an SIORCB, and links to RSS I/O Control to write the message. One or more lines of output may be printed, but this module calls RSS I/O Control only once for each message, unless the error occurred in AT mode.

The module identification code consists of the last two alphabetic characters in the Module ID. For example, RSS Language Control's module ID is CEHLC. The identification code which it places in register 0, as input to this module, is LC.

Class 0 messages (I/O-generated messages) may require additional lines of output containing further information defining the error. The I/O error recovery routines flag the corresponding indications (message control flags) in the SIORCB to show the information that should be printed and move the correct information into the SIORCB. This routine checks the message control flags in the SIORCB and sets up the corresponding fields as additional message lines. The additional information may include any or all of the following:

Symbolic device address

Seek address

Physical path address

CSW

PSW

Sense data

Alternate path

Channel logout area

An I/O error which occurs during the message processing results in an attempt to write the original message to the Operator's terminal (retry), except in the case of SP Attention received. This message, accompanying a return code of four results in a return to the calling routine, which will recall Message Writer to print the Attention received message.

The symbolic device address of the operator's terminal is the first entry of the TSSS Device Allocation Table (SSDAT). A failure in the Write to the operator's terminal causes an exit to the TSS/360 System Error Processor.

If TSSS is in AT mode, this routine calls RSS I/O Control to write the original command string from the command input buffer onto the System Programmer's terminal.

RSS Message Writer is only a means of informing the system programmer of unusual conditions that may occur during RSS execution. The SP, after interpreting the diagnostic information, must make the necessary corrections; RSS requires no response.

Message Writer output format:

CEHidxxx 24-Character Message Text

where id represents the module that encountered the error and xxx represents the class code and message number.

See Appendix I for a complete list of messages.

RSS Disconnect (CEHBD)

Chart 16

This module disconnects an MSP prior to exiting from RSS; this module is called upon all occurrences of a DISCONNECT command.

ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: This routine is called by the RSS External Interrupt Processor (CEHAE) and the RSS SVC Service Processors (CEHDR) if return code 4 (DISCONNECT) has been passed from Language Control. The entry point is CEHBDA.

MODULES CALLED: If the MSP is remote this routine calls the RSS Unloader (CEHBU), TSS/360 Supervisor Core Allocation (CEAL01), and the TSS/360 Queue GQE on TSI routine (CEAAF).

EXITS: Exit is to the main RSS Exit entry point -- CEHBEA (see Chart 18). If an external interruption is pending, this module exits to RSS External Interrupt Processor at CEHAEB.

OPERATION: If an external interruption has been queued, (ESVQXT in the RSS Status Save Area is on) this module turns off ESVQXT and links to the RSS External Interrupt Processor (CEHAE) at CEHAEB. If no interruption is pending, this module determines if the DISCONNECT command came from a remote MSP by testing the "manual key interruption" field in the TSS/RSS Status Save Area (ESVMSPM). If this field is not void, the MSP activated the system with the manual interruption key at the console. This routine clears the "manual key activation" field in the RSS Status Save Area (ESVMSPM) and exits to RSS Exit (CEHBE) at its primary entry point (CEHBEA).

If the DISCONNECT command came from a remote MSP, this module clears the "MSP connected" field (ESVMSPCN) in the RSS Status Save Area and then clears the TSSS Active Device Table (SADT). This module links to the RSS Unloader (CEHBU) to page out RSS and restore TSS/360. When control returns, if no error occurs, this module sets the TSS External Page Table (CHAEXT) to all hexadecimal F's and clears the pointers to the page tables.

This module then tests the "intervening run" flag (ESVNVRUN) in the RSS Status Save Area. If the flag is set to X'FF' a RUN command has not been issued during the current terminal session. This module indicates the condition of "no intervening run" by setting register 15 to 16. It then sets ESVNVRUN to X'00', restores the TSI pointer in the Device Group Table, and exits to RSS Exit (CEHBEB).

If ESVNVRUN is set to X'0F', one or more RUN commands have been issued during the current terminal session. This module links to Supervisor Core Allocation (CEAL01) to request a storage block large enough to build a General Queue Entry (GQE) and a Message Control Block (MCB). On return this module builds a GQE in the external interruption format (byte 59 = X'FF'), sets up an MCB with MCBCD1 equal to X'5F', and calls the TSS/360 routine Queue GQE on TSI (CEAAF) to queue an interruption that will terminate the LOGON1 module's TWAIT condition. (For an explanation of the TWAIT condition in the LOGON1 module, see "VSS Environment Part 1.") This module sets the exit PSW equal to the address of the Queue Scanner, so that the interruption will be issued. This module then sets ESVNVRUN to X'00', restores the TSI pointer in the Device Group Table, and exits to RSS Exit (CEHBEB).

RSS Unloader (CEHBU)

Chart 17

This module restores the TSS/360 Supervisor to its state preceding the activation of RSS. To do this it causes the saved TSS/360 pages to be paged in, after writing out any changed, transient RSS pages onto an external device. These operations alternate on a page-by-page basis.

ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: The Unloader is called by RSS Exit and RSS Disconnect (CEHBD) at CEHBUA. In a restart procedure the Unloader is called by the RSS External Interrupt Processor (CEHAE).

MODULES CALLED: This module calls RSS I/O Control (CEHEA) to read TSS/360 or write RSS pages. Input to I/O Control consists of an SIORCB with the external and the real storage addresses. The External Page Location Address Translator (CEHBT) is also called to calculate the physical location of the data to be read or written, before I/O Control is called.

EXITS: Exit is to the calling routine.

OPERATION: The Unload process is a loop, governed by the number of entries in the TSS Pageable Table and the Symbol Page Table defined under "RSS Loader (CEHBL)" (see Charts 05 and 18). This routine sets the table pointers to the first entries in the TSS Pageable Table and the TSS External Page Table (CHAEXT). Page count is the length of the TSS Pageable Table. This routine checks an entry in the Page Table, and if the "in storage" bit is zero, the page of storage is RSS and must be exchanged for a TSS/360 Supervisor page. If the "in storage" bit is one, the pointer is updated to point to the next entry.

If the page of storage defined by the table entry is RSS, this module further checks to determine if restart was in progress (ESVPROI in the RSS Status Save Area is on). If restart was in progress the

write operation is bypassed. If restart was not in progress, this module checks the RSS page to determine if it was changed during RSS execution by testing the "changed" bit in the protection keys for the page. If it was changed this module gets the corresponding entry in the RSS External Page Table and requests a one-page write by linking to RSS I/O Control. If the block of storage was not changed, the write operation is bypassed.

This module now requests a page read by RSS I/O Control. The TSS External Page Table contains both the storage address and the external location of the overlaid TSS Supervisor page. This module flags the corresponding TSS Pageable Table entry as "not in storage," and restores the protection keys for the TSS page from the TSS External Page Table.

The Unload loop control checks are in the following order:

1. Page counter equal zero?

2. Symbol table unload in process?

The second is determined using a switch (ESVSTUN): when "on" it indicates that the symbol unload is in process; when "off" that transient RSS is being unloaded. This module turns ESVACT on after it completes the unloading for transient RSS and resets the table entry pointers to the beginnings of the tables. (The switch is turned off prior to exit from the Unloader.)

When all the RSS nonresident pages have been unloaded, and the corresponding TSS Supervisor pages restored, the Unloader unloads the Supervisor Symbol Dictionary, using the same loop logic, with the exception that previous references to the TSS Pageable Table are now to the Symbol Page Table.

Before returning to the calling routine, this module determines whether the RSS Loader used pages from the Core Block Table. If it did, those pages must be placed on the unavailable chain of the Core Block Table.

RSS Exit (CEHBE)

Chart 18

This module restores TSS/360 status and returns control to TSS/360 from RSS, unloading RSS if necessary.

ATTRIBUTES: This module is resident and executes with DAT active.

ENTRIES: This module is called by the RSS External Interrupt Processor (CEHAE) and the RSS SVC Service Processor (CEHDR) at CEHBEA if either of them receives a RUN indication from Language Control. This module is also called by RSS Disconnect (CEHBD) at CEHBEA, for a local MSP or, if the disconnecting MSP is remote, Disconnect calls this module at CEHBEB.

MODULES CALLED: If unloading RSS is necessary, this module calls the RSS Unloader (CEHBU). RSS Exit also links to the TSS/360 Inter-CPU Communications module (CEAIC) to restart the other CPU, if the configuration is duplex.

EXITS: This module exits to TSS/360 by loading the current PSW (the old PSW that was stored when RSS was activated) from the TSS/RSS Status Save Area (ESVCPSW), or by calling the Queue Scanner (CEHJQ). If an external interruption is pending, this module exits to the RSS External Interrupt Processor (CEHAE) at CEHAEB.

OPERATION: If RSS Exit is entered at CEHBEA it checks for an external interruption pending and branches to the External Interrupt Processor if it finds one. Otherwise, it calls the RSS Unloader (CEHBU) to page out the transient RSS modules and restore the TSS/360 Supervisor. (If entered at CEHBEB from RSS Disconnect, this function has already been performed.) On return of control from the unloader, if the unload operation was unsuccessful, a major system error is declared. If unloading was successful, this module sets the TSS External Page Table (CHBEXT) to all hexadecimal F's, and clears the pointers to the page tables.

If the RUN command was preceeded by a LOGON, the intervening run flag (ESVNVRUN) is set to indicate this fact. It is then determined whether the system was running in duplex mode or not. If it was, then information from the Inter-CPU communications module (CEAIC) save area is used to restore the second CPU's status and restore it to normal operation by turning the RSS lock byte off. The status of the main CPU is then also restored. If the exit address is that of the Queue Scanner, this module turns off the wait bit, resets the control bits, and branches to the Queue Scanner. If the exit address is not that of the Queue Scanner, this module determines if the exit PSW is in the problem state. If it is in the problem state, and an external interruption is pending, exit is to the TSSS External Interrupt Processor (CEHAE). Otherwise, exit is to the Queue Scanner. If the PSW indicates the supervisor state and an external interruption is pending, again exit is to CEHAE. If no external interruption is pending, however, this module exits via a load PSW on the exit PSW to the point of interruption in TSS.

VSS ENVIRONMENT, REAL STORAGE

A specific portion of the TSSS control nucleus is responsible for activating and deactivating VSS for a specified task. The modules involved are thus logically a part of VSS environment but are physically a part of RSS (resident in real storage); they bear RSS-type module identifiers (CEHxy).

Three procedures can initiate activation of VSS for the purpose of connecting a TSP:

1. The VSS command of the TSS/360 command language is used at the SYSIN terminal of a logged-on task.

2. The CONNECT command of RSS is used by the connected MSP, specifying a task to which the TSP shall be connected. A TSP cannot be connected to an idle terminal by the CONNECT command.

"Connected" implies TSP capability at a terminal, whether or not the TSP is using VSS; the connected state is ended only by disconnecting (normally via the DISCONNECT command). Figure 7 shows the real storage activation procedures.

The VSS command indicates that the connected TSP has preempted the task's SYSIN terminal, until he issues a RUN command. When a RUN command is executed, the terminal again serves as the task's SYSIN terminal. For an Attention to be delivered to TSS/360, however, it must be followed immediately by end-of-block (this is sometimes referred to as a "void command"). Otherwise, an Attention reactivates VSS and dedicates the terminal to it instead of to the task.

The CONNECT command indicates that the connected TSP has control and the task's SYSIN terminal is locked out until a RUN command is issued. When a RUN command is executed, the task's SYSIN terminal is again available to the task. VSS may be reactivated at any time with an Attention from the TSP terminal.

VSS initial activation (via any of the ways described above) involves the following steps; subsequent activation involves all but the first step:

1. Finding the Task Status Index (TSI) for the specified task by linking to the Find TSI routine (CEHCF).

2. Building and saving a duplicate TSI so that the task may be restored when VSS is deactivated.

3. Initializing and setting fields in the original TSI to indicate to TSS/360 that VSS has been activated for the specified task.

4. Building and queueing a General Queue Entry (GQE) as a "VSS activate interrupt" for the task.

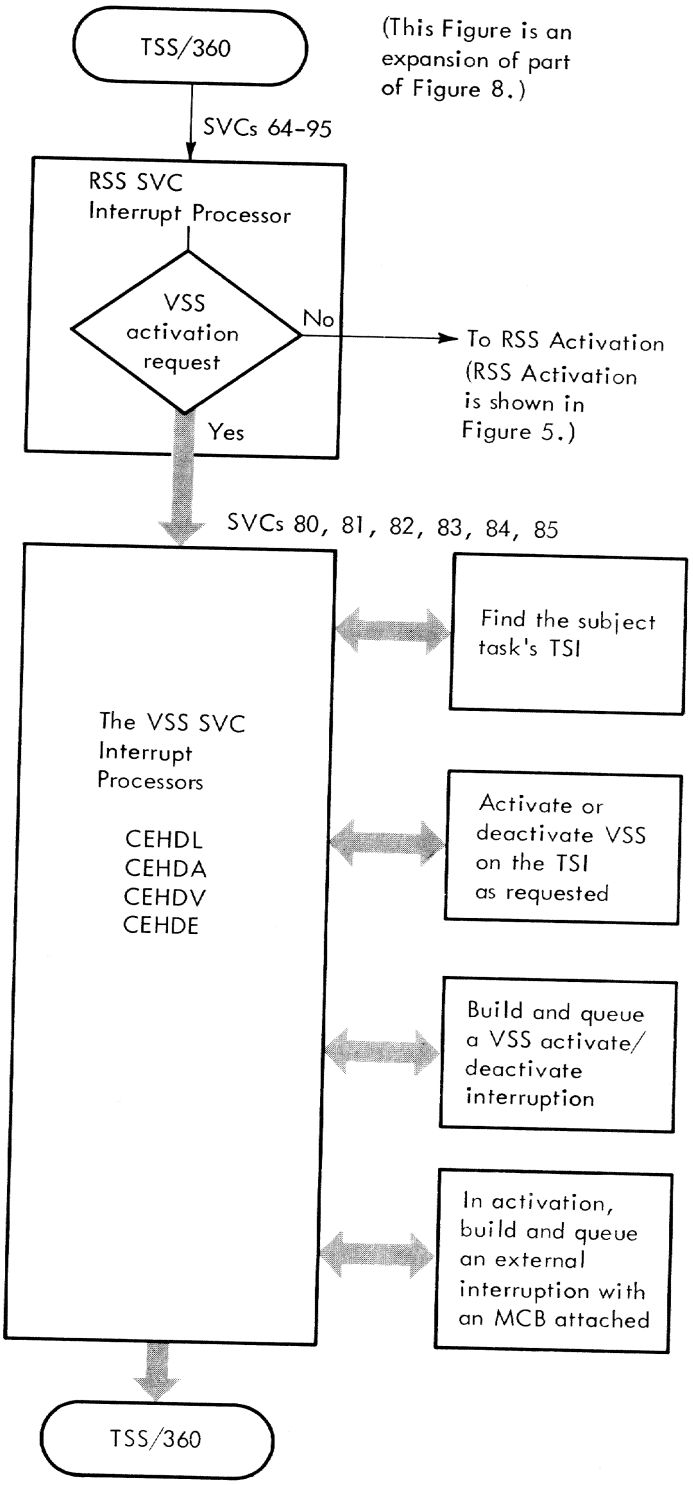


Figure 7. Overview of VSS environment and real storage

5. Building and queuing an external interruption for the task with an MCB containing TSP terminal information.

6. Returning control to the TSS/360 Queue Scanner.

This activation sequence is initiated by an SVC resulting from LOGON VSS, by an SVC resulting from TSS/360 processing of the VSS command, or by the RSS CONNECT command processor. In the latter case, in which the MSP initiates activation, a RUN or DISCONNECT command must be issued by the MSP before control passes to the Queue Scanner.

Two interruptions are required in the VSS activation sequence. The first, the "activation" interruption, contains a code which indicates the activation circumstances (for example, code 0 for LOGON) to the VSS routines at the virtual storage level. The second, the "external" interruption, indicates that there is an MCB in the Interrupt Storage Area (CHAISA) which contains the TSP terminal information. TSS/360 moves the MCB into the task's ISA when it executes the external interrupt, but, in so doing, it overlays task status information. For this reason, two interruptions are queued. The receipt of the activation interruption prior to the MCB-bearing external interruption causes the VSS Activation Interrupt Processor (CZHNV) to store the entire ISA in a special PSECT, CZHPSR, thus preserving the task status. The VSS activation sequence is shown in Figure 8.

Subsequent activation of VSS (the TSP remains connected but has relinquished control with the RUN command) is initiated either via execution of an AT SVC or with an Attention from the TSP. In the latter case, unlike AT SVCs, the activation upon receipt of the asynchronous interruption has the same results as initial activation -- the TSP has control at his terminal until he issues a RUN or DISCONNECT command.

An AT SVC may have been implanted in shared virtual storage or in private virtual storage; either type activates VSS. In shared virtual storage, execution of an AT SVC by any task activates the VSS of that task, whether or not the task in control is the "parent" task (the task whose VSS implanted the AT SVC) or is a task with a TSP connected.

If a shared virtual storage AT has global qualification, execution of the SVC by each task causes execution of the associated dynamic statement after VSS is activated within the task. If an AT does not have global qualification but the SVC is in shared virtual storage, the VSS of each non-parent task is deactivated as soon as qualification is determined.

Activation of VSS, by whatever means and at whatever time, involves saving a copy of the TSI and thus saving any pending task interruptions. VSS interruptions are queued to the working TSI and processed as long as VSS is active; upon VSS deactivation, pending task interruptions are again pointed to by the original TSI.

Each of the SVC processors and the TSP Asynchronous Interrupt Processor, as VSS activation routines, exit to the TSS/360 Queue Scanner and do not regain control. Deactivation is signaled from the VSS environment area via SVC 82, the processing of which completes deactivation of VSS and restoration of the task's status.

ATTRIBUTES: Each of the routines in this part of the environment area is resident, non-recursive, serially reusable, and executes with Dynamic Address Translation (DAT) inactive.

LOGON RSS/VSS SVC Processor (CEHDL)

Chart 19

This module activates RSS for an MSP or, within a specified task, VSS for a TSP as the result of an SVC 81 being executed by the TSS/360 LOGON1 module.

ENTRIES: The RSS SVC Interrupt Processor (CEHAS) directs control to this module at entry point CEHDLA.

This module expects as input parameters a Message Control Block (MCB) containing:

1. The Task IDs (TID) of the sending and receiving tasks

2. The Symbolic Device Address (SDA)

3. An RSS/VSS indicator.

These parameters are provided by the TSS/360 LOGON task created for the two-terminal case. (See "TSSS Environment, TSSS LOGON Interface.")

MODULES CALLED: This module calls the following modules for each of the listed entry conditions:

Entry condition	Modules called
1. RSS LOGON	None
2. VSS LOGON	Find TSI routine (CEHCF) RSS Interrupt Switching (CEHCS) Queue VSS Interrupt (CEHCQ)

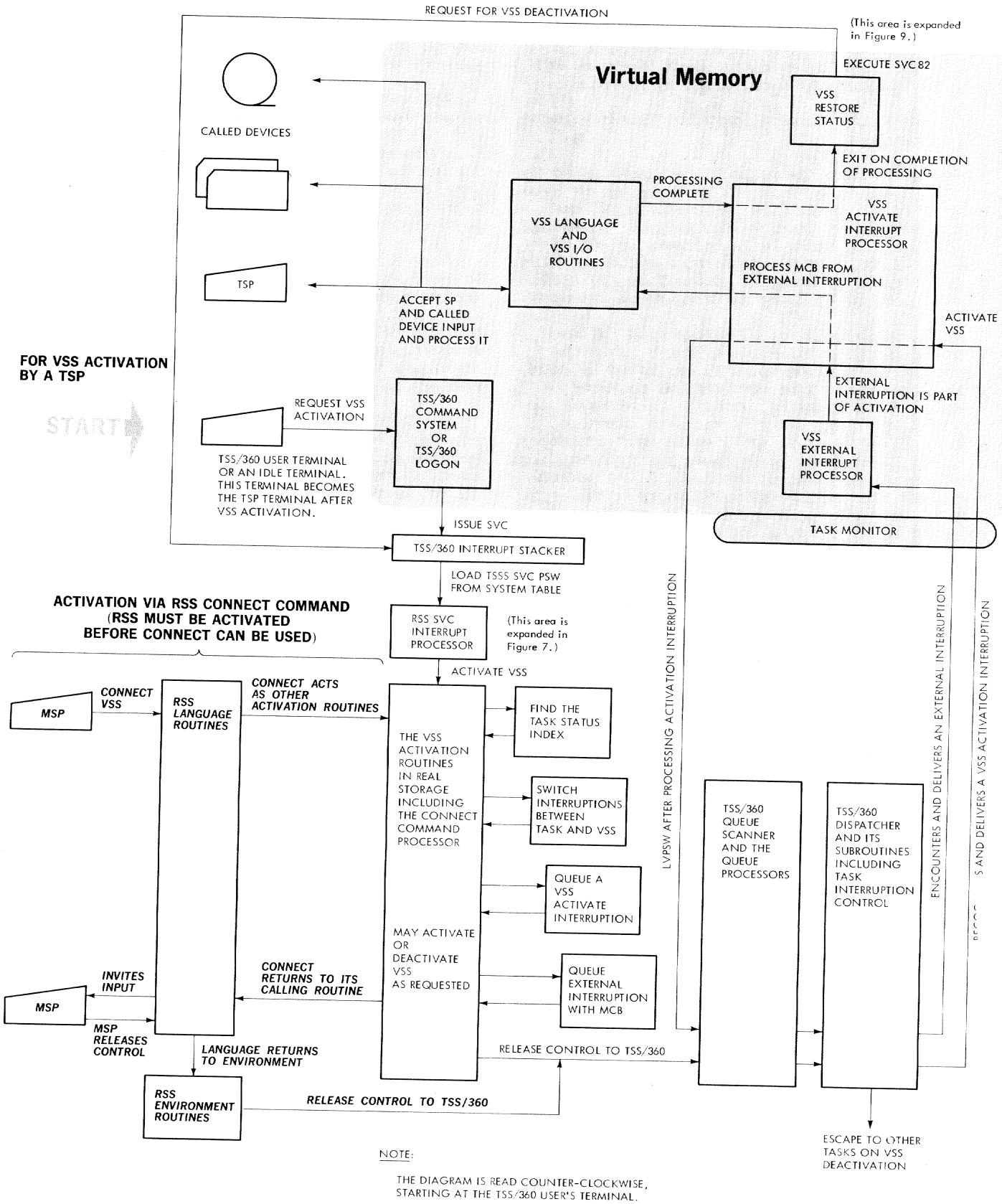
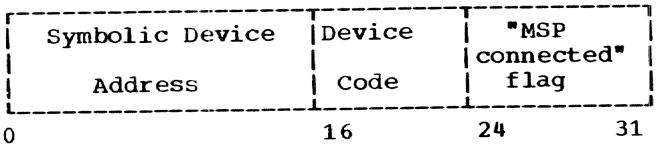


Figure 8. The VSS activation processor

3. Invalid LOGON — Supervisor Storage Allocation (CEAL01) Queue GQE on TSI (CEAAF)

EXITS: This module exits to the TSS/360 LOGON task for all VSS LOGON attempts, as well as for an invalid RSS LOGON attempt via LPSW (old SVC). In the case of a valid RSS LOGON, this module exits to the Channel Interrupt Processor (CEHAC).

OPERATION: This module tests the RSS/VSS indicator in the Message Control Block (MCB). If the indicator is RSS, this routine takes the first steps toward activating RSS (see "Interruption Handling and RSS Activation"). If no MSP is connected, this routine places the symbolic device address (SDA) of the MSP's terminal in the first two bytes of the "MSP connected" field of the TSS/RSS Status Save Area (ESVMSPCN); the device code is placed in the third byte, and the fourth byte is set to indicate that an MSP is connected.



This module then saves the Task Status Index pointer from the Device Group Table (CHADEV) into an RSS Status Save Area field (ESVDVTSI). This TSI pointer is created for the LOGON task as a result of the MSP LOGON attempt. This module substitutes X'000003' as the MSP TSI pointer in the Device Group Table, and saves the LOGON task's task ID in the ESVTSKID field of the Status Save Area.

In order to record the intervening run situation if it occurs, this module sets the "intervening run" flag (ESVNVRUN) in the Status Save Area to X'FF'. (If a RUN command is executed, the RSS Exit module sets this flag to X'0F'.)

After completing initialization for RSS activation this module exits to the RSS Channel Interrupt Processor (CEHAC).

If the indication is VSS, this module calls the Find TSI routine (CEHCF), using the task ID from the MCB as an input parameter. If the input "receiving" task ID is valid, processing continues. If no TSP is connected, and VSS is not active, this routine calls the RSS Interrupt Switching routine (CEHCS), passing a pointer to the found TSI (register 1) and an "activate" indicator (zero in register 2) as parameters. Upon return of control, this routine sets the "two-terminal" flag in the TSI and sets the symbolic device address from the input MCB into the TSISDA field of the TSI.

This module then links to the Queue VSS Interrupt module (CEHCQ) at CEHCQA to build and queue a VSS activate interruption GQE (code 0). On return of control, this module again calls the Queue VSS Interrupt module, but at entry point CEHCQB, to build and queue an external interruption which has an MCB attached for the subject task. The MCB contains TSP information which will be used subsequently by the VSS Activate Interrupt Processor (CZHNV) to initialize the device tables.

Upon return of control, this routine saves the "sending" TSI pointer from the Device Group Table in the RSS Status Save Area. This TSI pointer is assigned to the LOGON task. It is replaced by the "receiving" task's TSI pointer. The "receiving" task is the task within which VSS is to be activated. This module then sets register 15 to zero to indicate successful LOGON, and exits to the TSS/360 LOGON task via LPSW (old SVC). (See Figure 4 for a description of the LOGON task interface with TSSS.)

The following error conditions are recognized by the LOGON RSS/VSS SVC Processor and cause it to reject the LOGON request and return to the LOGON task (via LPSW) with an appropriate non-zero return code.

1. MSP already connected — RC=4
2. TSP already connected to receiving task — RC=8
3. Invalid Task ID return from Find TSI module. — RC=12

(If an MSP disconnects without an intervening RUN command, after logging on, the RSS Disconnect module insures that the LOGON task receives a return code of 16 to indicate the situation.)

RSS Interrupt Switching (CEHCS)

Chart 20

By saving the Task Status Index (TSI) information for the current task and by setting the "VSS active" fields in the original TSI, this module activates VSS. During subsequent processing, any reference to the current TSI will apprise the TSS/360 Supervisor of VSS activation within the subject task. This module performs an equivalent "deactivation" function in which it restores the saved TSI information.

ENTRIES: This module is called at CEHCSA by the RSS/VSS LOGON SVC Processor (CEHDL), the VM AT Execution SVC Processor (CEHDA),

the VSS Command SVC Processor (CEHDV), VSS Exit (CEHDE), and the RSS CONNECT Command Processor (CEHKW). It expects as input parameters: (1) a code indicating the requested operation (register 0), and (2) a pointer to the current TSI (register 1).

MODULES CALLED: In order to build (activation) or return (deactivation) a duplicate TSI, this module calls Supervisor Core Allocation (CEAL01) or Supervisor Core Release (CEAL02), respectively.

EXITS: Exit is to the calling routine.

OPERATION: This routine first tests the input "activation/deactivation" code in register 0.

Activation (Code = 0): This routine links to the Supervisor Core Allocation routine, requesting a TSI-size block of storage. On return of control, it copies the input TSI in the allocated block. This routine places the address of the duplicate TSI in the "VSS alternate TSI pointer" field of the input TSI (TSIVTP). It then sets the "VSS active" flag in the input TSI (TSIVS). By setting the General Queue Entry (GQE) pointer (TSIFPQ), the task interruption count field (TSITIC), and the pending flags (TSIPMF) to zero, this routine dequeues all previously queued interruptions for the subject task, as recorded in the input TSI. The saved, duplicate TSI maintains a record of the task's pending interruptions. As a result, the input TSI is assigned to VSS.

Deactivation (Code = 4): This module resets the "VSS active" flag in the input TSI. It gets the address of the saved, duplicate TSI from the input TSI (TSIVTP). By referring to the duplicate TSI, this routine restores the count field, the pending flags, and the GQE pointer in the input TSI. This routine returns the storage allocated to build the duplicate TSI to TSS/360 by linking to the Supervisor Core Release routine (CEAL02). This deactivation process restores the task its state prior to the VSS activation request.

Find TSI (CEHCF)

Chart 21

Given a valid task ID, this routine develops a pointer to the corresponding Task Status Index (TSI).

ENTRIES: This module is called at CEHCFA by the RSS CONNECT Command Processor (CEHKW), RSS VM Access (CEHCB), RSS Disconnect (CEHBD), and the RSS/VSS LOGON SVC Processor (CEHDL). A valid task ID must be in bytes two and three of register 1.

MODULES CALLED: None.

EXITS: Exit is to the calling routine.

OPERATION: This module serially searches the active list of TSIs within the system, and, if necessary, the inactive list of TSIs. This routine compares the task ID field of each TSI (TSIID) against the task ID it received as input. If this routine finds a match, it passes a pointer to the identified TSI as a return parameter to the calling routine.

If, after searching both the active and inactive TSI lists, this module does not find a match, it stores a "task not found" message control word in register 0, and executes the error return procedures.

The pointers to the first and last TSIs in both the active and inactive lists are maintained in the TSS System Table (CHASYS).

Queue VSS Interrupt (CEHCQ)

Chart 22

This module builds and queues General Queue Entries (GQEs) on an input Task Status Index (TSI) in which the "VSS active" fields have been set. As a result, either a VSS activate interruption or a VSS external interruption is delivered to the task represented by the input TSI.

ENTRIES: This module is called at CEHCQA by the LOGON VSS SVC Processor (CEHDL), the VSS Command SVC Processor (CEHDV), the VM AT Execution SVC Processor (CEHDA), the TSP Asynchronous Interrupt Processor (CEHAQ), and the CONNECT Command Processor (CEHKW) to queue an activation interruption. As input parameters this module expects (1) a pointer to the TSI (register 0), and (2) the requested interruption code in bytes 2 and 3 of register 1.

This module is called at CEHCQB by the LOGON VSS SVC Processor (CEHDL), the VSS Command SVC Processor (CEHDV), the VM AT Execution SVC Processor (CEHDA), and the CONNECT Command Processor (CEHKW) to queue an external interruption for VSS. No input parameters are expected.

MODULES CALLED: In order to build and queue a GQE, this module calls two TSS/360 routines: Supervisor Core Allocation (CEAL01) and Queue GQE on TSI routine (CEAAF).

EXITS: Exit is to the calling routine.

OPERATION: If entered at CEHCQA, this routine saves its entry parameters and calls the Supervisor Core Allocation routine, re-

questing 64 bytes of storage. On return of control, it builds a GQE in the allocated storage, using the general format for an external interruption General Queue Entry. This routine places the address of the TSI in the "TSI pointer" field (GQETSI) and the VSS interruption code in the "external interruption code" field (GQEINT) of the General Queue Entry.

When it has completed construction of a GQE, this routine calls the Queue GQE on TSI routine (CEAAF) at CEAAF2 to queue the GQE as a VSS activation interruption. CEAAF2 is a special entry point reserved for VSS use. On return of control, this module returns control to its calling routine.

For a list of the VSS activation interruption codes and the purpose of each, see "VSS Activation Interrupt Processor (CZHNV)."

If entered at CEHCQB this module calls Supervisor Core Allocation, requesting a block of storage large enough to build a GQE and a Message Control Block (MCB). This block of storage is 152 bytes long. On return of control, this module builds a GQE for an external interruption and attaches an MCB. The MCB is the same in all cases and contains the task ID from the TSITID field of the TSI and the symbolic device address from the TSIDDA field of the TSI. If the XTSI is not in storage, this module calls RSS VM Access to bring it into main storage and then stores the control registers in the last eight words of the MCB in order to pass them to VSS.

This module then sets the external interruption code and calls the Queue GQE on TSI module (CEAAF) at CEAAF2 to queue the external interruption. On return of control, this module exits to its calling routine.

When called at CEHCQC, this module calls CEAL01 for a 64-byte GQE area, which it sets to zero. Then the TSI pointer is placed in the GQE along with program interruption code 52. This module then calls the Queue GQE on TSI module, (CEAAF), and on return of control from CEAAF, returns control to the calling routine.

VSS Command SVC Processor (CEHDV)

Chart 23

This module activates VSS as the result of successful execution of the VSS command by the TSS/360 command language system. The Command System executes an SVC 83 to invoke the TSSS control nucleus.

ENTRIES: SVC 83 is received by the RSS SVC Interrupt Processor (CEHAS), which directs control to this module at CEHDVA.

MODULES CALLED: This module calls the RSS Interrupt Switching routine (CEHCS) to activate VSS, and Queue VSS Interrupt (CEHCQ) to queue a VSS activate interruption and to queue a VSS external interruption.

EXITS: Exit is to the Queue Scanner.

OPERATION: This module locates the current task's TSI via a pointer in the Prefix Storage Area (PSATPT). This module uses the TSI to determine the symbolic address of the SYSIN device (TSISIN). A pointer to the TSI (register 1) and an "activate" indicator (zero in register 0) are the input parameters to the RSS Interrupt Switching routine. Upon return of control, this module sets the "TSP connected" bit in the TSI (TSIVT). It sets the symbolic device address of the TSP terminal field in the TSI (TSISDA) equivalent to the TSISIN field.

This module links to the Queue VSS Interrupt module (CEHCQ) at CEHCQA to build and queue a VSS activate interruption GQE (code 1). On return of control, this module again calls the Queue VSS Interrupt module, but at entry point CEHCQB, to build and queue an external interruption which has an MCB attached for the subject task. The MCB contains TSP information which will be used subsequently by the VSS Activate Interrupt Processor (CZHNV) to initialize the device tables. Upon return of control, this module exits to the Queue Scanner.

As a result of this module's operation, an activation interruption (code 1) and an external interruption are queued against the current task. The authority of the user who issued the VSS command and the absence of another TSP for this task are checked by TSS/360 before SVC 83 is executed.

Virtual Memory AT SVC Execution Processor (CEHDA)

Chart 24

This module activates VSS as a result of execution of an AT SVC previously implanted in private virtual storage or shared virtual storage by VSS for a TSP.

ENTRIES: The RSS SVC Interrupt Processor (CEHAS), upon recognition of an SVC code 80, 84, or 85, calls this routine at CEHDAA.

MODULES CALLED: In order to activate VSS this module calls the RSS Interrupt Switching routine (CEHCS) and the Queue VSS Interrupt subroutine (CEHCQ).

EXITS: Exit is to the Queue Scanner.

OPERATION: This module receives SVCs 80, 84, and 85. SVCs 80 and 85 are the AT SVCs implanted in private virtual storage or shared virtual storage, respectively. SVC 84 is implanted by the AT SVC Execution Processor to signify the completion of AT SVC processing. This routine saves its entry condition for later reference.

This module uses the Task Status Index (TSI) pointer in the Prefix Storage Area (PSATPT) to locate the current TSI. A pointer to the TSI (register 1) and an "activation" indicator (zero in register 0) are the input parameters, which this routine passes to the RSS Interrupt Switching routine. On return of control, this module tests its saved entry condition to determine the appropriate VSS activate interruption code. It calls the Queue VSS Interrupt subroutine at CEHCQB to build and queue the appropriate VSS interruption -- code 2 for SVC 80, code 4 for SVC 84, or code 6 for SVC 85. The interruption code indicates to the VSS Activate Interrupt Processor (CZHNV) which SVC condition caused the activation procedures.

On return of control, this module again calls the Queue VSS Interrupt module, but at entry point CEHCQB, to build and queue an external interruption which has an MCB attached for the subject task. The MCB contains TSP information which will be used subsequently by the VSS Activate Interrupt Processor (CZHNV) to initialize the device tables. This module then exits to the Queue Scanner.

TSP Asynchronous Interrupt Processor (CEHAQ)

Chart 25

This module receives and processes a TSP asynchronous interruption (Attention) from a terminal to which a TSP is connected when TSS/360 is running. It initiates the reactivation of VSS for that TSP.

ENTRIES: The TSS/360 Queue GQE on TSI routine (CEAAF) recognizes a TSP Attention and enters this module at CEHAQA by loading the RSS Queue GQE on TSI PSW from the TSS/360 System Table (SYSRS6).

MODULES CALLED: This module calls the RSS Interrupt Switching routine (CEHCS), if necessary, and calls the Queue VSS Interrupt subroutine (CEHCQ).

EXITS: Exit is to the Queue Scanner.

OPERATION: If the fields in the TSI that indicate the VSS-active state have not already been set, this routine calls RSS Interrupt Switching to activate them. On return, or if VSS has already been activated, this routine links to Queue VSS Interrupt to build and queue a VSS Activate Interrupt. Since the interruption that caused activation is an Attention, the activation interruption is a code 3 interruption.

Note: Since a TSP Attention does not require TSP terminal information to be initialized, an MCB is not required. Consequently, no external interruption is queued for this condition.

VSS Exit (CEHDE)

Chart 26

This module restores the current task to its pre-interruption state, and deactivates VSS as requested by the VSS RUN or DISCONNECT commands. It also processes a "void" command request.

ENTRIES: The RSS SVC Interrupt Processor, upon recognition of an SVC 82, links to this module at entry point CEHDEA. As input parameters, this module requires: a code in register 0 indicating the requested exit condition; the virtual storage address of the task status save area in register 1 (the first two bytes of the status save area have been saved in register 2); and the virtual storage address of the TCT (Terminal Control Table) slot.

MODULES CALLED: This module calls the RSS Interrupt Switching routine (CEHCS) to deactivate VSS. If the requested exit condition indicates a DISCONNECT or "void" command, this routine calls the Supervisor Core Allocation routine (CEAL01) and the Queue GQE on TSI routine (CEAAF) to build and queue an interruption for the task.

EXITS: Exit is to the Queue Scanner.

OPERATION: This module issues a Load Real Address instruction for the address of the task's status save area it received as an input parameter in register 1 and saves that address. It then calls the RSS Interrupt Switching routine (CEHCS), passing it a pointer to the Task Status Index (register 1) and a "deactivate" indicator (4 in register 0) as input parameters. On return of control, it checks the saved input code:

Code	Meaning
0	RUN
4	DISCONNECT
8	Void

12 Program interruption and disconnect

If VSS exit was requested by a RUN command, the task status (as it is recorded in the VSS paging buffer) is stored in the task's XTSI. Any changes or VSS-implanted AT SVCs remain unchanged.

If the originating command was a "void" command, this routine builds a GQE with an MCB attached and calls the Queue GQE on TSI routine (CEAAF) to queue a TSS/360 Attention interruption for the task.

If the exit condition is a program interruption and disconnect, the Queue VSS Interrupt routine (CEHCQ) is called to request program interruption 52. Processing then continues in the same manner for both a disconnect and a program interruption and disconnect. The "TSP connected" bit in the Task Status Index is turned off.

This module checks to see if VSS is connected and, if it is, stores the task status in the TSI and exits to the TSS Queue Scanner. If VSS is not connected, the real address of the TCT slot is loaded, and the exit code is again checked. If it is a void, the "reissue I/O" flag is turned off. The task status is then saved in the TSI, and this module exits to the Queue Scanner.

VSS ENVIRONMENT, VIRTUAL STORAGE

As noted under "VSS Environment, Real Storage," the activation of VSS is initiated at the real storage level by a group of resident modules in the control nucleus. A VSS activation interruption is processed at the virtual storage level when the task receives its next time slice. If the activation interruption code signifies an Attention, the interruption is processed immediately, resulting in activation if VSS is not currently executing. Otherwise, activation procedes as follows:

1. A flag is set in the VSS Status Save Area to indicate that the VSS activation sequence is in progress.

2. A flag is set in the Interrupt Storage Area (CHAISA) to indicate to the Task Monitor that VSS is active.

3. The ISA and its associated task status are saved.

4. Interrupts are enabled via LVPSW.

When interruptions are allowed, the external interruption queued by the resident modules in the control nucleus is delivered to the VSS External Interrupt Processor. Since the external interruption is in the activation sequence, the VSS External Interrupt Processor delivers the interruption to the VSS Activate Interrupt Processor. The activation sequence continues as follows:

5. Flags are set in the VSS Status Save Area to indicate (1) that VSS is active, (2) conversational mode or AT mode, and (3) one-terminal or two-terminal case.

6. If the activation is an initial connection, the Language Input Device Table (CHALCR) is initialized from information in the Message Control Block appended to the external interruption's General Queue Entry. In any case, if the activation is in conversational mode, Language Control is then called; if in AT mode, the VSS AT Execution Processor is called instead.

7. Upon return from Language Control (with a RUN, DISCONNECT, or "void command" indication), task status is restored and linkage is made to the VSS deactivation routine in real storage via execution of SVC 82.

If VSS is already active, the processing of a TSP attention at the virtual storage level is a variation of Attention processing in the above sequence. In that case, when the Attention interruption is processed, a flag is set to mark the interruption as "received"; then the old VSS VPSW is loaded.

The interaction between the VSS environment routines in virtual storage is shown in Figure 9.

If Language Control returns an indication that a "void command" was executed (EOB without any input, without prior continuation character), the exit from virtual processing (via SVC 82) is unchanged. The real storage VSS Exit routine recognizes the exit condition and queues a TSS/360 Attention against the restored task.

Other VSS environment processing in virtual storage includes the forcing of a DISCONNECT because the task is being logged off; acceptance of a virtual program interrupt, code X'30', in order to notify the TSP of an error (addressing an I/O device not allocated to the task); and the VSS Real Core Access function, which passes parameters to RSS Real Core Access via the execution of an SVC.

The VSS Message Writer routine is analogous to the RSS Message Writer routine.

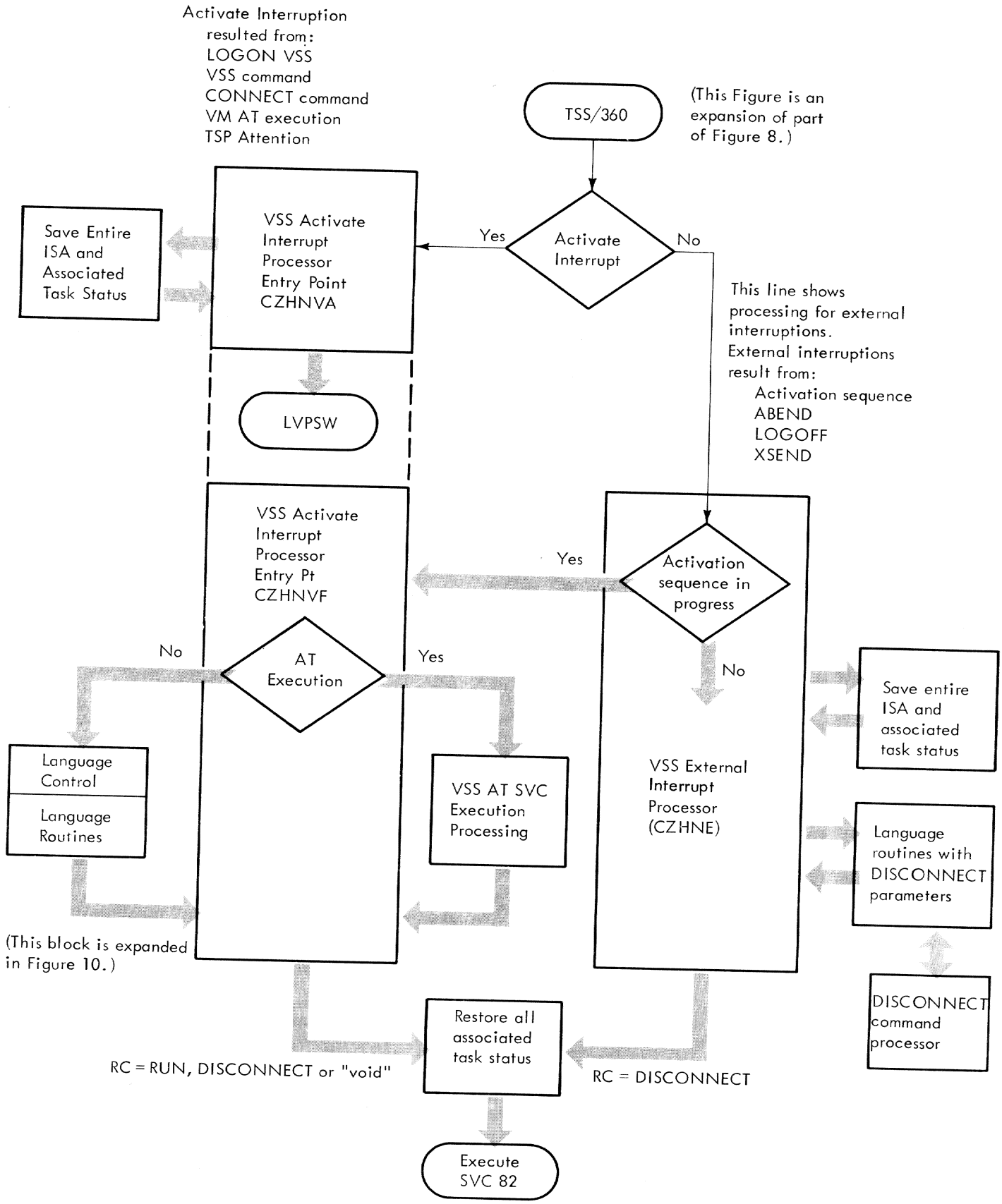


Figure 9. Overview of VSS environment, virtual storage

Attributes

The VSS environment modules described in the following section are, as with the remainder of VSS, resident in the Initial Virtual Memory of the task. They are serially reusable and nonrecursive, and they execute in privileged mode.

VSS Activate Interrupt Processor (CZHNV)

Charts 27,28

This module:

1. Accepts and processes the VSS activate interruptions in the VSS activation sequence that are queued by the resident modules in the control nucleus.

2. Processes the external interruptions in the VSS activation sequence.

3. Accepts and processes, when VSS is active, all unexpected SVC, asynchronous, timer, and data set paging interruptions.

In the activation sequence this module saves task status, examines the code associated with the activation interrupt, determines the state of VSS, and takes the appropriate action.

ENTRIES: Primary entry to this module is defined by the instruction address of the new VSS VPSW; the entry point name is CZHNVA. This module is called at CZHNVA by the TSS/360 Dispatcher when an activate interruption occurs. This module is also called at CZHNVF by the VSS External Interrupt Processor (CZHNE) when it has encountered an external interruption in the VSS activation sequence.

The following entry points are used when VSS is executing and the TSS/360 Dispatcher recognizes one of the listed conditions:

Entry Point	Entry Condition
CZHNVB	Unexpected SVC interruption
CZHNVC	Asynchronous interruption
CZHNVD	Timer interruption
CZHNVE	Data set paging interruption

MODULES CALLED: This module calls the TSS/VSS Status Save routine (CZHPS), when processing the activate interruption. When processing the external interruption, it calls either the VSS AT SVC Processor (CZHZA) or VSS Language Control (CZHXC), depending on the activation condition.

EXITS: Exit from this module depends on the entry condition.

1. If this module is entered at CZHNVA it generally exits by loading the old VPSW. This, in effect, enables interruptions for the subject task, allowing the external interruption to be executed. However, on TSP Attention, this module performs the activation sequence without an external interruption, and exits to VSS Restore Status, except when VSS is already executing, in which case it exits by loading the old VPSW.

2. If this module is entered at CZHNVF on an external interruption it exits to VSS Restore Status (CZHPR).

3. If this module is entered at CZHNVB, CZHNVD, or CZHNVE, it exits to VSS Restore Status with a return code of zero (RUN).

4. If this module is entered at CZHNVC, it exits by loading a virtual PSW.

OPERATION: When entered at CZHNVA this module checks for activation interruption code 3 (TSP attention). If the interruption was caused by a TSP attention, and VSS was not already active, the VSS active flag is set on, and the CKALOC macro is issued. If the return code from CKALOC indicates a successful operation, control is passed to VSS Language Control (CZHXC). When control returns, this module exits to VSS Restore Status CZHPR. If VSS was active at the time that the attention interruption was received, the TSP "attention received" flag is turned on, the registers are reloaded, and the module exits via a Load PSW instruction.

If the interruption is not a TSP attention interruption, the VSS activation sequence is performed. The "VSS active" flag is set in the Interrupt Storage Area (CHAISA), and VSS Status Save (CZHPS) is branched to in order to save task status at the time of VSS activation. When control is returned from Status Save, external interruptions are enabled and the module exits by loading the virtual old PSW.

When this module is entered at CZHNVF from the VSS External Interrupt Processor (CZHNE), it checks the interruption codes in bytes two and three of the virtual old PSW. If the code is zero (resulting from a LOGON VSS command), the TSP device is added to the task device list. If the code is one (VSS command) or five (RSS CONNECT), processing then continues in the same way as for code zero -- a test is made for VSS connected. If VSS is connected, this module branches to an internal subroutine.

For code one, this subroutine checks for outstanding synchronous I/O interruptions. If there are any, the WAIT macro is issued, and at its completion, the check for I/O interruptions is made again. If there are none, the CKALOC macro is issued to check for MTT operation in the active devices and assume control of I/O. For codes 0 and 5, the synchronous I/O interruptions check is omitted. If the CKALOC return code indicates a successful operation, control is returned to that portion of the module that branched to this subroutine via register eight. If CKALOC operation was unsuccessful, the VSS Message Writer Routine (CZHNM) declares a permanent I/O error, and control is returned to the calling routine.

After the CKALOC processing, or if VSS was not connected, the "VSS connected" flag is set in the ISA, the Qualify Table is initialized, and the address of the physical path for the device is placed in the SADT. Then the SSDAT is searched for the SDA. If the SDA is not found, a system error is issued, and control is passed to VSS Restore Status. When the SDA is found, the ADDEV macro is issued to add the device to the task device list, and this routine branches to the internal subroutine described above.When Language Control returns, exit is to VSS Restore Status.

If the interruption code is 2, 4, or 6, the "VSS active" flag is set in the TSS/VSS Status Save Area. If VSS is connected, the subroutine that checks for synchronous I/O interruptions outstanding and issues CKALOC is called. After CKALOC has successfully completed its operation, or if VSS was not connected, control is passed to the VSS AT Execution Processor (CZHZA). When control returns, exit is to VSS Restore Status.

When the entry is at CZHNVB, the result of an unexpected SVC interruption, a SYSER is issued, and exit is again to VSS Restore Status with a return code of zero (RUN).

An asynchronous I/O interruption results in this module's being called at entry point CZHNVC. The SADT is searched for a device entry matching the one which received the interruption. When one is found, if the interruption was expected, the "asynchronous interruption received" flag is turned on, and exit is to the point of interruption via a Load Virtual PSW instruction. If the interruption was not expected, a new asynchronous I/O interruption virtual PSW is loaded to pass control to the Task Monitor.

An unexpected external (timer) interruption results in this module's being entered at CZHNVD where a SYSER is issued, and control is passed to VSS Restore Status with a RUN return code. When CZHNVE is the entry point, an unexpected data set paging interruption has occurred, and a SYSER for a major software error is issued. Exit is again to VSS Restore Status with a return code of zero.

VSS Status Save Routine (CZHPS)

Chart 29

This routine saves TSS/360 status for the task within which VSS is being activated or within which it is operating.

ENTRIES: This routine is called by the VSS Activate Interrupt Processor (CZHNV) and by the VSS External Interrupt Processor (CZHNE) at CZHPSA.

MODULES CALLED: None.

EXITS: Exit is to the calling routine.

OPERATION: Saving the relative TSS/360 task status involves saving the entire Interrupt Storage Area (CHAISA) into a special CSECT (CZHPSR) and then moving the required data from the ISA into a predefined Status Save Area (CHAEVS) within VSS. The required data includes:

1. The virtual new PSWs
2. The virtual old PSWs
3. General purpose registers
4. Floating point registers
5. Control registers
6. The current virtual PSW (that is, interruption code and instruction counter).

For further information concerning the TSS/VSS Status Save Area see, Appendix F.

VSS External Interrupt Processor (CZHNE)

Chart 30

This module accepts and processes the VSS external interruptions that occur while VSS is active.

ENTRIES: This module is entered at CZHNEA, which is defined in the new external VPSW. Entry to this routine may be from the TSS/360 Dispatcher on a VSS external interrupt, or it may be the result of a type 1 call from the TSS/360 LOGOFF or ABEND modules.

MODULES CALLED: This routine calls the VSS Status Save routine (CZHPS) and VSS Language Control (CZHXC).

EXITS: If the external interruption is a part of the activation sequence, this routine exits to the VSS Activate Interrupt Processor (CZHNV) at CZHNVF. Otherwise this module exits to the VSS Restore Status routine (CZHPR).

OPERATION: On entry, this module sets the "input mode" flag in the Input Device Table (CHALCR) to indicate a special entry condition to Language Control, which it then calls. As an input parameter this module moves an 11-byte data field into CHALCR. The data used as input to Language Control consists of a one-word length field (X'0000000A') and a text field initialized with the DISCONNECT command.

To indicate successful processing of the DISCONNECT command, Language Control returns code of four. If the saved entry condition is code 16, this module exits to VSS Restore Status with an output parameter of 16 to request a "disconnect and return to calling routine" as the VSS exit condition. If the saved entry condition is not 16, this module exits to VSS Restore Status with an output parameter of 12 to request an XSEND disconnect as the exit condition.

VSS Program Interrupt Processor (CZHNP)

Chart 31

This routine processes the VSS program interruptions that occur while VSS is executing.

ENTRIES: When the TSS/360 Program Interrupt Processor encounters a VSS program interruption, it calls this module at CZHNPA by loading the new program VPSW.

MODULES CALLED: None.

EXITS: The occurrence of a code 30 (hexadecimal) program interruption causes exit to be made to a special entry point within VSS I/O Initiation/Posting (CZHSBC). Otherwise this module exits by loading the old program VPSW.

OPERATION: This module determines if the interruption code is X'30'. (An attempt by VSS I/O Control to perform I/O on a device not allocated to the task caused the program check.) In this case, this routine places a return code of four in register 15 and exits to a point within VSS I/O Initiation/Posting. The I/O calling routine will then be returned a message control word so that the system programmer may be informed of the I/O failure.

If this module encounters a paging exception interruption (code 17), it sets up an error message parameter and returns to the routine that called the routine that caused the paging exception.

If this module encounters an addressing exception (code 5) when VSS VM Access (CZHPB) is in control, it sets up an error message parameter and returns to the routine that called VSS VM Access. In both cases, this module sets a return code of 8 in register 15 to indicate an invalid virtual storage address.

If anything else caused the interruption, this module effectively ignores it, exiting via LVPSW.

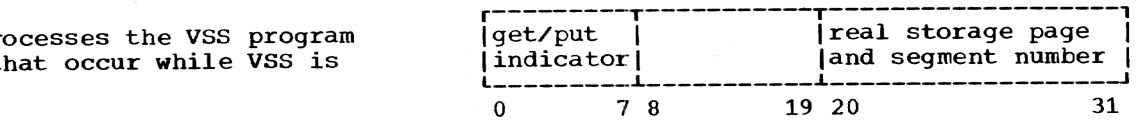
VSS Real Core Access (CZHPA)

Chart 32

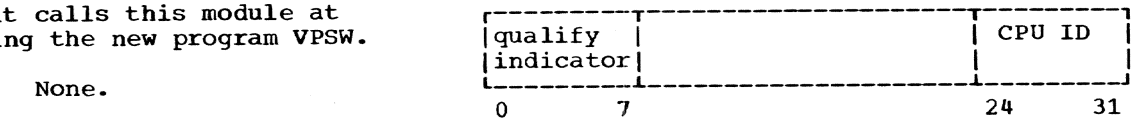
This routine initiates communication with RSS in order to access real storage while operating in a task environment.

ENTRIES: This module is called at CZHPAA by those VSS modules wishing to acquire or replace a page of real storage. These modules include the VSS copies of the REMOVE (CZHYR), SET (CZHYS), and DUMP/DISPLAY (CZHYD) Command Processors. Input to this routine is:

Register 1:



Register 2:



where "qualify indicator" is either RM qualified or RM unqualified and
CPU ID is either 1, 2, 3, or 4.

MODULES CALLED: None.

EXITS: Exit is to the calling routine.

OPERATION: TSSS does not allow any direct violation of the real/virtual boundary. Thus, this routine invokes RSS to acquire or replace a page of real storage for VSS.

This module tests the input "get/put" indicator, it implants the appropriate SVC in the first two bytes of the VSS paging buffer, and executes the SVC remotely.

The resulting interruption causes RSS to be activated and loaded. The RSS SVC Service Processors routine (CEHDR) gets con-

trol after the activation procedures are completed. As requested by the input parameters in the RSS Status Save Area (supplied by this routine), RSS Real Core Access performs the "get/put" operation.

For a "get" request this module supplies these input parameters:

Reg. 0 Input address of the requested page

Reg. 1 Virtual storage address of the VSS paging buffer

Reg. 2 Qualify Table Entry

For a "put" request, this module supplies the above parameters, and also saves the two bytes of the paging buffer, overlaid by the SVC, in bytes two and three of register 3.

RSS Real Core Access performs the "get/put" logic between the real address supplied as an input parameter and the RSS paging buffer. The RSS SVC Service Processor is responsible, in case of "get", for moving the contents of the RSS paging buffer into the VSS paging buffer.

If the request is "get", this module implants and executes an SVC 65. If the request is "put", this module implants and executes an SVC 66. RSS returns control to this module via LPSW upon completion of the requested operation. This module then returns control to its VSS calling routine.

VSS VM Access (CZHPB)

Chart 33

This module enables a TSP to refer to a virtual storage page. Upon special request, this module may determine whether a virtual page is shared or not by issuing an SVC 73.

ENTRIES: This routine is called at CZHPBA by the AT SVC Processor (CZHZA), the AT Command Processor (CZHYA), the DUMP/DISPLAY Commands Processor (CZHYD), the SET Command Processor (CZHYS), VSS Symbol Resolution (CZHWS) and the REMOVE Command Processor (CZHYR). This module expects the following input parameters:

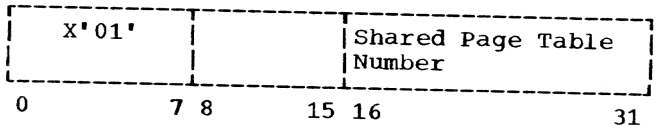
Reg. 1 -- a virtual storage address

Reg. 2 -- a "get/put" indicator in byte 0
X'AA' -- get
X'BB' -- put
X'CC' -- shared page determination request

MODULES CALLED: None.

EXITS: This module exits to its calling routine. If the input request is "get", the virtual storage page requested is in the VSS paging buffer. If the input request is "put", the contents of the paging buffer are placed at the location specified by the virtual storage address. When applicable, the shared segment indicator and shared page table number is placed in register 1, which is in the following format if the virtual storage address is contained in shared storage:

Register 1



If the page is not shared, register 1 contains all zeros.

OPERATION: This routine sets the VSS VM Access active flag (EVSVMAC) in the TSS/VSS Status Save Area (CHAEVS). This flag is set so that addressing exceptions (code 5) and paging exceptions (code 17) may be handled properly by the VSS Program Interrupt Processor when they are caused by VSS VM Access.

This module then tests the "get/put" indicator in register 1. If the request is "get", this module moves the contents of the virtual storage page specified to the paging buffer to (CZHPAR). If the request is "put", this module moves the contents of the paging buffer to the storage location of the virtual storage page specified.

A special indicator (X'CC') requests the RSS VM Access routine (CEHCB) to determine if the page is shared. In order to invoke the RSS VM Access routine, this module implants and executes an SVC 73. (See "SVC Service Processors.")

Following execution of SVC 73, or after all the "get/put" processing is complete, this module sets off the "VSS VM Access active" flag and returns control to its calling routine.

VSS Message Writer Routine (CZHNM)

Chart 34

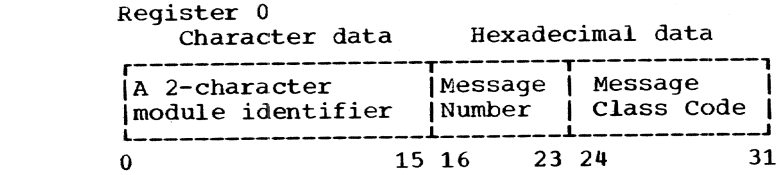
This routine passes diagnostic information to the TSP. VSS diagnostic messages are divided into three classes:

Class 0	VSS I/O errors
Class 1	User (TSP) errors
Class 2	TSSS System errors

The messages are written by VSS I/O Control, and the output is on the TSP's terminal.

ENTRIES: The Message Writer routine is called at CZHNMA by VSS Language Control (CZHXC) and VSS Scan Control (CZHXS). As an input parameter, this module expects a message control word in register 0.

Register 0



MODULES CALLED: This routine calls VSS I/O Control (CZHSA) to initialize the VSS I/O system and print the message.

EXITS: Exit is to the calling routine.

OPERATION: This routine tests the "TSP connected" flag in the Status Save Area (EVSMODE). If no TSP is connected to the executing task, this routine executes a normal return to the calling routine, as if it had successfully completed the message operation. (As a result of AT execution with global qualification, VSS may be executing within a task to which no TSP is connected and for which no TSP terminal is addressable for printing a diagnostic message.)

If a TSP is connected to the task in control, VSS Message Writer uses the input message number and message class code as an index to select the message text. Each of the message classes includes messages that contain pertinent information concerning each type of error recognized. The TSP is responsible for analyzing the message and taking the appropriate action.

A given message may consist of one or more lines of output, depending upon the error. Each VSS message has its first line formatted as follows:

CZHidxxx 24-character (maximum) message text

where id represents the module that encountered the error and xxx represents the class code and message number.

In addition, Class 0 messages (VSS I/O) may have additional lines of output:

Symbolic device address

Seek address

Physical path address

CSW

PSW

Sense data

Alternate path

Channel logout area

If VSS is in AT execution mode, this module calls VSS I/O to print the source statement in addition to the message text.

VSS Restore Status (CZHPR)

Chart 35

This module provides the virtual address of the VSS Status Save Area (CHAEVS), restores the task status, saved by the VSS Status Save routine, and requests the restoration of the task (the deactivation of VSS) by executing SVC 82.

ENTRIES: This module is called at CZHPRA by the VSS Activate Interrupt Processor (CZHNV) and the VSS External Interrupt Processor (CZHNE). As an input parameter it expects a code indicating the desired VSS exit condition (RUN, "void," or DISCONNECT).

MODULES CALLED: None.

EXITS: This module exits by remotely executing an SVC 82, which causes entry to the resident environment module responsible for deactivating VSS -- VSS Exit (CEHDE).

OPERATION: This module restores the entire Interrupt Storage Area (CHAISA) from the special VSS PSECT save area (CZHPSR). If the exit condition is disconnect, this module turns off the "VSS connected" flag (ISAVSC) in the ISA.

This module sets the address of the VSS Status Save Area (CHAEVS) in register 1, saves the requested exit condition code in register 0, and saves the contents of the first halfword of the Status Save Area in register 2. This module then inserts SVC 82 into the first halfword of the Status Save Area. It then executes the SVC 82, which will cause entry to the VSS Exit module (CEHDE) in real storage.

This module then sets up the linkage to VSS exit (CEHDE), and if the SDA is not zero, issues the CKALOC macro to determine if the terminal is being operated under MTT, and if not to relinquish control over terminal I/O. If the return code from the macro instruction is a 3, SVC 82 is issued to return control to VSS Exit. If the return code is not a 3, a system error is declared.

INTRODUCTION

The routines that make up TSSS Language Control perform the functions requested by the system programmer, calling upon TSSS I/O and Environment routines for service as needed. Language Control is linked to by TSSS Environment whereupon input is accepted and processed. Language Control interfaces (via I/O) with the system programmer, depending on the commands being issued, eventually linking back to TSSS Environment for deactivation. Figure 10 provides an overview of TSSS Language Control.

MODES OF OPERATION

Conversational Mode: When input is received from the system programmer terminal, the mode is "conversational" until a RUN or DISCONNECT command is encountered. In this mode, a $ is written at the terminal upon activation, upon completion of all operations requested by the last input statement, or upon aborting part or all of a command statement. It signifies that TSSS is ready to accept the next input statement from the system programmer.

Conversational mode is ended by a CALL, RUN, or DISCONNECT command, and is reentered by an END or STOP command in call mode, a STOP command in AT mode, or an Attention (except for a local MSP) in run mode. Certain error conditions also cause resumption of conversational mode.

AT Mode: AT SVCs implanted in TSS/360 (via execution of the AT command) upon their execution, activate TSSS. An "AT mode" bit is set and Language Control initiates processing of a stored dynamic statement, instead of reading an input device. RUN is implied when processing of the dynamic statement has been completed, unless a CALL, STOP, or DISCONNECT command is encountered, or an Attention is received from the system programmer's terminal.

Call Mode: Execution of the CALL command causes the active entry in the Input Device Table to be changed to contain the address of a card reader or tape drive instead of the system programmer's terminal. Language Control then causes the new input device to be read in the same way as the terminal, until an END, RUN, STOP, or DISCONNECT command is encountered, an Attention is received from the system programmer's terminal, or physical end-of-file or an error condition causes control to be returned to the terminal (resumption of conversational mode).

A CALL command may be executed in AT mode. In that case, each subsequent statement is processed in AT mode, even though it emanates from a card reader or tape drive.

Run Mode: The alternative to the TSSS executing modes (conversational mode, call mode, and AT mode) is run mode -- the state of TSSS when TSS/360 is executing but a system programmer is still connected.

PROCESSING TSSS INPUT

The processing of input is essentially the same for RSS and for VSS, and for all modes. The input buffer is 256 bytes long, although card input can be only 80 bytes in length. Each input statement is first translated into a polish string, before being executed.

The Polish String: The source statement accepted by Language Control is converted into Polish notation, and hereafter referred to as the "polish string". The polish string example, appearing in the Source to Polish module description, illustrates the Early Operator Reverse Polish used by TSSS.

This polish string is constructed in such a way that, when it is completed and subjected to a left-to-right scan, operands are encountered and resolved before the related operator is encountered, and the requested operation performed on the data fields designated by the operand elements. Keywords (all commands except IF, which is treated internally as an operator), having the greatest relative weight, are encountered last. Thus, the keyword execution subroutines, which are invoked to service the system programmer's request, have completely resolved operands with which to work.

The SCB: The unit of common storage used to define symbols and for passing parameters between language routines is the Symbol Control Block (SCB). Pointers to skeleton SCBs, to completed SCBs, and to a parameter list containing pointers to the active SCBs are the input and return parameters.

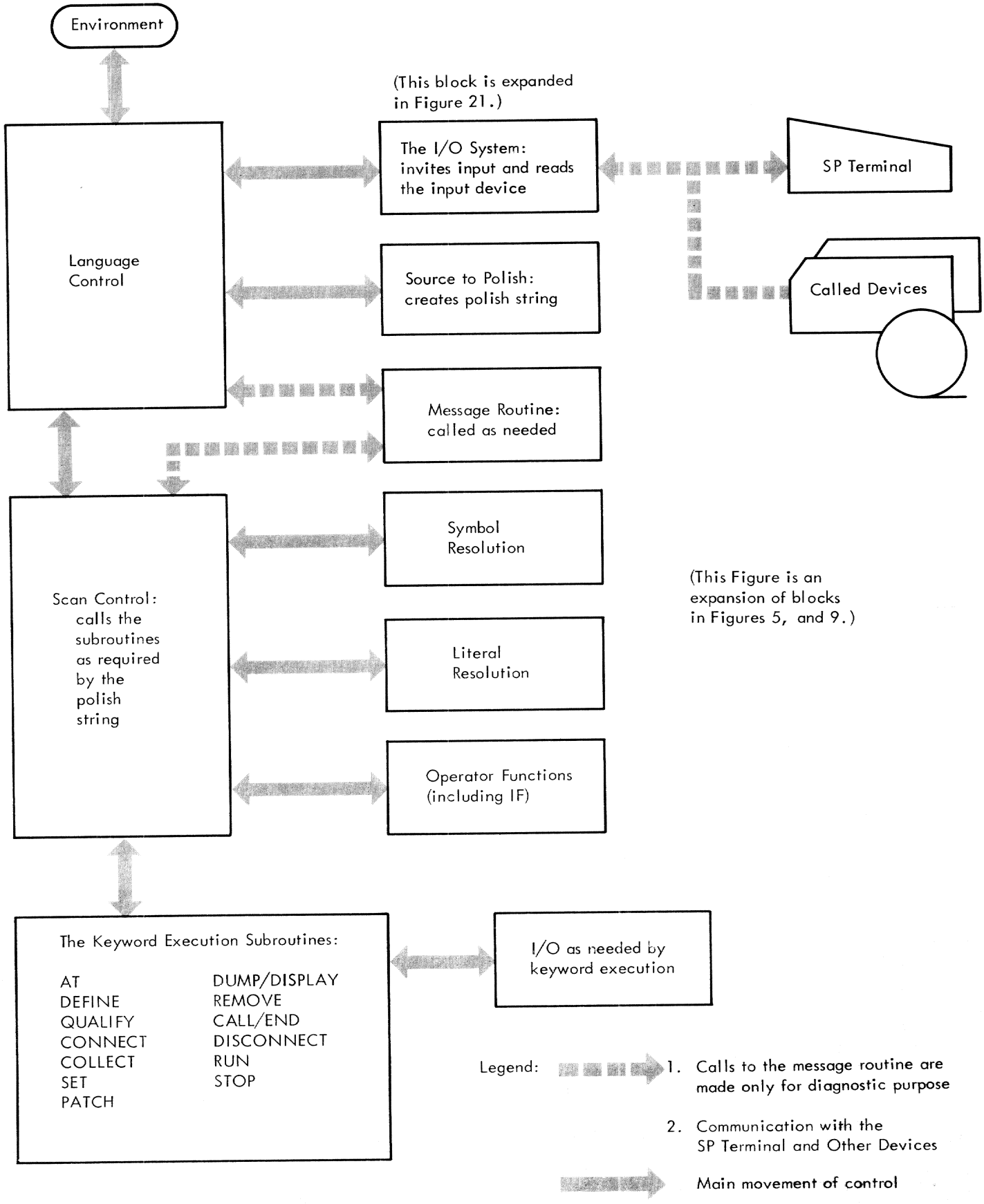


Figure 10. Overview of TSSS language

For example, the Scan Control routine provides a skeleton SCB for each symbol or literal, and a pointer to each SCB as input to the symbol and literal resolution routines. It receives the same pointer in return, with the SCB filled in (that is, resolved). There are 12 of these "skeleton" SCBs, and they reside on the Scan Control parameter list.

Each SCB is 48 bytes long. The fields of an SCB are used to define symbol (or literal) attributes and to point to the data field represented by the SCB. (It should be noted that a data field address is the sum of the value in the "base address" field and the value in the "pointer" field, although the COLLECT command execution routine is the only one that manipulates the pointer field in normal execution without the field itself being designated an operand.) The SCB fields also convey other information in intermodule communication.

When the SCB is used as a means of recording symbol definition, the SCB is further defined by function as either a temporary or a permanent SCB. A permanent SCB is one used to record system symbol definitions. These SCBs compose an assembled symbol table of 24 entries. (Note that $B, $P, $L, $T, $S, and $ID, defined as system symbols in the TSSS Systems Reference Library publication, are treated as operators internally.)

Temporary SCBs are those used to define SP symbols (those created by the system programmer with the DEFINE command). They are created dynamically during TSSS operations and are resident on either the SP Symbol Table or the Global Symbol Table during a given SP's terminal session. The data fields are identical in both types; a permanent SCB may be copied directly into a temporary SCB without a change in format. (See Figure 11 for the format of the SCB; see also Appendix E.)

Defining an SP symbol involves building the SCB in the next space in a reserved block of storage (the symbol table) and, if the symbol has "independent definition" as opposed to being an "alias," allocating the specified amount of storage at the other end of the table, if available. (See Appendix E for a description of the SP Symbol Table.) If sufficient space is not available an error message is written out to indicate it. There is one such private symbol table for the MSP, one for each TSP, and one global table that contains the global SP symbols defined by all TSPs and used with global ATs (see "RSS/VSS Differences," below).

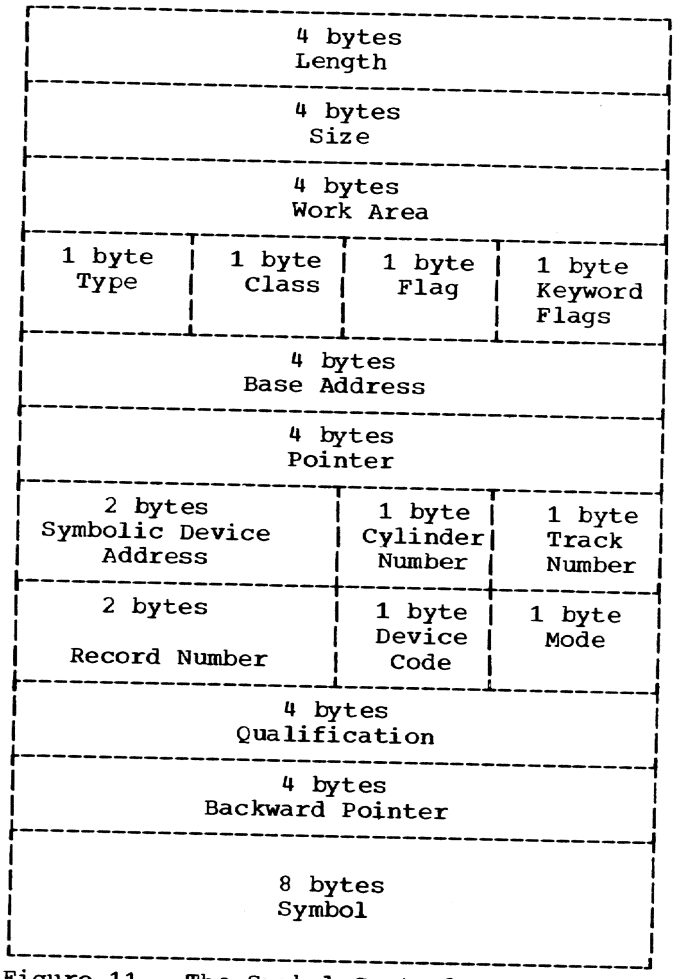


Figure 11. The Symbol Control Block (SCB)

Resolution of a symbol involves searching one of the above symbol tables or the appropriate table of TSS/360 external symbols and filling in a temporary SCB accordingly.

General Operation: In the general operation of TSSS language, Language Control accepts an input command string, causes it to be translated into polish notation, and calls Scan Control to direct the processing of the elements in the polish string. The keyword execution subroutines call I/O Control as needed to perform their functions.

These subroutines perform thorough error checking, and, if an error occurs, execute the "error return procedures" by passing a return code in register 15 and message control word in register 0 to Scan Control. Depending on the severity of the error, Scan Control either: (1) calls the Message Writer routines itself (return code 16) and, on return, continues the scan or (2) terminates the scan (return code 4) and returns error parameters to Language Con-

trol, so that the latter may call the message routines.

A RUN or DISCONNECT command (or the implied presence of one of them in certain instances) causes language processing to cease. These command processors use return codes to indicate, first to Scan Control and then to Language Control, that the latter is to return to the Environment routine that called it. (See Figure 10.)

RSS/VSS Differences: Although the language routines are functionally identical for RSS and VSS, some differences in operation and output exist. The most important difference lies with the AT command. As mentioned earlier, an AT command causes the AT Command Processor to implant the appropriate SVC in the location specified, save the original instruction from that location, and save the remainder of the source input statement. An AT SVC may be implanted in the real storage of the TSS/360 Supervisor, in a task's virtual storage, or in shared virtual storage.

In order to keep a record of the circumstances surrounding the implantation of an AT SVC, the AT Command Processor builds an AT Control Block (ACB), in which it saves the original instruction and provides a pointer to the dynamic statement (remainder of the source statement). Both the ACB and the dynamic statement are saved in an AT Table, which is constructed dynamically in a reserved block of storage in a manner similar to the building of an SP symbol table. (Each AT table is one page in length.) If a TSP wishes to implant an AT in real storage, RSS is activated to perform the service. The RSS AT Command Processor (CEHKA) implants the AT SVC as if it had been requested by an MSP. The records of all RSS-implanted ATs are kept in the MSP's AT Table.

An AT may be qualified as private or global. In most cases, the ACBs and dynamic statements of ATs implanted with a private qualification are kept in the system programmer's private AT Table. However, the record of a TSP-implanted AT in shared virtual storage is kept in the Shared Global AT Table, to which all tasks have access.

If an AT is implanted in shared virtual storage with a private qualification, every task which encounters the AT SVC executes the SVC but does not execute the associated dynamic statement. If this AT has global qualification, every encountering task executes the SVC and the stored source statement, even if a TSP has never been connected to the encountering task. Because there is no means of inter-task communication, all TSP globally qualified ATs are recorded in the Shared Global AT Table, so that the VSS routines in the encountering task may refer to the ACB corresponding to the executed AT SVC.

An MSP's globally qualified ATs (which are in shared virtual storage) are no different from his private ATs in virtual storage, except that execution of the dynamic statement is dependent on qualification.

Each SP symbol in a dynamic statement accompanying a TSP's globally qualified AT must be globally qualified, so that all tasks, by referring to the Global Symbol Table, have access to their SCBs, as well as to the data field which the symbol defines.

When a system programmer issues a DISCONNECT command, all AT SVCs and the related data in the AT Tables are removed as one of the functions of the DISCONNECT Command Processor, with the exception of TSP ATs implanted by RSS in real storage.

ATTRIBUTES

The RSS versions of the Language modules are non-resident and serially reusable. They operate in the supervisor state with DAT active. The VSS versions are resident in each task's IVM. They are serially reusable and operate in the privileged mode.

LANGUAGE PROCESSING ROUTINES

AT SVC Processor (CEHJA/CZHZA)

Charts 36,37

This routine locates a dynamic statement supplied by the system programmer and stored at the time of AT implantation. It directs execution of the statement by the language area and then provides the means for restoring TSS/360 (the implied RUN that ends a dynamic statement).

ATTRIBUTES: The RSS version of this routine is resident and executes with DAT active. Both copies are serially reusable and non-recursive. The CSECTs are read-only.

ENTRIES: The RSS copy of this routine is called by the RSS SVC Service Processor (CEHAS) at CEHJAA while processing SVCs 67, 68, 69, 71, and 72. The VSS copy is called by the VSS Activate Interrupt Processor (CZHNV) at CZHZAA while processing SVCs 80, 84, and 85.

MODULES CALLED: This module calls RSS or VSS Virtual Memory Access (CEHCB, CZHPB) during the processing of an AT SVC 72 or 85 to determine the shared page number.

This module also calls Language Control (CEHLC/CZHXC) to process the stored dynamic statement. In RSS, when dealing with RSS-implanted ATs in virtual storage, this module calls RSS Symbol Resolution (CEHMS) to locate the public CSECT for the implantation of a return SVC.

Under certain error conditions, this module calls the REMOVE Command processor (CEHKR/CZHYR) and the Message Writer routines (CEHCM, CZHNM).

EXITS: Exit is to the calling routine.

OPERATION: The current PSW (stored when TSSS received control) is used to construct an AT ID, which serves as an index for searching the appropriate AT Table. If the SVC code indicates an AT in shared virtual storage (SVCs 72 and 85), the AT ID is the current PSW instruction address plus one, with the appropriate Shared Page Table (SPT) number substituted for the 12 high-order bits of the address. For SVCs 69, 71, and 80 the AT ID is the current PSW instruction address. SVCs 67, 68, and 84 do not have corresponding AT Control Blocks (ACBs) in the AT Tables.) (See Appendix G.)

This module checks for invalid SVCs, printing an error message for a conversational task and causing a program interruption to a nonconversational task if the SVC issued is not valid. It then finds the appropriate ACB in the system programmer's AT Table (RSS or VSS) or the shared AT Table (VSS only), depending on the SVC code. If it does not find the ACB, it rechecks the location of the AT. If the SVC is there, it prints an error message and returns to the calling routine. If the SVC is not there, it just returns.

When the ACB is found, this module checks for an SVC in shared virtual storage. If the AT in question is not in shared virtual storage, this module executes the "link-out" procedures: it (1) moves the command text and an "end of buffer" character into the Language buffer area (CHALCR), (2) turns on the "input in storage" flag in the Input Device Table (CHALCR), (3) sets the AT mode switch on, (4) saves the old Qualify Table, (5) builds a new Qualify Table from information in the ACB, and (6) links to Language Control (CEHLC/CZHXC) to process the AT command string. All RSS-implanted ATs are executed in this manner.

If the ACB is for a VSS-implanted AT in shared virtual storage, this module checks for global qualification. If the ACB is qualified globally or if it is applicable to the executing task, this module executes the "linkout" procedures already described. If it is not qualified globally or is not applicable to the task, this module bypasses the "link-out" procedures for processing the command statement and checks for chaining (more than one AT command specifying a given instruction location). This module loops to examine each chained ACB, if any are found to exist.

Upon return from the "link-out" to Language Control, this module restores the old Qualify Table and checks the condition of the "RUN with an operand" flag in the appropriate Status Save Area (ESVRNAT in RSS, EVSRNAT in VSS). This flag is set by the RUN Command Processor (CEHKN/CZHYN) when it has an operand with the RUN command. If the flag is on, the AT SVC Processor recognizes it as an unconditional RUN request and returns control to the calling routine with a RUN return code without executing the original instruction. It also bypasses any chained ATs at the location under consideration. If the flag is not on, this module checks its return code from Language Control. If any return code but RUN (RC=0) occurs, this module returns control to its calling routine, passing the return code from Language Control. If the return code from Language Control is RUN, this module relocates the AT and checks for chaining, as already described.

When the last ACB in a chain has been processed, the overlaid instruction (stored in the ACB) is inspected to determine if the operation code is LPSW or any of the branch instructions. If it is, and no error is likely, the operation is simulated. If an error seems likely to occur if the instruction is executed, this module removes the AT SVC under consideration by a call to the REMOVE Command Processor (CEHKR/CZHYR). On return of control, it restores the current PSW IC, turns off the AT mode switch, and returns to the caller.

AT SVC Return Procedures: When AT SVC processing is complete, this module locates and examines an AT Relocation (Execution) Area, which in RSS is a special resident area, and in VSS is a special CSECT with user read/write attributes (see Figure 12). If a slot in the AT Relocation Area is available this module locks the slot and begins the return procedures. The locking guards against the situation which exists if a program check occurs when the original overlaid instruction is executed.

If a slot in the AT Relocation Area is not available, this module calls the Mes-

Real Storage

Lock Byte	Original Instruction	Return SVC	Instruction Length	Original AT Location	Pointer to Lock Byte	Unused
2 bytes	2-6 bytes	2 bytes	2 bytes	4 bytes	4 bytes	0-4 bytes

Virtual Storage

Lock Byte	Original Instruction	Return SVC	Instruction Length	Original AT Location	Pointer to Lock Byte	Relocation Area Identifier	Unused
2 bytes	2-6 bytes	2 bytes	2 bytes	4 bytes	4 bytes	8 bytes	0-4 bytes

Figure 12. The AT Relocation Area

sage Writer routine (CEHCM/CZHNM) to indicate that the AT SVC Relocation Area has been filled and, on return of control, to call Language Control to prompt the terminal for further instructions.

In the return procedures, this module moves the original instruction to the AT Relocation Area. SVC 67, 68, or 84 (RSS RM, RSS VM, or VSS, respectively) is placed in the two bytes immediately following the instruction, and portions of the ACB is also saved in this area. The current PSW instruction address is changed to point to the new location of the original instruction, and this routine turns off the AT mode switch and returns to its calling routine with return code zero. The AT relocation area is shown in Figure 12 and is formatted as follows:

RM: CEHJAB--8 areas, each 20 bytes

VM: CZHZAB--8 areas, each 28 bytes

```
AREA+0  2 bytes lock byte
    +2  2-6 bytes of instruction
  contiguous to this:
     2 bytes of return svc
     2 bytes instruction length
     4 bytes AT location field
     4 bytes pointer to lock byte to
       be reset

  in real storage:
       0-4 bytes unused (dependent on
       instruction length)

  in virtual storage only:
     8 bytes of character identifica-
       tion in the form
       CEHJAVAT if RSS-implanted;
       CZHZAVAT if VSS-implanted.
       0-4 bytes unused (dependent on
       instruction length)
```

When this routine is entered because SVC 67, 68, or 84 was executed, it checks the validity of the VM AT ID and adjusts the current PSW instruction address to point to the instruction immediately following the AT SVC (that is, following the overlaid instruction). This routine then exits to its calling routine with a return code of zero.

Language Control (CEHLC/CZHXC)

Chart 38

Language Control requests input from a system programmer's terminal or other input device and directs the processing of the source string supplied by the I/O area.

ENTRIES: Language Control is called at CEHLCA/CZHXCA by:

RSS External Interrupt Processor (CEHAE)

RSS SVC Service Processors (CEHDR)

VSS Activate Interrupt Processor (CZHNV)

VSS External Interrupt Processor (CZHNE)

RSS/VSS AT SVC Processor (CEHJA/CZHZA)

The calling routine must correctly initialize the Input Device Table (CHALCR).

MODULES CALLED: Language Control calls:

Module Name and ID	Reason for Call
I/O Control (CEHEA/CZHSA)	1. If input device is terminal, to invite input by writing a $ sign on the terminal. 2. To read the input device.
Source to Polish (CEHLP/CZHXP)	To create a polish string from the source string.
Scan Control (CEHLS/CZHXS)	To process the polish string.

Message Writer Routine (CEHCM/CZHNM)	To print error messages.

EXITS: Exit is to the calling routine when this module receives indication of a RUN or DISCONNECT command from Scan Control. It indicates this condition to the calling routine by return codes of zero and four, respectively. Language Control also exits on recognition of a "void" command with a return code of eight.

OPERATION: If the input data is not already in Language Control's input buffer, and if the input device is a terminal, this module invites input by causing a $ to be printed at the terminal. On return of control from I/O Control, or if the input device is not a terminal, Language Control causes the input device to be read by a call to I/O Control. The input device address is in the first word of the Input Device Table, and the device may be the system programmer's terminal, a card reader, or a tape drive. Language Control requests a read of 256 characters (which will include an end-of-transmission character) into the Language input buffer (CEHLCT/CZHXCT).

If, on return of control from I/O Control, the residual count is still 256, this module recognizes the void command, and exits with a return code of eight. If the state is RSS, this module ignores the situation and loops back to invite input.

If an error code is returned from I/O Control, this module calls either RSS or VSS Message Writer to write out an error message. Language Control recognizes a return code of 16 from I/O Control as an "end of file" indication. In call mode, in this case, Language Control performs the end function.

Note that if TSSS is in call mode, and an asynchronous interruption is received, Language Control performs the end function and causes a diagnostic message to be printed at the terminal. On return of control from the message routine, this module prompts the terminal for input.

After the input has been read, Language Control calls Source to Polish to create a polish string from the source string (see Chart 39). If Source to Polish sends back an acceptable return code, Language Control links to Scan Control, which is responsible for all subsequent language processing for the input statement. Language Control does not regain control until:

1. The polish string has been processed (return code = 0).
2. A command is executed which requires Language Control to request additional input (return code = 0).
3. A RUN or DISCONNECT command is executed (return code = 8 or 12).
4. A major error occurs (return code = 4).

If any but the third case causes return, this module loops back through its entire logic as previously described (see Chart 38), although, in case of an error, this module first causes a diagnostic message to be written at the terminal. If return is caused by a RUN or DISCONNECT command, this module translates the return code into one that is acceptable to the Environment module that called it (zero or four), and returns to that Environment routine.

Note that VSS Language Control is responsible for providing a copy of the Real Core Symbol Table (CHARST) in the IVM of each task in which VSS is activated. The VSS routines Symbol Resolution (CZHWS), Address to Symbol Resolution (CZHMA), and Storage Map Format (CZHMM) refer to this table and must be able to address it.

Using the ISASDS pointer from the Interrupt Storage Area, VSS Language Control refers to the Shared Data Set Table (CHASDS) and obtains a pointer to the hash table (SDSHAS). By using the hash value of CHBRST as a displacement into the hash chain, this module finds the correct Shared Data Set Member (CHASDM) from which it obtains the address of the Shared Page Table (SDMSPT).

This module then inserts the segment number in the high-order half of register 1 and the shared page table address in the low-order half of register 1, and issues an SVC 238 to connect the segment to the Shared Page Table.

On return of control from SVC 238, this module appends the relative page number (SDMFSP) to form a virtual storage address consisting of the segment and page with displacement of zero.

VSS Language Control saves the address, which is now the address of the Real Core Symbol Table, in a language save area (CZHXCL).

Source to Polish (CEHLP/CZHXP)

Chart 39

Source to Polish scans left to right and selects, one by one, items from a source string and classifies each as either an

operator or an operand. It forms a polish string from these operators and operands.

ENTRIES: This module is called at CEHLPA/CZHXPA by Language Control (CEHLC/CZHXC). It expects a pointer to the input string as an input parameter.

MODULES CALLED: None.

EXITS: Exit is to the calling routine.

OPERATION: The construction of a polish string from an input source string involves use of three work strings:

1. An operator string
2. A name string
3. A symbol string (for symbols and literals)

Note: All tables referred to below are found in Appendix D.

Operator String: An operator string is a "pushdown stack" which can contain operators or keywords, each represented as a halfword code. This halfword is composed of an identifier and a weight factor, as determined from the Tables of Codes (see Appendix D, Table of Codes). The operator string is initialized with an EOB character, as this character is the "heaviest" of the acceptable characters. The comparative weight of the last item in the operator string and the character under examination from the source string determines whether (1) the last item will remain in the operator string and be followed by the source character, or (2) the last item will be copied into the name string and be replaced by the source character.

Name String: The name string eventually becomes the polish string. The name string contains operators and keywords (represented as halfword codes) after they have been forced off the operator string. This string also contains halfword pointers to the symbols and literals in the symbol string. The first byte of the pointer halfword is always hexadecimal 80 (X'80') to indicate that the halfword is a symbol pointer; the second byte is the displacement from the beginning of the symbol string.

Symbol String: Each symbol string entry contains the length of the symbol (byte 0) and the type of symbol (byte 1), obtained from the Table of Codes. The remainder of the entry contains the symbol as it appeared in the source string in EBCDIC. Character literals have been edited by this time.

Work Strings to Polish String: In order to construct the polish string, this module:

1. Examines an item in the source string and classifies it, updating the pointer to the next item.
2. Fetches the appropriate action code from the matrix by multiplication and calls an internal subroutine to perform the function indicated by the action code (see Table of Action Codes).
3. When the end-of-command or end-of-block characters are reached, makes the symbol string follow the name string of halfword codes and pointers, in order to form what is now a polish string.

This module recognizes each item in the source string as the result of a left-to-right scan which stops at a delimiter. The following are delimiters that also are operators and are encoded immediately:

```
+ / * = < : > &  %
```

The other delimiters are:

```
. - ( ) $ b , ' ;   (X'26')
```

Some delimiters must be judged in context (for example, a $ following a blank may be the first character of a system symbol, as in $PSW, or of an operator). Source to Polish recognizes a keyword through comparison with a list of acceptable keywords. This module checks for a valid keyword syntactically -- the first item following a semicolon (end of command) must be a keyword. If this module does not find a match in the keyword list it indicates a syntactical error to Language Control.

A period and a left parenthesis immediately following a symbol or literal is regarded as an offset operator by Source to Polish. When a period that is not followed by a left parenthesis appears in the source string, it is considered in context, and, if it follows $RM or $VM, is regarded by Source to Polish as the explicit qualification operator. It is encoded as X'1B'. A sample source string would appear as follows:

```
SET $VM(13).SYMBOL=0
```

If the period follows the letters $PATCH or $AT in the source string, it indicates an operator representing an entry in the Patch Table or an entry in the AT Table. Thus, while $PATCH and $AT are system symbols, "$PATCH." and "$AT." are encoded as X'1C' and X'1D', respectively.

If an item in the source string is not an operator, a system symbol, or a keyword, Source to Polish tentatively classifies it as:

- A symbol, if it is of eight or less alphameric characters with an alphabetic first character.
- An integer literal, if the first character of the source item is numeric.
- A literal with the proper type attribute, if it is enclosed in quotes and preceded by A, C, L, or X.

The following paragraphs describe the logic of the processing which follows classification and results in the creation of a polish string from the following source string. In a polish string (X'26') equals EOB.

```
IF ABLE > 5 DISPLAY ABLE; RUN L'1000' Δ
```

The first item, "IF", is classified as an operator and, through reference to the Table of Codes, the module assigns it a weight of X'0C' and a type of X'10'. As an operator, it is compared with the last entry in the operator string ("pushdown stack"). The operator string is initialized with an X'0D0E', signifying an end-of-block character (represented by the "delta" in the source string). The EOB is of greater weight than the IF operator, so the IF is entered in the operator string:

Operator String

0D0E	0C10

Source to Polish examines the next item in the source string and, finding it a symbol (ABLE), enters a pointer to the symbol in the name string; it enters the symbol itself in the symbol string:

Name String

	8000

The hexadecimal digit X'80' signifies that the entry is a symbol, and the final X'00' signifies that it has a zero displacement from the origin of the symbol string. The first fullword of the name string is left unused at this time. When the name string is completed this fullword contains a length factor.

Symbol String

0403	C1C2	D3C5
1 t	A B	L E

The X'04' in the symbol string is a length factor for the immediately following symbol; the X'03' is a type factor (in this case, external or SP symbol). When the next symbol pointer is to be added to the name string, Source to Polish adds the two-byte length of the pointer to the length factor of the last symbol in the symbol string to arrive at the displacement factor for the next symbol's pointer.

Processing continues in the same fashion until Source to Polish encounters the DISPLAY keyword. This keyword is assigned a weight factor of X'0C', and a type factor of X'05' (see Table of Codes). This module compares this weight against the last entry in the operator string. The operator string at this point appears as follows:

Operator String

0D0E	0C10	0608
EOB	IF	>

The DISPLAY keyword's weight is greater than that of the "greater than" character. This module removes the "greater than" indication from the operator string and places it in the name string. It compares DISPLAY against the new "last" entry in the operator string. Although DISPLAY has the same weight as the IF operator, the DISPLAY occurred later in the source string, and it forces IF off the operator string onto the name string. As the DISPLAY keyword's weight is not greater than or equal to the weight of the EOB character, this module copies DISPLAY into the operator string, overlaying the position held by IF. After Source to Polish resolves the next item (the symbol ABLE), the operator string, the name string, and the symbol string appear as shown in Figure 13.

OPERATOR STRING:

0D0E	0C05
EOB	DISPLAY

NAME STRING:

8000	8006	0608	0C10	8009	
PTR	PTR	>	IF	PTR	

SYMBOL STRING:

0403	C1C2	D3C5	0105	F5 04	03	C1	C2	D3	C5
1 t	A B	L E	1 t	5	1	t	A	B	L E

Figure 13. The operator, name and symbol strings in polish construction

The next character in the sequence is a semicolon (;), which is an end-of-command character and has the weight (X'0D0F') to force DISPLAY off the operator string onto the name string. As an "end" character, it causes this module to hook-up the symbol string to the name string, after it has copied the end-of-command character into the name string.

Source to Polish counts the number of bytes in the symbol string. In this case (15 bytes) this module attaches an unused byte (it will contain zeros) to the end of the symbol string. (The count must be even so that the next polish string created from the input source string will begin on a halfword boundary.) This module also counts the number of halfwords in the symbol string and places this number in a fullword at the end of the name string. It places the total length of the name string, including the last fullword, in the vacant fullword at the beginning of the name string. The completed name and symbol strings, when this module attaches them to form a polish string, are shown in Figure 14.

After forming this polish string, Source to Polish determines if the end character is an end-of-command (semicolon) or an end-of-block (EOB). If it is an end-of-command, more input in the source string exists. Source to Polish builds another polish string for this input immediately following the end of the existing polish string. It leaves the first fullword following the existing polish string vacant, creating the new name string after this fullword. This module uses the same working storage to construct the new operator and symbol strings that it used for the old ones. In the given input string, an end-of-block character terminates the second polish string. After combining the name and symbol strings, Source to Polish returns control to Language Control, passing back a pointer to the origin of the first polish string. The completed polish strings appear in Figure 15.

The AT keyword is given special treatment during Source to Polish operation. Given the input string:

```
AT X,Y,Z DISPLAY A;STOP
```

The following is the equivalent of the finished polish string:

X	Y	Z	DISPLAY A;STOP	AT	

After placing the AT command on the operator string and representing the symbols X, Y, and Z on the name string, Source to Polish recognizes the next item as the keyword DISPLAY (the beginning of the AT command's input string) and takes a special path in which it considers the remainder of the input string (DISPLAY A;STOP) as a single character constant. The entry in the name string is given the length factor of the remainder of the source string and the type factor of a character literal (X'08'). Scan Control will, upon recognizing this entry in the polish string, construct an SCB containing a pointer to the character literal designated.

As a result, after Scan Control forms its parameter list of SCBs and links to the AT Command Processor, the SCB at the end of the list will represent a pointer to the AT source string, and all SCBs before the last one will represent the locations at which the implantation is to be made.

Scan Control (CEHLS/CZHXS)

Chart 40

Scan Control scans a polish string, passing control and parameters to the appropriate service routine upon the occurrence of each symbol, literal, keyword, or operator.

ENTRIES: Scan Control is called at CEHLSA/CZHXSA by Language Control (CEHLC/CZHXC). Input parameters are (1) a pointer to the origin of the polish string and (2) a pointer to the source string. It must also have a list of the available Symbol Control Blocks (SCBs).

MODULES CALLED: Scan Control calls:

Module Name and ID	Parameters Passed
Symbol Resolution (CEHMS, CZHWS)	A pointer to an SCB containing the symbol under consideration.

Polish String:

0010	8000	8006	0608	0C10	8009	0C05	0D0F	0012	0403	C1C2	D3C5	0105	F504	03C1	C2D3	C500
cnt	ptr	ptr	>	IF	ptr	DIS-PLAY	;	cnt	1	t	A	B	L	E	1 t 5	1 t A B L E Fill

Figure 14. The polish string partially constructed

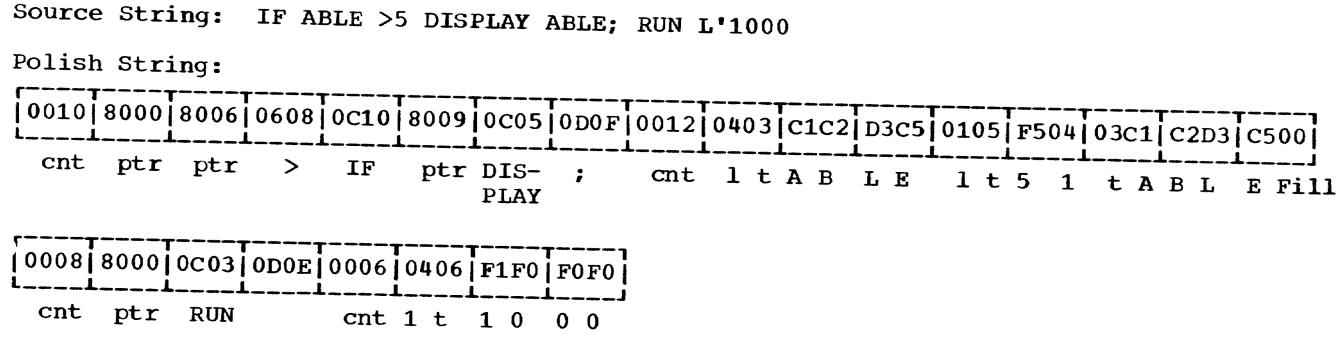


Figure 15. The completed polish string

Literal Resolution (CEHLL/CZHXL)	A pointer to the polish string entry which contains the literal and its length and type attributes and a pointer to a skeleton SCB.
Operator Functions (CEHLA/CZHKA)	A pointer to the parameter list containing the input operator code.
The Keyword Execution Subroutines	A pointer to the parameter list.
Message Writer Routine (CEHNM, CZHNM)	A message control word.

EXITS: Exit is to Language Control. This exit is taken when Scan Control encounters (1) an error that makes further processing inappropriate, (2) a RUN or DISCONNECT command, (3) the end of the polish string (EOB), or (4) a command requesting additional SP input.

OPERATION: The polish string, which Scan Control examines, is arranged so that Scan Control encounters the symbols and literals of a command statement before it encounters the operators and keywords, and generally the operators before the keywords. As Scan Control processes the symbols and literals in the polish string, it builds a parameter list for subsequent use by the operator functions and the keyword execution subroutines.

Scan Control examines each item in the polish string. If the item is a symbol or literal, Scan Control calls Symbol or Literal Resolution, respectively, for classification. It passes to Symbol Resolution a pointer to the Symbol Control Block (SCB) in which it has placed the symbol and its qualification.

If the item is a literal, Scan Control passes to Literal Resolution a pointer in register 1 to a special parameter list that consists of a pointer to the polish string entry to be resolved and a pointer to the skeleton SCB, already containing qualification, into which the literal is to be resolved. This parameter list is shown in Figure 16.

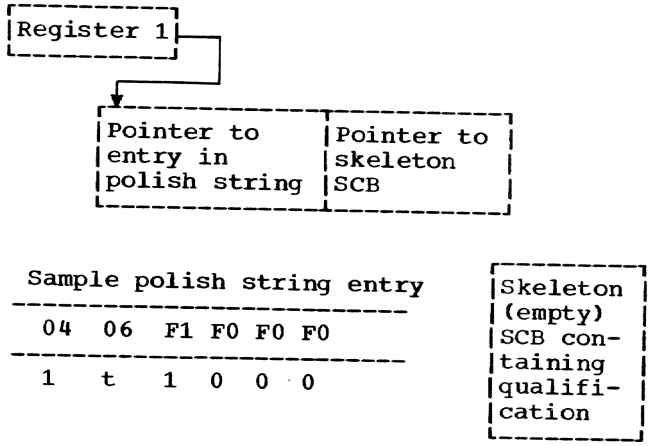


Figure 16. The scan control parameter list for literal resolution

When Scan Control encounters a keyword or operator, it links to the corresponding routine, passing a pointer to the parameter list as shown in Figure 17. The called routine makes use of the current, filled-in SCB(s) to perform the requested function.

The A in Figure 17 represents the pointer passed to the Keyword Execution Subroutines. B represents the pointer passed to the Operator Functions.

The parameter list may be as long as 14 fullwords (that is, there may be as many as 12 SCBs). The extra fullword containing the operator code is used as an index to a branching table within the Operator Functions module (see Charts BN, BO, BP, BQ,

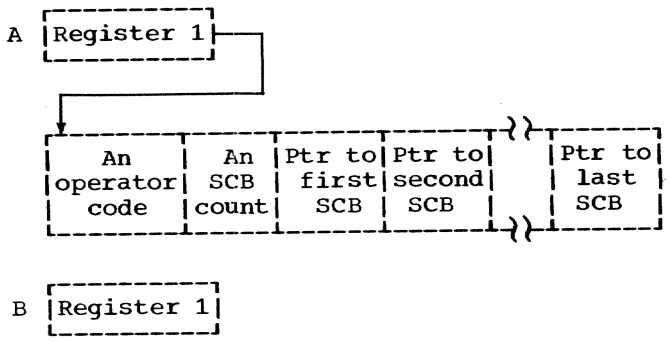


Figure 17. The scan control parameter list for operator functions and keyword execution

and BR) to determine which routine performs the requested function.

If Scan Control recognizes an explicit qualification operator (X'1B'), it links to Operator Functions (CEHLA/CZHYA), which checks for $RM or $VM in the input SCB. The explicit qualification operator is always placed in the polish string immediately before the symbol to be qualified. If the qualification in the current SCB on the parameter list is valid, this SCB on the parameter list is reinitialized to contain only this explicit qualification.

On return of control, Scan Control retains this same SCB as input to Symbol Resolution (CEHMS-CZHWS), in effect performing all qualification procedures itself. Symbol Resolution (or Literal Resolution) does not alter the incoming qualification of the SCB when it resolves the input symbol (or literal).

Scan Control expects one of five return codes from its called routines: zero, four, eight, twelve, or sixteen.

Zero indicates that processing has been successfully completed. Scan Control continues the scan of the polish string.

Four, when accompanied by a Message Control Word in register 0, indicates that the called routine encountered a serious error. Scan Control returns control to Language Control with a return code of four and passes the Message Control Word for use as input to the Message Writer routine.

Four without an accompanying Message Control Word indicates that a STOP, CALL, END, or AT command has been processed, and indicates that termination of the scan should be requested. An IF operator which results in a "false" condition also returns this code. Scan Control returns control to Language Control with a return code of zero to request that Language Control invite additional input from the SP, or, if in AT mode, that Language Control execute the "run" procedures.

Eight indicates that a RUN command has been executed. Scan Control returns control to Language Control to indicate this.

Twelve (X'0C') indicates that a DISCONNECT command has been executed. Scan Control returns control to Language Control to indicate this.

Sixteen (X'10') indicates that the called routine encountered a minor error and has constructed a Message Control Word in register 0. Scan Control calls the Message Writer routine to post this error, and on return of control continues the scan.

Scan Control also returns control to Language Control when it reaches the end of the entire polish string (end-of-block character). Scan Control does little error checking, relying on its called routines for detailed error checking.

RSS Symbol Resolution (CEHMS)

Chart 41

This module resolves and classifies (1) system-programmer defined symbols (SP symbols), (2) system symbols, and (3) external symbols. This module is also called by RSS Literal Resolution to assist in resolving address constants.

ENTRIES: This module is called at CEHMSA by Scan Control (CEHLS) and Literal Resolution (CEHLL). The input parameter is the address of a skeleton SCB containing the symbol under consideration and the qualification given it in the polish string (see Figure 18). RSS AT SVC Processor (CEHJA) calls this module to locate the user's Read/Write CSECT to implant a return SVC when the original AT SVC was in virtual storage.

MODULES CALLED: This module calls VM Access (CEHCB) when searching the Task Dictionary.

EXITS: Exit is to the calling routine.

OPERATION: This module first checks whether or not the first character in the symbol is a dollar sign, indicating a system symbol. If it isn't, the MSP-defined symbol table is searched. If the symbol is not found there, it is looked for in real or virtual storage. If it cannot be found, the "undefined" flag is set on, and control is returned to the calling routine.

System Symbol	Meaning (Qualification)
$R	General registers
$C	Control registers
$E	Floating point registers
$CAW	Channel Address Word
$CSW	Channel Status Word
$DOUT	Specify output device for DUMP
$PSW	Current PSW
$PPSW	Program interruption PSW
$SPSW	SVC interruption PSW
$XPSW	External interruption PSW
$IPSW	I/O interruption old PSW
$MPSW	Machine Check interruption old PSW
$AT	Dynamic statements in AT table for current SP (RM or VM)
$PATCH	Patcnes; analogous to $AT (RM or VM)
$IO	Output device specification
$VAM	Output device specification
$MAP	Specifies a TSS/360 storage map of external symbols with their hexadecimal addresses
$RM(n)	Real storage (RM) qualification (dummy SCB)
$VM(n)	Virtual storage (VM) qualification (dummy SCB)
$TSKID	Current task ID
$DHDR	Label for output of DUMP command
$STATUS	Primary system status indicators
$TASK	Primary task status indicators

Figure 18. RSS System Symbol Table

If the symbol is $AT, $PATCH, or $MAP, the Symbol Control Block (SCB) is copied without the qualification entry, and control is returned to the calling routine. If it is a variable system symbol ($R, $C, $E, $PSW, $PPSW, $IPSW, $XPSW, $SPSW, or $MPSW), the SCB is copied without the qualification entry, and the base for the qualification is resolved using the CPU ID. If the CPU ID is not valid, an error code of four is returned. If the symbol is any other system symbol or an MSP-defined symbol, control is returned to the calling routine with the normal return code of zero.

If the symbol is an external symbol in real storage, its address, type (hex), and class (external), are stored in the SCB, and control is returned to the calling routine. If the symbol is in virtual storage, it is located using a special paging subroutine included in this module. Its type and class are stored in the SCB, and control is returned to the calling routine.

VSS Symbol Resolution (CZHWS)

Chart 42

This module classifies and resolves system symbols, SP symbols, and external symbols for VSS. It is also called by Literal Resolution to resolve address constants for VSS.

ENTRIES: This module is called at CZHWSA by Scan Control (CZHXS) and Literal Resolution (CZHXL). It requires a pointer to an SCB containing the symbol as an input parameter (see Figure 19).

MODULES CALLED: This module calls VSS VM Access (CZHPB) while searching the Task Dictionary.

EXITS: Exit is to the calling routine.

OPERATION: This modules checks for type of symbol in the same way as RSS Symbol Resolution (CEHMS). If a $ appears in the first byte of the symbol field in the input SCB, the symbol is classified as a system symbol. This module searches the System Symbol Table it maintains (see Figure 19). If the symbol is found, this module copies the permanent SCB into the input SCB and returns. If not, this module sets error parameters before returning control.

If the symbol is not a system symbol, this module determines if the symbol has a private or global qualification and searches the appropriate SP symbol table. This table may be the TSP's private symbol table or the Global Symbol Table to which all TSPs have access. This routine copies the permanent SCB into the input SCB and returns control if it finds a match.

If the symbol is not found, this module determines if the symbol has a real or virtual storage qualification. If the

System Symbol	Meaning (Qualification)
$R	General registers
$C	Control registers
$E	Floating point registers
$DOUT	Specify output device for DUMP
$PSW	Current PSW
$SPSW	SVC interruption old PSW
$XPSW	External interruption old PSW
$IPSW	I/O interruption old PSW
$APSW	Asynchronous interruption old VPSW
$TPSW	Time interruption old VPSW
$PPSW1	Recoverable Data Set Paging VPSW
$PPSW2	Program interruption old VPSW
$AT	Dynamic statements in AT table for current SP (RM or VM)
$PATCH	Patches; analogous to $AT (RM or VM)
$IO	Output device specification
$VAM	Output device specification
$MAP	A TSS/360 storage map of external symbols with their hexadecimal addresses
$RM(n)	Real storage (RM) qualification (dummy SCB)
$VM(n)	Virtual storage (VM) qualification (dummy SCB)
$TSKID	Current task ID
$DHDR	Label for output of DUMP command
$TASK	Primary task status indicators

Figure 19. VSS System Symbol Table

qualification is real, this module searches the Real Core Symbol Table (CHARST) in Initial Virtual Memory. (This table is obtained for VSS via the CNSEG macro instruction, SVC 238. See "Language Control", operation description.) If the symbol has a VM qualification, this module searches the Task Dictionary Table (CHATDY).

If the symbol is not found, the input SCB is flagged as undefined, and Symbol Resolution returns to its calling routine. If it is found, it puts the standard attributes into the input SCB before it returns control.

Literal Resolution (CEHLL/CZHXL)

Chart 43

Literal Resolution examines an entry in the polish string which Scan Control has tentatively recognized as a literal. If this entry meets all the requirements for a literal, this module builds an SCB for it.

ENTRIES: Literal Resolution is a subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHLLA/CZHXLA. Input parameters are (1) a pointer to the entry in the polish string and (2) a pointer to the SCB list, showing the location of the next available working SCB.

MODULES CALLED: Literal Resolution calls Symbol Resolution (CEHMS/CZHWS) if it encounters an address constant literal. As a parameter it passes a pointer to the SCB containing the literal.

EXITS: Exit is to Scan Control.

OPERATION: The entry in the polish string to which Literal Resolution is passed a pointer consists of:

1. A byte count of the entire entry (first byte).
2. The type of operand (second byte).
3. The literal entry, in EBCDIC (as it appeared in the source string).

Literal Resolution uses the type factor to pick out the internal routine which checks the entry. If the literal is valid, this module constructs an SCB for it, which must contain a type (character, hexadecimal, etc.), a literal classification, length and size attributes, a base, and a pointer. If the literal is an address constant, this routine calls Symbol Resolution, which is then responsible for completing most of these fields in the SCB.

For each type of literal, some special processing is required. If the literal type is character, it was edited during conversion from source to polish, and this routine gives the literal the length and size of its polish string. A location literal is permitted a field of no more than six (RM) or eight (VM) hexadecimal digits.

This module translates from EBCDIC and packs a hexadecimal literal, before building an SCB for it, by using an internal, closed, serially reusable subroutine called the Editor. This module translates and packs an integer literal and also converts it to binary before building an SCB for it.

Normal output from Literal Resolution is a new SCB in the SCB list, to which this routine passes a pointer when it returns control to Scan Control. Detection of any condition that reveals an invalid literal, such as an invalid type or invalid byte count, causes Literal Resolution to execute the error return procedures. This module indicates a minor error to Scan Control (return code = 16) if the value of an integer literal exceeds 2, 147, 483, 646 (the packed integer, converted to binary, exceeds four bytes), or if the type is hexadecimal and the string contains other than a hexadecimal digit.

With the exception of a location literal, literal qualification in RSS is RM; in VSS it is VM. Location literals may be assigned other qualifications.

Operator Functions (CEHLA/CZHXA)

Charts 44 to 48

The operator functions are many small subroutines which accept parameters and operate on them. The operation performed may be arithmetic, subscripting, comparison, or any of the other operations allowed in the TSSS Command Language (defined below under "Operation"). The result of the operation is placed in an SCB or in a work area pointed to from an SCB.

ENTRIES: The module is a subroutine of Scan Control, called by it at CEHLAA/CZHXAA. Entry to the individual operator subroutines is via a branching table. The input parameter is a pointer to a parameter list, containing the operator code as an index to the branching table.

MODULES CALLED: The Operator routines call RSS Real Core Access (CEHCA), RSS VM Access (CEHCB), VSS Real Core Access (CZHPA), and VSS VM Access (CZHPB) when they perform an operation on a data field.

In addition, the $ID routine calls Address to Symbol Resolution (CEHMA/CZHWA) to locate a symbol in real or virtual storage, and the $VAM routine calls the External Page Location Address Translator (CEHBT/CZHRT) to resolve a physical data location in the CHHR format.

EXITS: Exit is to Scan Control.

OPERATION: Each operator has a unique number assigned to it and placed with it in the polish string (see the Table of Codes in Appendix D). When Scan Control calls Operator Functions, it passes a pointer to the parameter list, the first word of which contains this operator code. When multiplied by four this code serves as an index to the branch table, to permit entry to the requested subroutine.

After the operation is performed (see Charts 44 to 48 inclusive), and the result placed in either an SCB or work area, the SCB count is, in most cases, reduced by one. The subroutine returns control to Scan Control with a pointer to the SCB list, containing a pointer to the result of the operation. (The error checks for each operator are shown on the logic charts.)

The acceptable arithmetic operators (see Chart 44) are addition (+), subtraction (-), unary minus (-), multiplication (*), and division (/). These operations are performed with fixed point arithmetic instructions.

The relational operators "greater than," "equal to," and "less than" result in a comparison (see Chart 45). A logical indication of "true" (X'FF') or "false" (X'00') is returned and also stored. It should be noted that when the equal sign (=) is used in a command string with certain keywords (such as SET), it is treated as a delimiter, recognized by Source to Polish as such, and never reaches Operator Functions.

The Boolean operator "logical AND" results in ANDing two parameters (one each from two SCBs); the "logical OR" results in ORing two parameters; and the "logical NOT" results in reversing the bits of a parameter with the Exclusive OR instruction. (See Chart 45.)

The IF command is an operator function that determines if the conditions established by the system programmer have been met. (See Chart 48.) It tests the result field of a previous logical operation. If it returns a "true" indication (return code = 0) Scan Control will continue scanning the polish string. If the indication is "false" (return code = 4, no message specified), the conditions have not been met, the remainder of the polish string is ignored, and further input is solicited.

The "%" operator provides indirect addressing capabilities by loading the specified data field into the base address field of an SCB. (See Chart 47.)

Subscript (left parenthesis immediately following a symbol or literal), range (:), or offset (a period and a left parenthesis

immediately following a symbol or literal) result in the manipulation of a data field within an SCB. The subscript operator (see Chart 46) handles $IO and $VAM symbols. $VAM calls the External RSS/VSS Page Location Address Translator (CEHBT/CZHRT). Using information supplied by the SCBs for these symbols, the operator fills an SCB that defines a data field on an I/O device.

Special cases of offset (for example, a period not followed by a parenthesis) include explicit qualification, the Patch Table Entry Operator, and the AT Table Entry Operator. Explicit qualification insures that either $RM or $VM is specified in an SCB later to be used for symbol or literal resolution.

The Patch and AT Table Entry Operators accept as input a resolved symbol after all other operations on it have taken place. They make up $PATCH and $AT SCBs, respectively, with the pointer attribute equal to the base address of the incoming SCB and a zero base address.

The $B, $P, $T, $S, and $L symbols (Chart 48) are classified externally as system symbols. Internally they are classified as attribute operators. These operator functions build SCBs that define the base address, pointer, type, size, or length attributes, respectively, for an SP-defined symbol.

The $ID operator calls Address to Symbol Resolution, which searches the appropriate table to locate the external symbol nearest to (less than or equal to) a hexadecimal address supplied as input. The found alphameric symbol and its address is placed in a buffer and pointed to by an SCB. The buffer is 12 bytes long. (See Chart 48.)

Address to Symbol Resolution (CEHMA/CZHWA)

Chart 49

This module locates the symbol nearest to (with an address less than or equal to) the address of the input operand in real or virtual storage.

ENTRIES: This module is called by the $ID routine of the Operator Functions (CEHLA/CZHKA) at CEHMAA/CZHWAA. The input parameter is a pointer to an SCB containing the address of the input operand.

MODULES CALLED: This module calls RSS and VSS VM Access (CEHCB, CZHPB) if the qualification is VM, in order to search the Task Dictionary Tables (TDY).

EXITS: Exit is to the calling routine.

OPERATION: This module determines if the qualification of the input address is real storage (RM) or virtual storage (VM).

If the qualification is RM, this module searches the Real Core Symbol Table (CHARST) for the address closest to the input address. The Real Core Symbol Table is loaded into real storage by RSS and resides in Initial Virtual Memory of a task for VSS. It is a table of external symbols and their addresses, sorted alphabetically in ascending order.

If the qualification is VM, this module searches the Storage Map Table (CHAMAP) within the Task Dictionary (TDY). This table is in ascending order of virtual storage addresses.

If a match is found, the symbol name and its address are placed in a buffer pointed to by an SCB, and this module exits. If no match is found, or if the input address is less than the addresses of the beginning entries in the respective tables, this module executes the error return procedures.

AT Command Processor (CEHKA/CZHYA)

Charts 50,51

The AT command allows the system programmer to specify a location in TSS/360 running code and a command language statement that is to be executed immediately before the instruction at the designated location. This module provides this capability by overlaying the first two bytes of the instruction with an AT SVC. This module also builds AT Control Blocks and makes entries in the SP's AT Table and the Shared Global AT Table.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKAA/CZHYAA. Parameters expected are a pointer to a parameter list containing an SCB count and pointers to the SCBs, the last of which contains the length of the command string and a pointer to it.

MODULES CALLED: In order to refer to the desired location, the RSS copy of this module (CEHKA) calls one of the following:

Module Name and ID	Reason for Call
RSS Real Core Access (CEHCA)	To locate a page of real storage.
RSS or VSS Virtual Storage Access (CEHCB,CZHPB)	To locate a page of virtual storage.

EXITS: Exit is to Scan Control.

OPERATION: In general, the explanation of an AT SVC is as follows. This module tests the AT Table to determine if there is enough space for the AT Control Block (ACB) and the source string for the current AT. (Each AT Table is one page in length. See Appendix E for a description of the AT Table.) If all conditions are valid, this module locates the page containing the location specified by the input SCB, using a storage access routine.

This module then checks for a valid operation code in the instruction to be overlaid. It does not overlay an SVC, Execute, or Diagnose instruction. If the qualification is VM, it will not overlay a privileged instruction. A check is then made to see if the address of the instruction is odd. If the operation code is valid, this module saves the original instruction in an ACB and overlays the instruction location with the SVC code. This module locates the source statement with a pointer in the last SCB on the parameter list. It saves the source statement at the end of the AT Table (CHAATB), and sets a pointer to the statement in the ACB. The ACB is constructed in the next available 28 bytes at the beginning of the AT Table. (The format of the AT Table is shown in Appendix E.) The ACB contains a unique 4-byte AT ID. The AT ID for either RSS or VSS in shared virtual storage is composed of the shared segment number in the high-order 12 bits, with the location specified in the low order 20 bits. Bit 7 of the lowest order byte is set to 1 to indicate that it is shared.

If an SVC instruction is found at the location specified by an AT command operand, this module determines if the SVC represents an AT command by searching the appropriate AT Table. If a match is found, this module activates the chaining process, unless the AT SVC was implanted by a different SP. In this case, the AT command is rejected.

The specific SVC code to be implanted is determined by the receiving location and the mode of TSSS. AT SVC codes implanted by RSS are 69 (real storage), 71 (private virtual storage), and 72 (shared virtual storage). The corresponding codes implanted by VSS are 69, 80, and 85.

Before this module can implant an AT SVC, it must determine the qualification of the input SCB as well as the type of storage designated. An AT SVC may be either privately or globally qualified, and the ACB is marked accordingly after it is determined in which AT Table the ACB will be built. In RSS this module only refers to the MSP's AT Table, regardless of qualification or storage location. Figure 20 shows the location used in VSS.

Characteristics of AT			Action Taken
Qualification	Location	SVC Code	Location of ACB
RM	Real storage	69	MSP's AT Table
non-global	Private virtual storage	80	TSP's AT Table
global	Private virtual storage	80	TSP's AT Table
non-global	Shared virtual storage	85	Global Shared AT Table
global	Shared virtual storage	85	Global Shared AT Table

Figure 20. ATs in VSS

VSS maintains a private AT Table for each TSP, whereas there is only one Shared Global AT Table, which all TSPs refer to. All encountering tasks execute an SVC 85 implanted in shared virtual storage; if the corresponding ACB is marked non-global, only the task whose VSS implanted the SVC executes the stored command string. If the ACB is marked global, all encountering tasks execute the command string. All AT Tables have the same format and the same DSECT (CHAATB). The CSECTs for the MSP's AT Table, the TSP's AT Tables, and the Global Shared AT Table are, respectively, CHBATBR, CHBATBVA, and CHBATBVB.

VSS does not directly implant an AT SVC in real storage for a TSP. Instead, the VSS version of this module builds a command string from the input command string, and implants and remotely executes an SVC 70, requesting that RSS implant an AT in real storage for it (see "RSS SVC Service Processors"). For this reason the AT SVC code and the location of the ACB are the same as if an MSP had requested that the AT be implanted. A sample command string to accompany the SVC 70 appears as follows:

```
QUALIFY $RM(x); AT $RM.L'1000'DISPLAY ABLE
```

where x may equal 0, 1, or 2.

If a globally qualified AT SVC is found at the receiving location, and the requested AT is also globally qualified, chaining is activated, unless the identification of the current task differs from that of the task which implanted the existing AT SVC. The identification of the implanting task is recorded in the ACB for that AT. If the task IDs are different the current task's AT is rejected.

When the AT Command Processor completes its operation, it returns control with a request for additional input, which causes Scan Control to discontinue scanning the polish string (return code = 4, no message specified in register 0). The following conditions cause the AT Command Processor to ignore the current AT request and execute the error return procedures.

1. The number of operands is invalid. The working SCB list must contain at least two SCB, and the last SCB must describe an AT command string.

2. Insufficient space remains in the AT Table to process the current AT request.

3. The qualification is invalid.

4. The location for implanting the AT is invalid.

5. An SVC (one that does not represent a chainable AT), an Execute, a Diagnose operation code, or a privileged instruction in virtual storage already exists at the specified location.

DEFINE Command Processor (CEHKE/CZHYE)

Chart 52

The DEFINE Command allows the system programmer to define private symbols which designate fields or to establish private aliases for system symbols, external symbols, or other SP-defined symbols (called SP symbols). This module builds temporary SCBs to record SP symbol definitions. (The SCBs are temporary in that they exist only for the duration of a given SP's terminal session. See "Processing TSSS Input".)

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKEA/CZHYEA.

MODULES CALLED: None.

EXITS: Exit is to Scan Control.

OPERATION: The operation of the DEFINE command is logically divided into two sections: establishing on alias and independent definition. The difference between the two is that establishing an alias does not require allocation of storage, whereas an independent definition does.

The command format for independent definition is:

DEFINE A.(o,l,t,s)

where any or all of o, l, t, and s, may be omitted, causing the assumption of these attributes:

offset	(o)	= 0
length	(l)	= 1 byte
type	(t)	= hexadecimal
size	(s)	= length

Zero is always assumed for offset, since the offset must always be zero in the case of independent definition (storage is being allocated).

The symbol, A, must be eight characters or less in length, begin with an alphabetic character, and contain no blanks or special characters. This module checks the qualification of the SCB containing the symbol and searches the appropriate symbol table. If the symbol has not been entered before, a new entry (SCB) is made. In any valid case, space is allocated for the new data field in the opposite end of the appropriate table, the working SCB count is reduced by one to show that the symbol has been processed, and control passes to the calling routine with a return code of zero. Note that if the new symbol is the same as a symbol already in the SP's private symbol table, any space previously allocated to that symbol is not reused until after a DISCONNECT command is executed.

The command format for establishing an alias is:

DEFINE A=B.(o,l,t,s)

where A is the new name to be established and B is some symbol which has been previously defined. B may be an external symbol, a system symbol, or another SP symbol. The result is that the attributes of the data field named B, as specified, are given to the name A, and any explicit attributes of A are ignored. All fields of the SCB for A are overlaid by these of this copy of the SCB for B, excepting that of the symbol itself. The specification of o, 1, t, s is as described under the discussion of independent definition.

This module then searches the appropriate symbol table. If a previously defined A is found in the SP's private symbol table, its definition is replaced by the new definition for A. If no such definition is found, the SCB for A is placed in the next available slot in the appropriate table.

The Tables: When a symbol, A, is privately defined by an SP with non-global qualification, it is recorded in the SP's private Symbol Table (CHASPM). All of this SP's references to A are resolved as this data field regardless of whether other As exist

in TSS/360. Only if the SP redefines the symbol A privately will it be resolved with a definition other than the original one. Each of the symbol tables discussed here is two pages in length.

The record of all symbol definitions by an MSP is kept in the MSP's private Symbol Table (CSECT:CHBSPMR), and is subject to the restrictions already discussed.

Depending on the qualification, a TSP-defined symbol may be recorded in either the TSP's private Symbol Table (CSECT CHBSPMVA) or the Global Symbol Table (CSECT:CHBSPMVB) to which all TSPs have access. If a TSP defines a symbol, A, with non-global qualification and later gives it another definition without changing the qualification, the first definition is lost.

However, a TSP may use a single symbol, A, to define two different data fields, and preserve both definitions, if one symbol is defined under global qualification and the other is not. But if a TSP tries to define a symbol, A, with global qualification, and this symbol, A, has already been defined with global qualification, the first definition of A is lost.

The following other conditions cause this module to execute the error return procedures:

1. The number of operands (SCBs) in the working SCB list is not one or two.
2. Undefined alias symbol ("B" in the format example).
3. Insufficient space in Symbol Table; insufficient working space.

QUALIFY Command Processor (CEHKQ/CZHYQ)

Chart 53

The QUALIFY command allows the system programmer to change his state of implicit qualification (RM, VM, or global). This module maintains the Qualify Table (CHAKQD) to record those changes. (The Qualify Table appears in Appendix E.)

ENTRIES: This module is a keyword execution subroutine of Scan Control, which calls it at CEHKQA/CZHYQA.

MODULES CALLED: None.

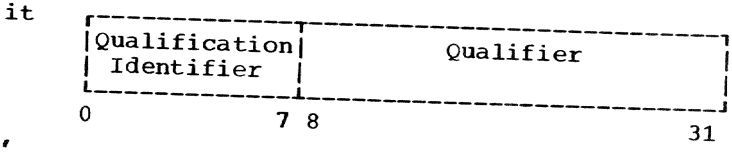
EXITS: Exit is to Scan Control.

OPERATION: The SCB parameter contains a fullword storage cell (MSWUNUS) that is the operand of the QUALIFY command.

The cell contains a number which may be from X'0000' to X'FFFF', representing a Task ID or CPU number.

In the symbol field (MSWSYMB) of the input SCB may be C'$RM' or C'$VM', representing real or virtual storage. If the symbol field is given as $VM without a Task ID (number=zeros), the qualification is global.

The following diagram depicts the format of the four-byte Qualify Table:



If RM is specified, this module updates the Qualify Table as follows:

1. If the number specified is zero (the system programmer specified no CPU number), the Qualify Table is set to all zeros.
2. If the number specified is X'0001' or X'0002', this module sets the qualifier in the Qualify Table to either X'000001' or X'000002', respectively. It sets the identifier to X'01'. Real storage (RM) qualification is thus determined by a X'00' or X'01' in the identifier field of the Qualify Table.

If VM is specified, this module updates the Qualify Table as follows:

1. If the number is all zeros, the qualification is global. This module sets the qualifier to X'00' and the identifier to X'02'.
2. If the number is from X'0001' to X'FFFF', the system programmer has designated a specific task ID for VM qualification. If the SP is a TSP, this number must match the current task ID. Any value, within the accepted range, is valid for an MSP. This module sets the qualifier to the value of the input number and the identifier to X'03'.

The following conditions cause this module to execute the error return procedures:

Storage ID is not $RM or $VM

Invalid CPU number specified

Invalid task ID given by TSP

RSS CONNECT Command Processor (CEHKW)

Chart 54

The CONNECT command allows the MSP to specify an existing TSS/360 task within which VSS is to be activated. The TSP is connected to that task at the SYSIN terminal for the task.

ENTRIES: This module is a keyword execution subroutine of Scan Control, which calls it at CEHKWA. The input parameter is a pointer to the SCB list containing only one SCB. The task number is specified as a data field in this SCB.

MODULES CALLED: This module calls the Find TSI routine (CEHCF), the RSS Interrupt Switching routine (CEHCS), and the Queue VSS Interrupt routine (CEHCQ) during the VSS activation.

EXITS: Exit is to Scan Control.

OPERATION: This module calls the Find TSI routine to locate the Task Status Index (TSI) for the task specified in the input parameter, passing a task ID in bytes two and three of register 1. On normal return this module checks the "TSP connected" flag in the TSI (TSIVT). If a TSP is not already connected to the designated task, this module links to RSS Interrupt Switching to activate VSS in the TSI. Input is a pointer to the TSI in register 1 and a code (zero) requesting activation in register 0.

When control is returned, this module sets the symbolic device address of the TSP terminal in the TSI (TSISDA) equal to the symbolic address of the SYSIN terminal (TSISIN). It links to the Queue VSS Interrupt module (CEHCQ) at CEHQA to build and queue a VSS activate interruption GQE (code 5). When control is returned, this module again calls the Queue VSS Interrupt module, but at entry point CEHCQB, to build and queue an external interruption which has an MCB attached for the subject task. The MCB contains TSP information that will be used by the VSS Activate Interrupt Processor (CZHNV) to initialize, the device tables. On return of control this module exits to the calling routine.

This routine executes the error return procedures on the following conditions:

The task number is invalid (as indicated by the Find TSI routine); the TSI cannot be found.

There is an invalid number of SCBs -- there must be exactly one SCB on the input parameter list.

The "TSP connected" flag in the TSI (TSIVT) is on.

COLLECT Command Processor (CEHKC/CZHYC)

Chart 55

The COLLECT command moves the data from one field to another and updates the SCB describing the receiving field, so that any subsequently collected data will immediately follow this data.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKCA/CZHYCA. The input parameter is a pointer to a parameter list containing two SCBs, the first of which must represent an SP symbol.

MODULES CALLED: This module calls the SET Command Processor (CEHKS/CZHYS) to move the data.

EXITS: Exit is to Scan Control.

OPERATION: If the length attribute of the second operand (Op2) is less than or equal to the remaining space described by the first operand (Op1), this module sets the length attribute of Op1 equal to that of Op2 and calls the SET Command Processor to move the data. When control is returned, this module increments the pointer attribute of the first operand in its permanent SCB by the length of the second operand, and returns control to Scan Control.

If there is not enough space, this module determines if the size attribute (total allocated space) of Op1 is greater than or equal to the length attribute of Op2. If it is, this module sets the pointer attribute of Op1 to zero and calls the SET Command Processor as above. If the length of Op1 is less than the length of Op2, this module sets the pointer attribute of Op1 to zero and the length attribute of Op1 equal to its size attribute. After setting the length attribute of Op2 equal to that of Op1 (effectively truncating the Op2 data field), this module calls SET to move the data. On return of control, this routine updates the pointer attribute for the first operand and returns control to the calling routine with a return code of 16 (minor error) in register 15 and "overflow" message parameters in register 0.

When the value of the pointer attribute of OP1 is equal to the size attribute, or is large enough that the length of OP2 is greater than the remaining space in the data field (size minus pointer), the pointer is reset to zero before the data is moved.

If the first operand represents a tape drive or a printer, the pointer remains zero during the collect operation.

This module executes the error return procedures on the following conditions:

The SCB list received as an input parameter does not contain exactly two SCBs.

Undefined symbol as the first operand.

Error return from SET Command Processor.

Op1 is not an SP symbol.

SET Command Processor (CEHKS/CZHYS)

Chart 56

This module moves the contents of a data field described by operand 2 into the location described by operand 1.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS) which calls it at CEHKSA/CZHYSA. It is also called by the COLLECT and PATCH Command Processors. The input parameter is a pointer to the parameter list containing a count field and two SCBs.

MODULES CALLED: Depending on the qualification of the operands, this routine calls the following modules to page in the required storage areas.

Module Name and ID	Qualification
I/O Control (CEHEA/CZHSA)	External
RSS Real Core Access (CEHCA)	RM (RSS)
VSS Real Core Access (CZHPA)	RM (VSS)
RSS Virtual Memory Access (CEHCB)	VM (RSS)
VSS Virtual Memory Access (CZHPB)	VM (VSS)

EXITS: Exit is to the calling routine.

OPERATION: If the input data is valid, this module checks the qualification of the operands. If an operand is either virtual or real, the data field is located or is brought into real storage by a storage access routine. If the qualification is external, this module links to I/O Control to perform the get function.

After locating the designated page, this module compares the length attributes of the two operands. If they are not equal, the operand 2 (Op2) data field is padded or truncated, depending on its length and type, before this module moves the data therein to the location specified by operand 1 (Op1). After performing the SET instruction, this module decrements the count field in the parameter list by two and returns to the calling routine.

The SET Command Processor provides a special facility when dealing with $DOUT as its first operand. If $DOUT is the first operand and the second operand is externally qualified, the SET Command Processor moves the resolved external symbol (now a symbolic device address) into the data field $DOUT. $DOUT represents a data field which is used by the DUMP/DISPLAY Command Processor as the symbolic device address of the device upon which a DUMP operation is to be performed. By setting $DOUT=$IO, the SP may specify any symbolic device address that exists in the SSDAT as the symbolic device address. (It must be the address of a printer or tape drive for execution of a DUMP command.)

By setting $IO=$IO, the SP may move data from one device to another. For instance, a dump written on tape may be transferred to the printer by setting $IO (address of the printer) equal to $IO (address of the tape drive). In this case, as SET recognizes the difference between a device and a location specification for Op1, it moves the data until end of file is reached on the tape. In the same manner, a card-to-tape operation can be performed. (Note: The SP may set one entire record on a disk equal to another disk record, but he cannot set the contents of entire tracks, cylinders, or direct access devices equal to other complete tracks, cylinders, or devices.)

This module executes error return procedures in the following circumstances:

The number of SCBs in the parameter list is not two.

Operand 1 is $DOUT, and operand 2 is not (1) an externally qualified symbol and (2) equal to a symbolic or actual device address contained in the SSDAT.

Qualification of the operands is not valid.

An undefined symbol is one of the operands.

The length of the field addressed by Op2 is greater than 4096.

PATCH Command Processor (CEHKP/CZHYP)

Chart 57

The PATCH command allows the system programmer to alter the contents of a data field in real or virtual storage, or on an I/O device. This module maintains sufficient information about this patch in the SP's Patch Table to enable the restoration of the original contents of the data field.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKPA/CZHYPA. The input is a pointer to a parameter list that must contain pointers to two SCBs, neither of which may be undefined.

MODULES CALLED: This module calls the SET Command Processor (CEHKS/CZHYS) to (1) move the data specified by the first operand of the PATCH command into the PATCH Table and (2) move the data specified by the second operand into the location specified by the first operand.

EXITS: Exit is to Scan Control.

OPERATION: If all entry conditions are valid, this module builds a dummy working SCB list as input to the SET Command Processor. This list, which is used to alter the data fields, contains two SCBs:

The first SCB describes the space in the Patch Table into which the original data of the data field must be moved to be preserved.

The second SCB describes the data field specified by the first operand of the PATCH command, except that its length will be the same as the length attribute of the second operand of the PATCH command.

This module calls the SET Command Processor to save the information at the location receiving the patch. On return of control, this module again calls the SET Command Processor, using the original SCB list as input. SET inserts the patch.

The PATCH Command Processor constructs a Patch Control Block (PCB) in the System Programmer's Patch Table to completely describe the patch. The PCB and its associated data are the only record of the patch and are used to restore the data field by the REMOVE Command Processor, if it is called.

This module executes the error return procedures under the following conditions:

The SCB list it received as a parameter from Scan Control does not contain two SCBs.

Either SCB is undefined.

The proposed patch would overlay a patch that is identified in the Patch Table.

Insufficient space remains in the Patch Table to save the patch record.

The SET Command Processor returned an error code.

DUMP and DISPLAY Commands Processor (CEHKD/CZHYD)

Charts 58,59

DUMP/DISPLAY causes the contents of a specified data field to be written on a system device (printer or tape drive) for DUMP or the system programmer's terminal for DISPLAY. The principal differences between the commands are the output devices and the line lengths.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS) which calls it at CEHKDC/CZHYDC for DUMP and CEHKDB/CZHYDB for DISPLAY. The input parameter is a pointer to a parameter list containing pointers to the relevant SCBs. (The entry point named CEHKDE/CZHYDE is the Format entry point. Format is a service subroutine of this module and is called only by it.)

MODULES CALLED: This module calls:

Module Name and ID	Reason for Call
Format	To prepare each print line for output.
RSS or VSS Symbol Resolution (CEHMS,CZHWS)	To resolve $DOUT as the address of the output device.
I/O Control (CEHEA/CZHSA)	To write each block of formatted data.
$AT and $Patch Format (CEHJF/CZHZF)	To format the data fields represented by the symbols $AT and $PATCH.
Memory Map Format (CEHMM/CZHWM)	To format the data represented by the $MAP symbol.
$TASK/$STATUS Format (CEHJH/CZHZH)	To format the data represented by the $TASK and STATUS symbols.

EXITS: Exit is to Scan Control.

OPERATION: If the SCB represents one of the system symbols $AT, $PATCH, $TASK, $STATUS, OR $MAP, this module calls the external formatting routines to process the symbol. Otherwise, after checking for valid input, this module calls its Format subroutine to read in the required storage and convert a full page of the desired data field (DUMP) or two lines (DISPLAY) into printable form and move it into the print buffer. If Format returns a code of zero, it has finished formatting the data field. A return code of eight indicates that there is more data to be converted and moved into the print buffer.

For each print operation this module constructs an SIORCB containing the requested device address, the buffer address, and the buffer length. It links to I/O Control to write from the buffer onto the specified device. When control is returned, this module tests the return code from Format to determine if further data remains to be printed. If it does, this module calls Format again and repeats the above process until the operation is completed, at which time it returns control to the calling routine. If the request is for dumping a track, only 41 records can be dumped in one operation.

On each return from I/O Control, this module also tests the SIORCB for an Attention interruption received during the processing. If the SIORCB is flagged "asynchronous interruption received," this module returns to Scan Control, indicating that an interruption of the scan is requested, by a return code of four and a message specified in register 0 to acknowledge receipt of the Attention.

This module executes the error return procedures as a result of an:

Invalid number of operands

Undefined symbol

Invalid storage qualification

Invalid output device specified

Error return code from the FORMAT subroutine

Error return code from I/O

The Format Subroutine

Chart 59

Format causes the required storage to be read, converts the data field into printable form, and moves it into a print buffer. The length and type of the data field and a dump/display code are passed to it in an SCB.

ENTRIES: The entry point for this subroutine is CEHKDE/CZHYDE. It is called only by DUMP/DISPLAY Commands Processor.

MODULES CALLED: For special formatting procedures this subroutine calls:

Module Name and ID	Reason for Call
RSS Real Core Access (CEHCA)	If the mode is RSS, and the qualification of the SCB is RM, to page in the requested storage.
VSS Real Core Access (CZHPA)	If the mode is VSS, and the qualification of the SCB is RM, to page in the requested storage.
RSS Virtual Memory Access (CEHCB)	If the mode is RSS, and the qualification of the SCB is VM, to page in the storage.
VSS Virtual Memory Access (CZHPB)	If the mode is VSS, and the qualification of the SCB is VM, to read the storage into the paging buffer.
I/O Control (CEHEA/CZHSA)	In RSS or VSS to locate an external page and to read it into main storage.

EXITS: Exit is to the calling routine.

OPERATION: After causing the required storage to be read, this subroutine checks the type attribute of the input SCB to determine how the data is to be converted and moved into the print buffer. The work area field of the SCB contains X'00000032' if the command was DISPLAY and X'00000065' if it was DUMP. When a display is specified, the buffer length is two lines (100 bytes); when a dump is specified, the buffer is a full page.

When a dump operation is being performed, a primary header, "TSS STORAGE PRINT," is inserted in the output buffer before any formatting is performed. After each 56 lines for storage dumps, a secondary header is inserted in the output buffer. This secondary header is a variable field of 80 bytes denoted by the system symbol $DHDR and located at entry point CEHKDH/CZHYDH. After formatting is complete, the contents of this buffer are moved to a working buffer, and this buffer is cleared to all blanks.

Specific line formatting depends upon the type of data:

Integer data is converted into a signed decimal number and then converted into printable format word by word, three (DISPLAY) or six (DUMP) words of data at ten digits (plus algebraic sign) per word for each print line.

Character data is moved into the buffer in continuous string, with a period used for each non-blank character that has no graphic.

Hexadecimal data is converted into printable form and moved into a buffer with four (DISPLAY) or eight (DUMP) words of data at eight digits per word for each print line, plus the character representation of that line of data.

An address precedes each line of data. If the buffer is filled before all the data is converted, this subroutine returns to DUMP/DISPLAY with a return code of eight. When all the data has been converted, it returns a code of zero.

Memory Map Format (CEHMM/CZHWM)

Chart 60

This routine formats print lines or display lines for printing a storage map when the $MAP system symbol is used as the operand of a DUMP or DISPLAY command.

ENTRIES: This module is called by the Format subroutine of DUMP/DISPLAY (CEHKD/CZHYD), at CEHMMA/CZHWMA. The input parameter is an address of an SCB containing a $MAP in the symbol field.

MODULES CALLED: This module calls the Virtual Memory Access routine (CEHCB, CZHPB) when referring to the Task Dictionary.

EXITS: Exit is to the calling routine.

OPERATION: If the storage map is requested with RM qualification, the RSS version of this module sorts the Real Memory Symbol Table (CHARST) by address. This table, residing in real storage for RSS and in IVM for VSS, is ordered in ascending alphabetical order. (In VSS no sort is performed.) This module formats a one-page buffer (4092 bytes) from this table and, if the end of the table has not been reached, returns control to the calling routine with a return code of eight. If the input SCB for $MAP is qualified character type, the address sort in RSS is bypassed and the table is printed alphabetically.

The symbols and addresses are returned to DUMP/DISPLAY as follows:

data	symbol	blanks	address	blanks
length	8	1	8	3

where length is the number of characters.

Five groups of 20 characters each constitute one formatted print line. Two groups of 20 characters each constitute one formatted display line.

When this module reaches the last entry in the table in RSS, it re-sorts the symbol table into symbol order and indicates that the end of the table was reached by a return code of zero. The sort by address is performed at initial entry only. In VSS this table may be read by every task.

If a virtual storage map is requested, this module formats the Virtual Memory Map Table (CHAMAP) within the Task Dictionary Table (CHATDY). The table is arranged in order of ascending virtual storage addresses; sorting is unnecessary.

The formatted print or display line is the same as described for the map of real storage. PSECT and CSECT names are listed in ascending order of addresses, and entry point names are listed in the order in which they appear within the Control Section Dictionary, with an asterisk immediately following the address.

$AT and $PATCH Format (CEHJF/CZHZF)

Chart 61

This routine formats print lines or display lines, using information from the AT and Patch Tables, when the $AT or $PATCH system symbol has occurred as the operand of a DUMP or DISPLAY command.

ENTRIES: This routine is called by the DUMP/DISPLAY Commands Processor (CEHKD/CZHYD) at CEHJFA/CZHZFA. The input parameter is the address of an SCB containing $AT or $PATCH in the symbol field. The base field of this SCB contains the address of the AT SVC or PATCH associated with the input symbol.

MODULES CALLED: None.

EXITS: Exit is to the calling routine.

OPERATION: This routine determines whether the request is for $AT or $PATCH and refers to the appropriate control block to obtain the information it will use to format a

print or display line. If the input system symbol has no control block, the entire AT or Patch Table is formatted.

Patch: This module gets the following information from the PCB:

1. Qualification entry
2. The address of the patch and, if qualification is external, the
   a. symbolic device address (2 bytes)
   b. cylinder number (1 byte)
   c. track number (1 byte)
   d. record number (1 byte)
   e. offset from beginning of record (2 bytes)
3. Address of saved data in Patch Table.

This module formats the Patch data as follows:

Storage Qualification	Patch Address	Original Data	Patched Data

No more than one set of patch data is printed on a line, but one patch may require more than one print or display line when formatted.

AT: This module gets the following information from the ACB:

1. Qualification of the AT address
2. AT SVC location
3. Qualification of the AT command string
4. Pointer to the text and length of text

This module formats the AT data as follows:

Storage Qualification	AT SVC Address	Text Qualification	Command Text

No more than one set of AT data is printed on a line, but one AT may require more than one print or display line when formatted.

When this module completes the requested operation it returns a code of zero. Until then, if the format area has not been used up, this module returns a code of eight, requesting return of control from its calling routine.

$TASK and $STATUS Format Routine (CEHJH/CZHZH)

Charts 62,63

This routine formats print lines representing the various status indicators (PSWs, registers, etc.) associated with the $STATUS system symbol in RSS and the $TASK system symbol in both RSS and VSS.

ENTRIES: This routine is called by the DUMP/DISPLAY Command Processor (CEHKD/CZHYD) at CEHJHA/CZHZHA. The input parameter is the address of an SCB containing $TASK or (in RSS only) $STATUS in the symbol field. The base field contains the subscript value which represents a task ID for $TASK or a CPU ID for $STATUS.

MODULES CALLED: This module calls RSS Find TSI (CEHCF), RSS Real Core Access (CEHCA), and VSS Real Core Access (CZHPA).

EXITS: Exit is to the calling routine.

OPERATION: The input SCB is examined to determine the type of command being executed. If the command is not DUMP, an error return is made to the caller. For a dump request, the input symbol is tested for $TASK or $STATUS (RSS only).

For the RSS $TASK case, the subscript value located in the base portion of the SCB is tested for zero. The pointer to the current TSI is fetched from the RSS Status Save Area. If the subscript is not zero, the RSS Find TSI routine is called to get the TSI address for the task ID. The TSI is examined to determined if the XTSI has been "swapped out." If it has, a call is made to RSS VM Access (CEHCB) with a code of X'DD' to read the XTSI into a buffer (CEHCAT). The header portion of the XTSI is then copied into an area of working storage. If the XTSI was not "swapped out," RSS RM Access is called to get the XTSI, and the XTSI header is copied into working storage. Next, the task's ISA is fetched via RSS RM Access (CEHCA), and formatting begins.

In formatting, data is converted to printable characters and labeled. The data that is formatted and the source of the data are as follows:

- Task ID, User ID, TSI: from the TSI
- Task's current PSW, general control and floating-point registers, XTSI: from the XTSI
- Old virtual PSWs: from the ISA

In the case of RSS $STATUS, the subscript value in the base portion of the

input SCB must be 0, 1, or 2. If it is not one of these values, an error return is made to the caller. If the subscript value is 0, the primary Status Save Area is used as the source of the data. If the subscript value is 1 or 2, the ID of the CPU in which RSS is active must be determined. If the CPU ID in the subscript matches the ID of the active CPU, the primary Status Save Area is used. If the IDs do not match, and the system is not in duplex mode, an error return is made to the caller; otherwise, the secondary Status Save Area is used.

Once the Status Save Area is determined, the TSI pointer in it is retreived and the corresponding XTSI is fetched as described above. Formatting is then performed for the following fields:

- Task ID, CPU ID, current PSW, all old PSWs, CAW, CSW, general, control, and floating-point registers: from the Status Save Area
- TSI and XTSI: as located

For the VSS $TASK case, the task's current TSI is fetched by locating the PSA using VSS Real Core Access (CZHPA), taking the current TSI pointer from the PSA and fetching the TSI by means of VSS Real Core Access. Formatting is then performed for the following fields:

- The task's current VPSW, all old VPSWs, general, control, and floating point registers, task ID: from the VSS Status Save Area
- User ID and the TSI: from the TSI

The XTSI is not included in the VSS $TASK output.

REMOVE Command Processor (CEHKR/CZHYR)

Chart 64

The REMOVE command allows the system programmer to (1) delete any AT SVC he has implanted with the AT command and restore the original instruction, or (2) restore the original data in any data field previously patched with the PATCH command. This module manipulates the AT or Patch Tables accordingly.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS) which calls it at CEHKRA/CZHYRA. It is also called by the DISCONNECT Command Processor (CEHKM/CZHYM) and the AT SVC Processor (CEHJA/CZHZA).

MODULES CALLED: This module calls RSS Real Core Access (CEHCA), RSS VM Access (CEHCB), VSS Real Core Access (CZHPA), VSS VM Access (CZHPB), and I/O Control (CEHEA/CZHSA) as required by the qualification of the SCB to locate and make available the requested page.

EXITS: Exit is to the calling routine.

OPERATION: If the input data is valid, this module determines whether the removal of an AT or patch is requested.

If the SCB specifies $AT, this module determines the page upon which the implanted AT is located and has it brought into storage by one of the storage access methods.

This module locates the AT Control Block (ACB) and replaces the implanted AT SVC with the original instruction. If the SCB described a specific AT (for example, REMOVE $AT.XYZ), this module causes the page to be restored and, after removing the ACB, returns control to the calling routine. If $AT was not accompanied by a parameter, this module repeats the removal process until all ATs described in the system programmer's AT Table have been removed. In VSS, it also searches the Global AT Table, removing all those global ATs whose ACBs contain the TSP's task identification.

Note that with RM qualification, the AT or ATs are assumed to be recorded by ACBs in the MSP AT Table.

If, in VSS, this module is requested to remove an AT in real storage, it builds a command string from the input command string, and implants and remotely executes an SVC 70, requesting that RSS perform the REMOVE function for a specific AT SVC in real storage. If the address of the AT SVC is not supplied, RSS will reject the REMOVE request, since VSS cannot request by default the removal of all AT SVCs and their corresponding ACBs from real storage and the MSP's AT Table. A sample command string to accompany the SVC 70 appears as follows:

```
QUALIFY $RM(x); REMOVE $AT.L'1000'
```

where x is 0, 1, or 2.

The removal of all ATs implanted by a specific SP, except those in real storage, for a TSP is the process requested by the DISCONNECT Command Processor when it calls this module. If an error is likely in the AT SVC processing, CEHJA/CZHZA calls this module to remove the erroneous AT SVC.

If the SCB specifies $PATCH, this module acquires the page containing the patched data field and removes the Patch Control Block (PCB). If the SCB describes a specific patch, this module causes the page to be restored, using the same storage access method, and returns control to the calling routine. If the $PATCH was not accompanied by a parameter, this module repeats the removal process until all patches described by the SP's Patch Table have been removed.

This module executes the error procedures under the following conditions:

Undefined symbol

Invalid REMOVE operand

Wrong number of operands

CALL and END Commands Processor (CEHKL/CZHYL)

Chart 65

The function of the CALL command is to respecify the symbolic address of the TSSS input device in order to cause the reading of a prestored set of TSSS statements from cards or tape. The function of the END command is to define the end of a set of TSSS statements invoked by a CALL command and to restore the original symbolic device address of the TSSS input device.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS) which calls it at CEHKLA/CZHYLA (CALL) and CEHKLB/CZHYLB (END).

MODULES CALLED: None.

EXITS: Exit is to the calling routine.

OPERATION:

CALL: The input SCB contains the address of the device to be specified as the system input device, with a designation of the type of literal. If the literal type is a character, it specifies a symbolic address. Any other designation is the actual address. This module respecifies the primary input device in the Input Device Table (maintained by Language Control) by inserting the designated address in the first word of the table and setting the called device flag (LCRCAL) on.

END: The END routine reverses the work of the CALL command. This module restores the original symbolic device address of the TSSS input device by referring to the second word of the Input Device Table.

Both CALL and END routines exit to Scan Control with a return code of 4, with no message specified. This return code causes interruption of the scan and the system programmer is asked for more input, except in AT mode, when a RUN command is implied by the execution of END. In the case of the CALL command, Language Control (CEHLC/CZHXC) causes the called device to be read without inviting input.

DISCONNECT Command Processor (CEHKM/CZHYM)

Chart 66

This module disconnects the MSP or TSP from a terminal for which RSS or VSS, respectively, has been activated. The DISCONNECT command is a means by which the system programmer returns control to TSS/360.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKMA/CZHYMA.

MODULES CALLED: This module calls the REMOVE Command Processor (CEHKR/CZHYR) to remove all ATs implanted for the system programmer who issued the DISCONNECT command.

EXITS: Exit is to Scan Control.

OPERATION: This module simulates a REMOVE $AT command, in that it builds an SCB containing $AT in the symbol field, and calls the REMOVE Command Processor, using this SCB as a parameter. On return of control, this module restores the original symbolic device address for the TSSS input device (the END function) in the Input Device Table. The module also reinitializes the pointers in the SP Symbol Tables.

This module exits to Scan Control with a return code of 12, which indicates the successful processing of a DISCONNECT command.

STOP Command Processor (CEHKT/CZHYT)

Chart 67

The execution of a STOP command discontinues the current scan by Scan Control, and resets the AT mode switch from AT mode to conversational mode.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKTA/CZHYTA. The input parameter is the address of the working SCB list.

MODULES CALLED: None.

EXITS: Exit is to Scan Control.

OPERATION: The working SCB list contains no SCBs. This module resets the AT mode switch in the Status Save Area, reverting from AT mode.

If STOP is correctly used, this module returns to Scan Control with a return code of 4 and no message specified. As a result, the current scan is discontinued, and Language Control (CEHLC/CZHXC) requests additional input and reads the input device. In CALL mode the STOP command terminates the call, and requests input from the terminal.

RUN Command Processor (CEHKN/CZHYN)

Chart 68

The execution of the RUN command causes TSS/360 operation to resume, either where interrupted or at a specified address.

ENTRIES: This module is a keyword execution subroutine of Scan Control (CEHLS/CZHXS), which calls it at CEHKN/CZHYN.

MODULES CALLED: None.

EXITS: Exit is to Scan Control.

OPERATION: The working SCB list contains either none or one SCB. If one, this module computes the address at which TSS/360 is to regain control (base plus pointer) and inserts it in the "current" PSW (that is, the old PSW, in RSS, or the old VPSW, in VSS, stored when TSSS gained control). No attempt is made to verify the instruction address. This module restores the original symbolic device address of the TSSS input device whether TSSS is in call mode or not. If RUN has an operand, in RSS it turns off the "wait state bit" in the current PSW.

This module exits to Scan Control with a return code of 8 in order to indicate that a RUN command has been successfully processed. If the input SCB is undefined, or the SCB count is greater than one, this module executes the error return procedures.

INTRODUCTION

The TSSS Input/Output routines are conceptually identical for RSS and VSS. All synchronous I/O operations are performed serially and the I/O system maintains control until the requested I/O operation has been completed. In RSS, the entire TSS/360 operation waits. (TSS/360 has been halted as a part of the RSS activation process.) In VSS, the associated task waits, but the remainder of TSS/360 executes normally, since the VSS I/O operation is handled at the hardware interface level by the TSS/360 resident supervisor.

In this discussion, the modules that process I/O requests without need for error handling are described first. Those modules that perform error recovery management are described separately as an I/O subsystem. (See Figure 21 for the normal flow of TSSS I/O processing. For the general logic flow of the I/O Error Subsystem, see Figure 22.)

NORMAL PROCESSING

The I/O Control module provides the interface between each routine requesting I/O operations (called the I/O calling routine), and the I/O system. The I/O calling routine requests a specific operation through the parameters it passes to I/O Control in the TSSS Input/Output Request Control Block (SIORCB). By initializing, changing, and completing fields of the SIORCB, and by passing a pointer to it as a parameter, the I/O modules communicate between themselves on the state of the I/O operation. Aside from the RSS Loader's separate copy, only one copy of the SIORCB is required for RSS; only one is required for VSS. (The SIORCB is completely described in Appendix C.)

From the parameters in the SIORCB, I/O Control determines which TSSS access method -- console, direct access device, sequential, or telecommunications -- will service the request. The chosen access method builds a channel program for the request and links to I/O Initiation (CEHEB for RSS, CZHSB for VSS).

RSS initiates the I/O operation directly, issuing the Start I/O instruction, and RSS I/O routines process the resulting interruptions. VSS initiates I/O by building a TSS/360 I/O Request Control Block (IORCB) from the input SIORCB and then issuing the IOCAL system macro instruction. TSS/360 performs the I/O operation, and the TSS/360 supervisor queues the resulting interruptions for the task, whose VSS I/O routines process them.

RSS I/O Completion receives control from RSS I/O Initiation after the latter has checked for a successful start of I/O. RSS I/O Completion enters a wait loop pending receipt of the interruption that will signal termination of the operation, alternately enabling and disabling interruptions from the channel. Only device end terminates the loop. VSS I/O Initiation/Posting receives control when its next time slice occurs, if TSS/360 has recognized and queued a VSS I/O device end interruption. From the information stored in the ISA by TSS/360, VSS Initiation/Posting posts the results of the I/O operation in the SIORCB and exits to RSS/VSS I/O Completion.

When the I/O Completion routine processes the I/O interruption, it returns control to the routine that requested the I/O operation with notification of successful I/O Completion, or, if an error condition exists, it links to the error recovery subsystem. The I/O calling routine eventually receives notification via a return code in register 15 of the status of the I/O operation. If the return code is nonzero, a message control word is passed in register 0.

Return Code	Meaning
0	Successful completion of I/O operation
4	Irretrievable I/O error or SP Attention received
8	TSSS system logic error
12	Invalid input
16	Unit exception, (end of file)

Attributes and Characteristics

All of the TSSS I/O routines are reenterable and non-recursive. The RSS version execute in supervisor state with DAT active; the I/O routines' residency is specified within each module description. VSS copies reside in the task's Initial Virtual Memory and execute in privileged mode. In the area of normal processing there are no modules treated as subroutines or "modules called."

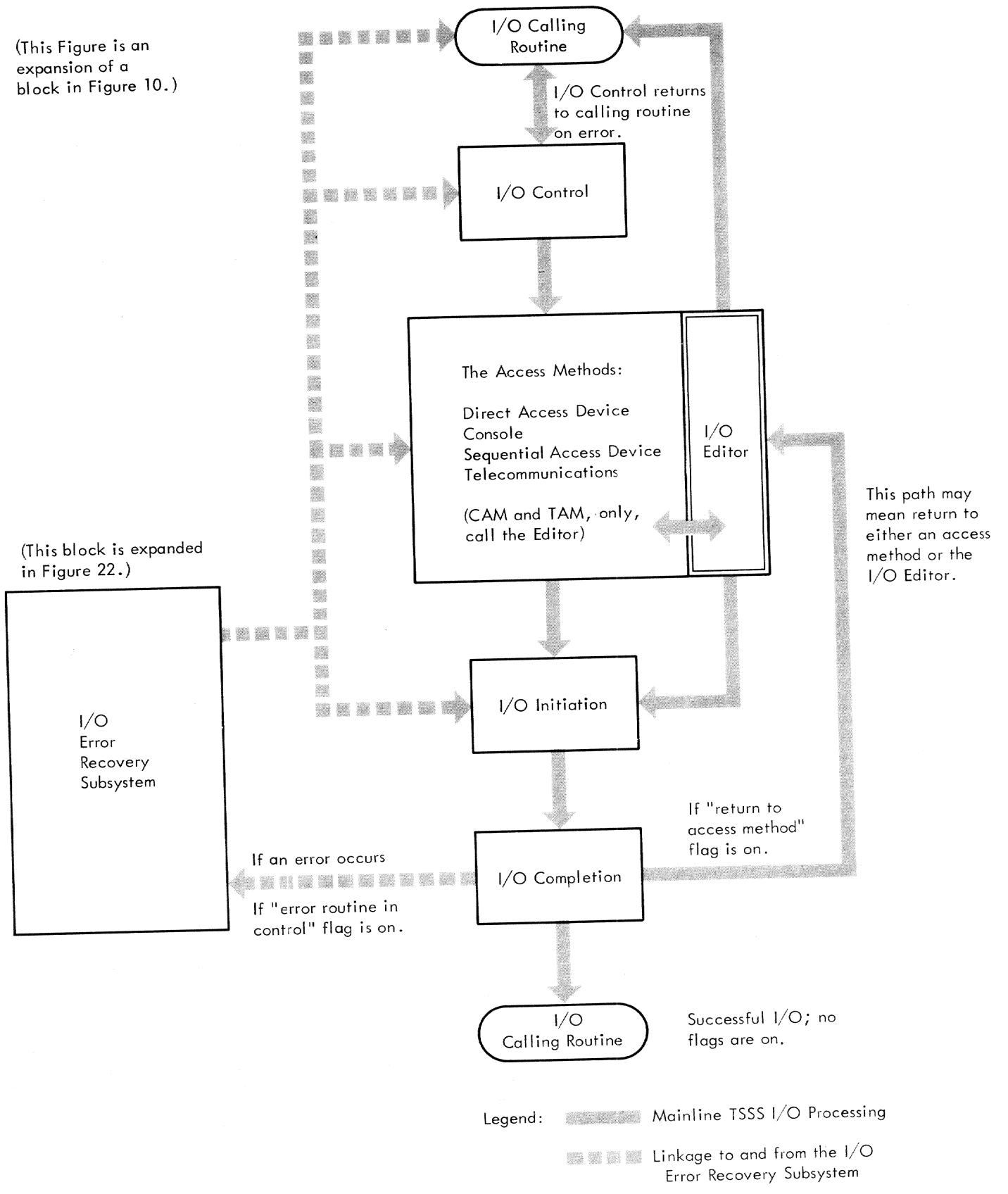
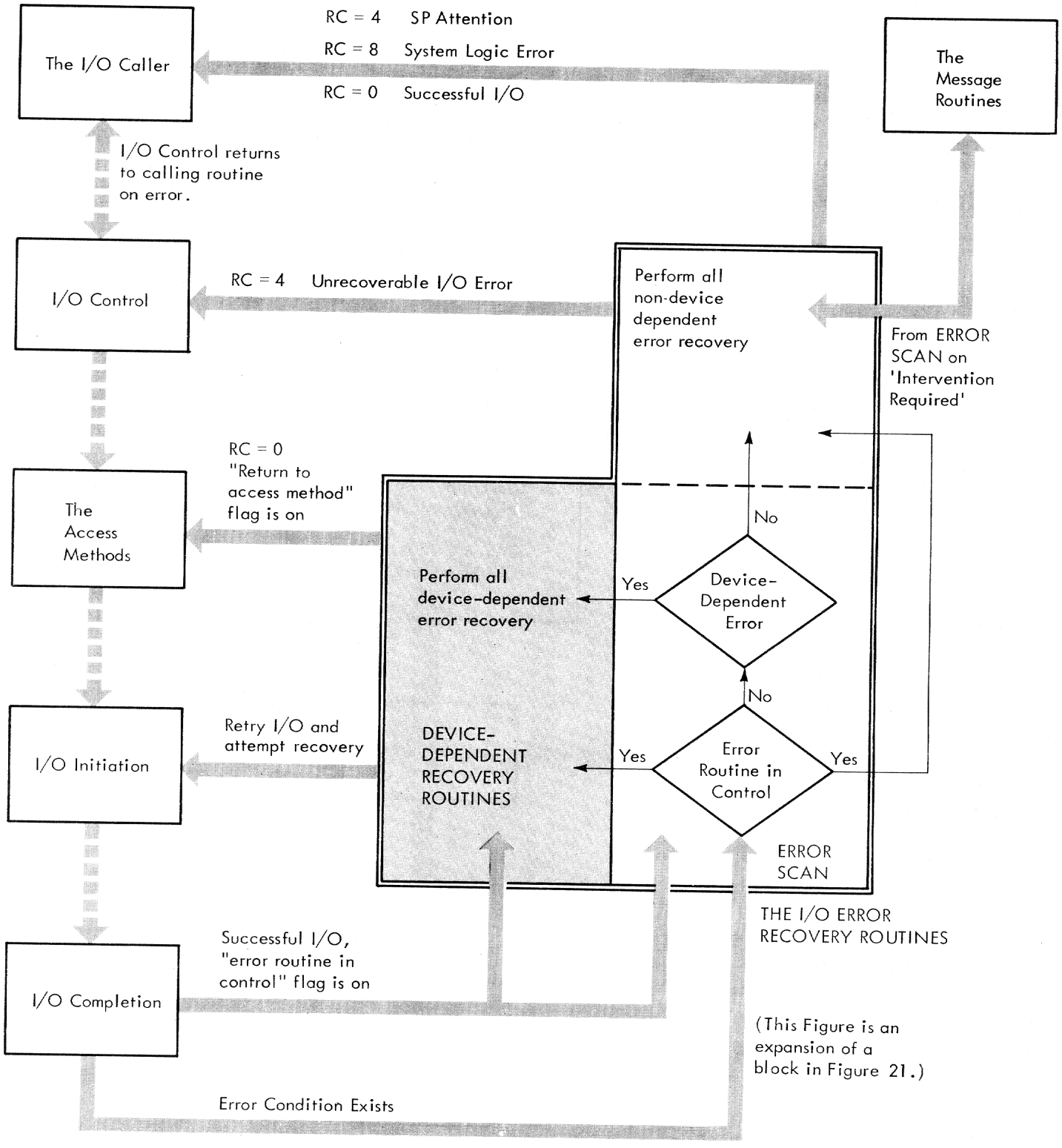


Figure 21. Overview of TSSS I/O processing



Legend:

Lines originating from the double-lined box indicate that any of the I/O Error Recovery Routines may make such a call. Dotted lines show rudimentary mainline I/O processing.

Figure 22. Overview of the TSSS I/O error recovery system

RSS/VSS I/O Control (CEHEA/CZHSA)

Chart 69

I/O Control acts as the interface between the I/O user (calling routine) and the I/O system. It determines which access method will service the I/O request and links to it.

ATTRIBUTES: The RSS copy of this module is resident.

ENTRIES: This module has two entry points:

CEHEAA/CZHSAA - Used for normal entry from I/O user routines.

CEHEAB/CZHSAB - Used by I/O Initiation or by an error recovery routine when a permanent I/O error has been detected.

EXITS: A normal exit is to one of the access methods, as requested by the parameters in the SIORCB:

Direct Access Device Method	CEHFA/CZHTA
Console Access Method	CEHFB/CZHTB
Sequential Access Method	CEHFC/CZHTC
Telecommunications Access Method	CEHFD/CZHTD

Under error conditions, exit is to the I/O calling routine.

OPERATION: I/O Control establishes a pointer to the SIORCB to be used as input to the I/O system, getting the pointer from the I/O calling routine's save area. I/O Control then initializes the system in order that it will be serially reusable. If the input data fields in the SIORCB are valid, it uses the symbolic device address (ECWASDA), passed by the calling routine, as a search argument for the TSSS Device Allocation Table (SSDAT). It moves the SSDAT entry into the SIORCB (ECWAGDE) and sets the pointer to the proper SADT entry. It then exits to the access method indicated by the device-defining information in the SSDAT.

The following error conditions cause this module to format a message control word in register 0 and return control to the I/O calling routine with a return code of 12.

Invalid data in the SIORCB

No SSDAT entry found for the input data

No physical path exists

Device not supported

Invalid SSDAT entry

If a permanent I/O error occurs, this module is called in order to try the alternate physical path, in an effort to bypass the error. If the alternate physical path may be used, and has not already been tried, this module exits to the access methods as usual. If it cannot be used, this module exits to the I/O calling routine with a return code of 4, and an appropriate message control word, signifying permanent I/O error.

Direct Access Device Access Method (CEHFA/CZHTA)

Chart 70

This module builds the channel program required to perform the requested I/O operation on a 2311, 2314, or 2301 direct access device. It also initializes fields in the SIORCB to be used by subsequent I/O modules.

ATTRIBUTES: The RSS copy of this module is resident.

ENTRIES: This module is called by I/O Control (CEHEA/CZHSA) at CEHFAA/CZHTAA. Input parameters are in the SIORCB.

EXITS: Normal exit is either to RSS or VSS I/O Initiation (CEHEB or CZHSB), depending on the mode. If invalid input is encountered, this module returns to the I/O calling routine.

OPERATION: If the input in the SIORCB is valid, this module determines if the I/O request is a Read Track Format. If it is, this module uses the input in the SIORCB to create a CCW list containing a Seek, Set File Mask, Read HA, Read R0, and 41 Read Count CCWs. This module initializes the first byte of the I/O calling routine's buffer with a X'29' (decimal 41) to show that 41 Read Counts are to be attempted. The channel program causes the counts of all the records on the track (up to a maximum of 41 records) to be read into the data area specified in the input parameters.

If the request is not Read Track Format, this module uses the input parameters to build a CCW list in the SIORCB to accomplish the requested I/O operation. On a Read request, if the "skip" flag is on in the SIORCB, a CCW is built to skip the number of doublewords listed in ECWASLEN before beginning a read. This routine stores the address of the first CCW to be executed in the CAW field of the SIORCB (ECWACAW), initializes pointers to the first and last CCWs in the chain, and exits to I/O Initiation.

The following error conditions cause this routine to restore the user's stored information, format a message control word, and return control with a code of 12 to the I/O calling routine.

Invalid operation code in SIORCB (ECWAOPCD)

Invalid data length specified in SIORCB (ECWALEN) for Read Track Format

Specified device is not a 2301, 2311, or 2314 direct access device

Channel program too long

Console Access Method (CEHFB/CZHTB)

Chart 71

This module creates the channel program required to perform a requested I/O operation on a 1052-7 Printer-Keyboard (commonly called the console). It also initializes fields in the SIORCB to be used by other I/O modules.

ATTRIBUTES: The RSS copy of this module is resident.

ENTRIES: This module is called by I/O Control (CEHEA/CZHSA) at CEHFBA/CZHTBA.

EXITS: Exit is normally to I/O Initiation (CEHEB-CZHSB). If an error is detected, this module exits to the I/O calling routine.

OPERATION: If the input in the SIORCB is valid, this module uses it to build CCWs in the SIORCB to perform the requested I/O operation. It stores the address of the first CCW to be executed in the CAW field of the SIORCB (ECWACAW).

If the request is for a read, this module sets the "return to access method" flag on in the SIORCB (ECWARTAM) and stores the address of the I/O Editor in the SIORCB (ECWARAM).

If the request is for a write, and the records are blocked, the length in the CCW will be the logical record length. This module initializes pointers to the first and last CCWs in the chain.

This module executes the error return procedures (return code 12) under the following conditions:

Invalid operation code

Invalid length specified

Sequential Access Method (CEHFC/CZHTC)

Chart 72

This module builds the channel program required to perform an I/O operation on a sequential device (card reader, printer, tape drive) as requested by the input parameters in the SIORCB.

ATTRIBUTES: The RSS copy of this module is nonresident.

ENTRIES: This module is called by I/O Control (CEHEA/CZHSA) at CEHFCA/CZHTCA.

EXIT: Exit is normally to I/O Initiation (CEHEB-CZHSB). Under error conditions, this module returns to the I/O calling routine.

OPERATION: If the input parameters are valid, this module builds CCWs in the SIORCB to perform the requested I/O operation. It stores the address of the first CCW to be executed (ECWACCWS) in the CAW field of the SIORCB (ECWACAW) and initializes pointers to the first and last CCWs in the chain.

If a request is for printing or writing tape and if a blocking factor is present, this module builds a list of CCWs instead of a single CCW.

If the request calls for a tape drive, and the tape is seven-track, this module builds a mode set CCW from information provided by the I/O calling routine in ECWAMODE and places it in the CCW list in the SIORCB. If the request is for a Read and the "skip" flag is on in the SIORCB, this module creates a CCW to skip the number of doublewords requested (ECWASLEN).

Under the following conditions this module executes the error return procedures (return code 12).

The requested device is not a card reader, printer, or tape drive.

An invalid operation code was received for the requested device.

An invalid forms motion byte was received from the printer.

The Read data length exceeds 256 bytes.

The channel program exceeds allowable length.

Telecommunications Access Method (CEHFD/CZHTD)

Charts 73,74

This module builds the channel program to perform an I/O operation on those devices supported by 2702 Transmission Control (1051, 1056, 1052, 2741, or Teletype Model 33 or 35 KSR terminal), from the input parameters passed to it in the SIORCB.

ATTRIBUTES: The RSS copy of this module is nonresident.

ENTRIES: This module is called by I/O Control (CEHEA/CZHSA) at CEHFDA/CZHTDA.

EXITS: Exit is normally to RSS or VSS I/O Initiation (CEHEB, CZHSB). If an error occurs, this module returns control to the I/O calling routine.

OPERATION: If the input parameters are valid, this module uses the information in the SIORCB to create a channel program in the SIORCB to perform the I/O operation. This module stores the address of the first CCW to be executed in the CAW field of the SIORCB (ECWACAW), turns on the "return to access method" flag (ECWARTAM), and sets up a reentry address in a field of the SIORCB (ECWARAM). This module initializes pointers to the first and last CCWs in the chain. The "issue Halt I/O sequence" mask is set on in the SIORCB (ECWATS).

Before a write instruction can be initiated, this module translates the output data to convert the EBCDIC characters to the appropriate device standard character codes.

On reentry, after a successful completion of the requested I/O operation, this module creates a channel program to issue a PREPARE command against the terminal, if the mode is RSS and the original request was Write. It stores the address in the CAW field, turns off the "return to access method" flag, turns on the "no interruption expected" flag in the SIORCB, and exits to I/O Initiation. The terminal is always left in a "prepared" state, so that the MSP is not locked out. In VSS, if the original request was write, this module returns directly to the I/O calling routine, since the Prepare command was chained to the original channel program by VSS I/O Initiation.

On reentry from a Read instruction, in RSS this module goes to the I/O Editor, which translates and edits the data read and moves it into the I/O calling routine's buffer before creating the channel program for the Prepare. In VSS this module merely exits to the I/O Editor. This module executes the error return procedures (return code 12) under the following conditions:

The requested device is not attached to a 2702 Transmission Control.

The operation code specified in the SIORCB (ECWAOPCD) is invalid for the requested device.

Data length of a write request is not greater than one.

For a read request, the specified data length exceeds 256 characters.

The channel program is too long.

I/O Editor (CEHFE/CZHTE)

Charts 75,76

If this module is called following a read operation, it edits the input, taking the action requested by the data within the input stream. If called by TAM it may also translate device standard character codes to EBCDIC.

ATTRIBUTES: The RSS copy of this module is nonresident.

ENTRIES: This module is called by I/O Completion when the following three conditions occur:

No errors occur during I/O processing.

The "return to access method" flag is on in the SIORCB.

The return address in the SIORCB is the address of this module.

The Console Access Method (CEHFB/CZHTB) turns on this flag and provides the Editor's address. This module is called by the Telecommunications Access Method when the operation requested is a read.

EXITS: This module exits in three possible ways:

Destination	Reason for Exit
I/O Initiation (CEHEB, CZHSB)	The continuation sequence occurs, or if there is a "cancel", to issue an additional read.
Telecommunications Access Method (CEHFD/CZHTD)	TAM was the access method in control and return is via the address in the SIORCB (ECWARAM).

The I/O Calling Routine	Neither of the above is true.

OPERATION: If the Telecommunication Access Method was in control, this module translates the input data from device type standard character codes to EBCDIC on a read. If the I/O request occurred in call mode (the call flag, ECWACAL, is on in the SIORCB), no edit is necessary. (In call mode, the 1056 Card Reader is the input device.)

Otherwise this module performs the edit function, removing all nonprintable characters. The following characters, or sequence of characters, require special action. The "pound sign" (#) is treated as a backspace character and is interchangeable with "backspace" on a 1052-7 only. On all other devices it is recognized as a character literal. The "left arrow" (←) is treated as a backspace character and is interchangeable with "backspace" on the Model 33 or 35 KSR Teletypewriter.

Backspace: This module deletes the backspace and the preceding character, unless backspace is the first character in the terminal read-in area, in which case only the backspace is deleted.

Backspace, carriage return: This module deletes all data read on the current read instruction, up through carriage return. If no good characters remain, this module issues another Read instruction. If good characters remain, the edit is performed on them.

Hyphen, carriage return: These characters at the end of a line indicate continuation. This module deletes end-of-block and issues another Read. Using the hyphen provides the capability of reading up to 256 characters in one linkage to the I/O system.

Following completion of its task, this module exits according to the "Exits" section of this description (see also Charts 75 and 76).

RSS I/O Initiation (CEHEB)

Chart 77

RSS I/O Initiation converts virtual storage addresses in the CCWs and the CAW to real storage addresses, issues the Start I/O Instruction, and checks for a good start.

ATTRIBUTES: This module is resident.

ENTRIES: This module is called by the RSS access methods, error recovery modules, and Error Scan and Recovery (CEHGE) at CEHEBA.

The TSS External Machine Check Interrupt Processor uses the entry points CEHEBB (Start I/O), CEHEBC (Test I/O), and CEHBD (Halt I/O) in order to test the actual instructions.

EXITS: A normal exit is to RSS I/O Completion (CEHHA). On the occurrence of a hardware failure, exit is to a special entry point within I/O Control (CEHEAB).

If a Load Real Address fails, control is passed to the I/O calling routine with a message control word in register 0 and return code 12 in register 15. If the "no interruption expected" flag is on in the SIORCB (ECWANIE), return is to the calling routine with return code zero in register 15.

OPERATION: This module retrieves the channel address word (CAW) from the SIORCB and stores it in location X'48', unless the modified CAW flag (ECWAMCW) is set in the SIORCB. If that flag is on, this module creates the CAW from the ECWAACAW field. After reconstructing the CCWs, this module initializes the pointer to the SIORCB in the SADT entry, pointed to by ECWASAPT, and turns on the appropriate flags. If requested, this module issues the "terminal Halt I/O" instruction before executing the Start I/O instruction.

The SIO instruction causes the condition code to be set. If the I/O Initiation is successful (condition code zero), this module exits to I/O Completion.

If the condition code is one, the Channel Status Word (CSW) has been stored. This module tests the CSW for the source of the condition code. If the CSW was stored as the result of a TSSS operation, this routine determines if the stored CSW belongs to TSS/360 or TSSS. If it does not belong to TSSS, this module simulates an interruption to TSS/360. When this module regains control, it reissues Start I/O. If the stored CSW belongs to TSSS, this module moves the stored CSW into the SADT, turns the "CSW stored on SIO" flag on, and exits to I/O Completion.

If the condition code is two, the channel is busy. This module enables and then disables interruptions for that channel, and reissues the SIO instruction.

If the condition code is three, the hardware is not operational. The module creates a message control word and exits to a point within I/O Control (CEHEAB).

VSS I/O Initiation/Posting (CZHSB)

Charts 78,79

This module creates an IORCB as input to TSS/360 and starts I/O for VSS via the TSS/360 IOCAL system macro instruction. On the occurrence of a synchronous interruption from the device, this module analyzes the status data as stored in the Interrupt Storage Area (CHAISA) and takes the appropriate action.

ATTRIBUTES: This module exists only in VSS.

ENTRIES: This module is entered at:

CZHSBA — If normal processing, from one of the access methods; if retry is in progress, from one of the error recovery routines.

CZHSBB — If a synchronous interruption occurs, from the TSS/360 Queue Scanner.

CZHSBC — If a code X'30' program check occurs, from the VSS Program Interrupt Processor (CZHNP).

All input required by this module is in the SIORCB.

EXITS: There are three possible exits from this module.

Destination	Reason for Exit
I/O Control (CZHSA), secondary entry point (CZHSAB)	SIO or HIO failed.
I/O calling routine	System logic error or device not allocated to task.
I/O Completion (CZHVA)	All other conditions.

OPERATION: This module uses the SIORCB to create an IORCB, which consists of a fixed section of flags and pointers, a variable size page list, and a variable size CCW list. This IORCB, in conjunction with the IOCAL macro instruction, causes TSS/360 to queue an I/O request for VSS.

If any special flags are on in the SIORCB, the corresponding flags are turned on in the IORCB. The delivery address for the resulting I/O interruption is set to the secondary entry point within this module for delivering interruptions (CZHSBB). This module then issues the IOCAL macro instruction, and forces time-slice end until interrupted by TSS/360.

If the Telecommunications Access Method built the channel program, this module constructs a Prepare CCW and chains it to the original channel program.

Upon entry at CZHSBB, as the result of the synchronous interrupt, this module examines the information stored in the ISA by TSS/360 to determine the success or failure of the I/O operation. This information includes the SIO condition code, the TIO condition code, and the CSW. This module exits accordingly.

When the VSS Program Interrupt Processor (CZHNP) encounters a program check caused by an attempt to perform I/O on an unavailable device (Code X'30'), it calls this module at CZHSBC. This module sets its return parameters to indicate invalid input (return code = 12) and returns control to the I/O calling routine.

RSS/VSS I/O Completion (CEHHA/CZHVA)

Chart 80

This module waits for and processes the synchronous I/O interruption, analyzes the resulting CSW, and, if necessary, links to Error Scan and Recovery.

ATTRIBUTES: The RSS version of this module is resident.

ENTRIES: This module is called by I/O Initiation and, under certain conditions, by the error recovery modules at CEHHAA/CZHVAA. Input parameters are in the SIORCB.

EXITS: There are four possible exits from this module.

Destination	Reason for Exit
Error Scan and Recovery (CEHGE)	An error exists.
One of the error routines	The "error route in control" flag is on.
One of the access methods	The "return to access method" flag is on.
The I/O calling routine	None of the above is true.

OPERATION: In RSS, if the CSW was not stored at Start I/O, and if the "interruption received" flag in the SADT is not on and if the SERR AUX queue is empty, this module waits for a device end interruption, by enabling and then disabling the appropriate channel. In VSS this module

does not receive control until the "interruption received" flag is on. When the CSW is stored as a result of device end, or if the CSW was stored at Start I/O, this module analyzes it. If an error condition exists, this module exits to the Error Scan and Recovery module (CEHGE). If not, it exits as described above under "Exits".

In RSS, if there is data in the SERR AUX queue, CEHHA checks the RSS active flag in each SERR AUX queue. If the flag is on, the queue is checked for CSW and sense information; if such information is present, the data is moved into the RSS work areas and processing continues as in the preceding paragraph.

THE I/O ERROR RECOVERY SUBSYSTEM

If the I/O Completion routine detects an error it links to the error recovery subsystem to attempt a retry of the operation that caused the error. (The error recovery subsystem is shown in Figure 22.) The interface module within the error recovery subsystem is RSS/VSS Error Scan and Recovery (CEHGE/CZHUE). This routine determines if a given error is device dependent or independent.

If the error is device dependent, Error Scan invokes the appropriate error recovery module, providing it with the necessary information to initiate error recovery procedures (for example, sense data). The number of retry attempts depends on the type of error encountered. There are four error recovery modules, corresponding to the four access methods:

- Direct Access Device Error Recovery (CEHGA/CZHUA)
- Console Access Device Error Recovery (CEHGB/CZHUB)
- Sequential Access Device Error Recovery (CEHGC/CZHUC)
- Telecommunications Access Device Error Recovery (CEHGD/CZHUD)

If the error is device-independent, Error Scan and Recovery attempts the recovery. If the error is corrected, the error recovery subsystem returns control directly to the I/O calling routine in language or environment or to the access method if the "return to access method" flag is on.

The error recovery modules may be reentered at several points during error recovery. The reentry points within the error-recovery modules are not referred to by external symbolic names. The reentry addresses are stored in the SIORCB by the particular error recovery module, depending upon which error it is processing. When Error Scan and Recovery reenters one of the error-recovery modules, it uses this field from the SIORCB (ECWARTN) as a branching address.

RSS/VSS Error Scan and Recovery (CEHGE/CZHUE)

Charts 81,82,83

This module identifies the error associated with an I/O request, and initiates error recovery. If the error is device-dependent, this module invokes device-dependent error-recovery routine, providing it with the information needed to recover (for example, sense data, filled-in SIORCB). If the error is device independent, this module attempts recovery.

ATTRIBUTES: The RSS copy of this module is resident.

ENTRIES: This module has five entry points.

Entry Point	Reason of Use
CEHGEA/CZHUEA	I/O Completion enters Error Scan whenever it detects an error.
CEHGEB/CZHUEB	Unit exception occurs on card reader or tape drive.
CEHGEC/CZHUEC	Intervention required on any device.
"Error Scan Reentry Point One"	When an error-recovery routine sets the "error routine in control" flag in the SIORCB, it also provides an error routine return address. If this flag is on when Error Scan is entered at CEHGEA/CZHUEA, this module exits to the error routine's address in the SIORCB, which may be this entry point within Error Scan itself. Entry may also be at this point if I/O retry is successful and I/O Completion finds the "error routine in control" flag on.
"Error Scan Reentry Point Two"	This module sets up this reentry point before exiting to I/O Initiation to issue a Sense instruction. It turns on the "error recovery in control" flag and is entered here from I/O Completion if the operation is successful.

Reason for Entry	Entry/Reentry Point	Reason for Exit	Exit Destination	Return Code
I/O Completion detects error.	CEHGEA/CZHUEA	Error in CSW not found. System error.	I/O User	8
No error detected; Error Scan is the "Error routine in control".	Reentry point 1	Successful I/O retry. "Return to access method" flag is not on.	I/O User	0
I/O Completion detects error.	CEHGEA/CZHUEA	To reissue I/O request or to issue SENSE.	I/O Initiation	N/A
Error Scan is routine in control; called by itself.	Reentry point 1	RC=12, indicating retry failure. Try again.	I/O Initiation	N/A
Error detected by I/O Completion; SENSE command has been issued.	Reentry point 2	1. Error is device dependent.	1. Device dependent error recovery	N/A
		2. Error is device independent, retry counter ≠ 0.	2. I/O Initiation	N/A
		3. Error is device independent, retry counter = 0.	3. I/O Control	4
Error indicated in CSW.	CEHGEA/CZHUEA	Permanent I/O error (try alternate path).	I/O Control (special entry point)	4
All retries have been attempted.	Reentry point 1	Try alternate path.	I/O Control	4
Return from SENSE Command.	Reentry point 2	Error on SENSE.	I/O Control	4
Permanent I/O error or retry failure.	CEHGEA/CZHUEA	Error routine in control.	The error recovery routines	4 or 12
I/O retry success; "return to access method" flag on.	Reentry points 1 and 2	Return to access method.	The access methods	0

Figure 23. Entries and exits from Error Scan

EXITS: Exits from Error Scan are shown in Figure 23.

OPERATION: Error Scan and Recovery (or simply "Error Scan") is never entered at its main entry point unless an error condition exists. It determines if the error is a channel control or interface control check; either condition indicates an irretrievable error, and Error Scan gives up control (see Figure 23). If the error is neither of these, this routine tests the "error routine in control" flag in the SIORCB, exiting to the error return address in the SIORCB if the flag is on.

If the flag is not on, this module determines the type of device on which the error occurred and executes an internal subroutine known as the Table Scan and Error Recovery subroutine. The tables it scans are specified by device. The Table Scan and Error Recovery subroutine tests the CSW and sense data for error conditions in accordance with the priority established in the appropriate scan table. If Error Scan does not detect an error during the Table Scan, a major system error has occurred. If a device-dependent error is detected, this module passes control to a device-dependent error routine, as determined by the Error Scan search table (see description of the error recovery routines). All device independent errors are handled by the error-recovery procedures in this module.

If the error is a unit exception for the 1052-7 Printer-Keyboard, this module returns control to I/O Initiation for up to 5 retries. For unit exception on the 1403 printer, this module links to the Sequential Access Device Error Recovery module (CEHGC/CZHUC). For unit exception on a tape drive or card reader, this module passes control to its own entry point, CEHGEB, to set the end-of-file return code of 16. For all other devices, the number of retries specified in the scan table is attempted.

If Error Scan detects a unit check, it saves the original request's information in the SIORCB itself and modifies the SIORCB to execute a Sense instruction, turns on the "error recovery in control" flag, inserts its second reentry point in the SIORCB as the error routine return address, and exits to I/O Initiation. When control returns, if the SENSE is unsuccessful, Error Scan exits accordingly. If the SENSE is successful the Error Scan processing continues.

As part of error recovery, Error Scan gets the retry count from the Scan Table, turns on the "error routine in control" flag, inserts its first reentry address as the error routine's return address, sets up the original SIORCB to reissue the original request, and exits to I/O Initiation.

If intervention required is the error, this module passes control to its own entry point CEHGEC. The internal subroutine at this entry point causes a message be written to the SP, specifying "intervention required," by using two different entry points in the RSS Message Writer routine (CEHCM). This routine normally calls the Message Writer at CEHCMA to write a message to the main operator's terminal or to the SP terminal, as required, using entry point CEHCMA. However, for an unrecoverable intervention-required error, this module calls the RSS Message Writer at CEHCMB.

If the intervention-required error occurs on a Terminal Access Method device, this module checks to determine if the situation occurred on a break operation. If it did, this module creates a channel program to enable the device. If the intervention-required error occurred on a 1052-7 Printer-Keyboard, this module creates a channel program to sound the control alarm.

Error Scan regains control at its main entry point if I/O Completion returned a code of 4 or 12.

Upon reentry at its first reentry point, this module determines if the return code is 0, 4, or 12. If 0, the I/O retry has been successful, and Error Scan exits accordingly. If it is 4, a permanent error exists, and this module performs the error-exit procedure. If it is 12, the retry of the original I/O request failed, and unless the retry count is now zero, Error Scan decrements the retry count by one, and repeats the process. If the retry count is zero, this module performs the error-exit procedure.

The error-exit procedure includes:

Loading a message control word in register 0; the message is "Permanent I/O Error".

Turning on the "print symbolic device address" flag and the "print actual path" flag in the SIORCB.

When applicable, turning on the "print CSW on error" flag, the "print PSW on error" flag, and the "print sense info on error" flag in the SIORCB.

Passing control to the secondary entry point (CEHEAB/CZHSAB) in I/O Control.

The I/O Error Recovery Routines

These routines perform all TSSS device-dependent error recovery. Each routine also handles certain problems that are unique to the devices for which it is responsible. The TSSS device-dependent error-recovery procedures are:

The error-recovery routine saves the calling routine information from the areas of the SIORCB that it uses for error recovery, turns on the "error recovery in control" flag (ECWAERCM), and inserts a return address in the SIORCB. Using the information in the SIORCB, passed to it by Error Scan and Recovery, the error routine builds a channel program in the SIORCB to attempt error recovery, storing the address of the first CCW to be executed in the CAW field of the SIORCB. It links to I/O Initiation (or to an access method) to try error recovery if necessary, then to retry the original request.

Upon reentry from either I/O Completion (CEHHA or CZHVA) or Error Scan, the error recovery routine checks the return code. A return code of 0 means that the I/O has been successfully completed, in which case the error recovery routine may:

Restore all user information, turn off the "error recovery in control" flag, place a return code of 0 in register 15 to indicate successful completion, and return either to the initial I/O calling routine or, if the "return to access

method" flag is on, to the address designated in the SIORCB (ECWARAM).

In some cases, it builds a new channel program and continues error recovery.

If the return code from Error Scan is 4, a channel control check or an interface control check has been detected, an unrecoverable I/O error. In this case, the routine restores the I/O calling routine's information, turns off the "error recovery in control" flag in the SIORCB, places a 4 in register 15, inserts a message control word in register 0, turns on the appropriate flags, and links to a special entry point in I/O Control (CEHEAB/CZHSAB) to try the alternate path.

A return code of 12 from Error Scan indicates a failure for the immediate error retry, other than a channel control check or an interface control check. The error recovery routine continues error recovery, unless the retry count is zero, in which case it follows the procedures for return code 4.

If a non-zero return code is received from Error Scan during execution of I/O that preceded the retry of the original request, the module involved follows the procedures for return code 4.

RSS/VSS Direct Access Device Error Recovery (CEHGA/CZHUA)

Charts 84,85

This module performs all the TSSS device-dependent error recovery associated with the 2311 Disk Storage Drive, the 2314 Direct Access Storage Facility, and the 2301 Parallel Drum.

ATTRIBUTES: The RSS version of this module is resident. It is serially reusable and nonrecursive.

ENTRIES: This module is called by Error Scan and Recovery (CEHGE/CZHUE) at:

Entry Points	Reasons for Entry
CEHGAA/CZHUAA	No record found
CEHGAB/CZHUAB	Seek check
CEHGAC/CZHUAC	Track condition check
CEHGAD/CZHUAD	File protection

EXITS: Exits are defined under "The I/O Error Recovery Routines."

OPERATION: For all entry conditions other than file protection, this module performs standard TSSS error recovery. File protect is a normal result of a Read Track Format request. No error recovery is required. This module calculates how many count fields have been read, excluding the home address record and record 0, and replaces the original 41 with the actual count in the first byte of the I/O calling routine's buffer. It then resets the conditions, restores the calling routine's information and returns control to the calling routine with zero in register 15.

If, however, the file protection occurred without the Read Track Format operation, this module performs the procedures for return code 4.

RSS/VSS Console Error Recovery (CEHGB/CZHUB)

Chart 86

This module performs all TSSS standard device-dependent error recovery on the 1052-7 Printer-Keyboard.

ATTRIBUTES: The RSS version of this module is nonresident.

ENTRIES: This module is called by Error Scan and Recovery (CEHGE/CZHUE) at:

Entry Points	Reason for Entry
CEHGBA/CZHUBA	Channel data check or bus out check
CEHGBB/CZHUBB	Equipment check
CEHGBC/CZHUBC	Attention

EXITS: In the case of an Attention, this module exits either to I/O Completion (CEHHA/CZHVA) or to I/O Initiation (CEHEB, CZHSB); exits for all other cases are defined under "The I/O Error Recovery Routines."

OPERATION: For all entry conditions other than Attention, this module performs standard TSSS error recovery. If a control alarm is required as part of error recovery, this module sounds it before exiting.

In the case of an Attention, if the operation was a write, this module sets on the "asynchronous interruption received" flag in the TSSS Active Device Table entry (SADT). It sets the device and channel end indicators and turns off the Attention bit before exiting to I/O Completion.

If the operation was a read, this module returns control to I/O Initiation to retry the original request.

RSS/VSS Sequential Access Device Error Recovery (CEHGC/CZHUC)

Charts 87,88

This module performs all the device-dependent error recovery for the 1403-2 Printer, the 1403-N1 Printer, the 2540 Card Reader, and the 2401 Magnetic Tape Unit, Models 1, 2, and 3.

ATTRIBUTES: The RSS version of this module is nonresident. The module is serially reusable and nonrecursive.

ENTRIES: This module is called by Error Scan and Recovery at:

Entry Points	Reason for Entry
CEHGCA/CZHUCA	Noise or data check on 2400 series
CEHGCB/CZHUCB	Channel data or bus out check on 2400 series
CEHGCC/CZHUCC	Unit exception or channel 9 on the 1403-2
CEHGCD/CZHUCD	Overrun or chaining check on 2400 series

EXITS: This module exits as described under "The I/O Error Recovery Routines."

OPERATION: If the operation was a read, this module checks for noise. A noise record is disregarded, and the original request is reissued. For all errors other than noise, the standard error recovery procedures are executed.

In addition, for unit exception or channel 9 on the printer, this module determines if all the lines have been printed. If they have, this module returns to the I/O calling routines with a return code of zero. Otherwise, it restarts the channel program at the next CCW.

RSS/VSS Telecommunications Error Recovery (CEHGD/CZHUD)

Chart 89

This module performs TSSS device-dependent error-recovery procedures for the devices attached to a 2702 Transmission Control: 1051/1056/1052, the 2741 communication terminal, and the 35 KSR Teletype.

ATTRIBUTES: The RSS version of this module is nonresident. The module is serially reusable and nonrecursive.

ENTRIES: This module is called by Error Scan and Recovery (CEHGE/CZHUE) at:

Entry Points	Reason for Entry
CEHGDA/CZHUDA	Attention
CEHGDB/CZHUDB	Status modifier

EXITS: If entry was for an Attention, this module exits either to I/O Completion (CEHHA/CZHVA) or to I/O Initiation (CEHEB, CZHSB). Other exits are defined under "The I/O Error Recovery Routines."

OPERATION: If entry is for status modifier, this module performs standard error-recovery. If entry is for an Attention, and the operation was a write, this module sets the "asynchronous interruption received" flag in the SADT entry of the SIORCB, turns off the Attention bit, and turns on the device and channel end in the SADT entry in the SIORCB before exiting to I/O Completion.

If the operation was a read, this module returns control to I/O Initiation to retry the original request.

Program Logic Manual

GY28-2022-2

Time Sharing Support System

Flowcharts on pages 81-170 were not scanned.

INTRODUCTION

These appendixes provide greater detail in certain areas than does the body of the document. Generally, these appendixes deal with tables and control blocks (appendixes B through F, inclusive).

The identification in parentheses which appears with a TSSS table name refers to the DSECT for that table. For example, TSS External Page Table has CHAEXT in parenthesis. The following TSS/360-compatible format is used in determining CSECT names for each table.

DSECT format is: CHAxxx
CSECT format is: CHBxxx

If two CSECTs exist for a single DSECT, one each in real and virtual storage, it is indicated as follows:

real storage CSECT: CHBxxxR
virtual storage CSECT: CHBxxxV

In addition, some DSECTS are represented by two CSECTs in real storage and two in virtual storage. In this case the indication is:

primary real storage CSECT: CHBxxxRA
other real storage CSECT: CHBxxxRB

primary virtual storage CSECT: CHBxxxVA
other virtual storage CSECT: CHBxxxVB

However, the name listed in this document with the table is the DSECT, unless one of the situations listed above occurs.

APPENDIX A: TSSS MODULE DIRECTORY

This appendix contains a list of all operating modules within TSSS in alphabetic order by module ID. Included for each module is its module ID, its chart ID, a list of each chart upon which it appears as a subroutine, and the modules that it calls.

Module Name and ID	Chart ID	Appears as a Subroutine on	Modules Called
RSS Channel Interrupt Processor (CEHAC)	09	---	None
RSS I/O Interrupt Processor (CEHAD)	08	---	None
RSS External Interrupt Processor (CEHAE)	01	---	CEHCH, CEHCC CEHBL, CEHLC CEHBU, CEHCM
RSS Program Interrupt Processor (CEHAP)	07	---	CEHBL, CEHCM
TSP Asynchronous Interrupt Processor (CEHAQ)	25	---	CEHCS, CEHCQ
RSS SVC Interrupt Processor (CEHAS)	10	---	CEHCH, CEHCC CEHCQ, CEHDA CEHDV
RSS Disconnect (CEHBD)	16	---	CEHBU, CEAL01 CEAAF
RSS Exit (CEHBE)	18	---	CEHBU, CEAIC
RSS Loader (CEHBL)	05	01, 07	CEHBT, CEHEA CEHCM
RSS External Page Location Address Translator (CEHBT/CZHRT)	06	06, 12, 13, 14, 17, 46	None
RSS Unloader (CEHBU)	17	01, 16, 18	CEHEA, CEHBT CEHCM
RSS Real Core Access (CEHCA)	12	11, 52, 56, 58, 64	CEHBT, CEHEA
RSS VM Access (CEHCB)	13, 14	11, 36, 49, 51, 52, 56, 59, 61, 62, 64	CEHCF, CEHBT CEHEA
RSS Inter-CPU Communications (CEHCC)	04	01, 10	CEAIC
Find TSI (CEHCF)	21	13,19,54	None
RSS Status Save Routine (CEHCH)	02, 03	01	CEAMW
RSS Message Writer (CEHCM)	15	01, 05, 07, 38, 40, 82	CEHEA

Module Name and ID	Chart ID	Appears as a Subroutine on	Modules Called
Queue VSS Interrupt (CEHCQ)	22	19, 23, 24, 25, 26, 54	CEAL01, CEAAF CEHCB, CEAAF2
RSS Interrupt Switching (CEHCS)	20	19, 23, 24, 25, 26, 54	CEAL01, CEAL02
Virtual Memory AT SVC Execution Processor (CEHDA)	24	---	CEHCS, CEHCQ
VSS Exit (CEHDE)	26	---	CEAAF, CEHCS CEAL01, CEHCQ
RSS/VSS LOGON SVC Processor (CEHDL)	19	---	CEHCF, CEHCS CEHCQ
RSS SVC Service Processor (CEHDR)	11	---	CEHCA, CEHJA CEHLC, CEHCB CEHCM
VSS Command SVC Processor (CEHDV)	23	---	CEHCS, CEHCQ
I/O Control (CEHEA/CZHSA)	69	05, 12, 13, 14, 15, 38, 56, 58, 59, 61, 64	None
RSS I/O Initiation (CEHEB)	77	Note: None of the I/O modules except Error Scan calls other modules as subroutines. These two columns are not applicable to TSSS I/O.	
Direct Access Device Access Method (CEHFA/CZHTA)	70		
Console Access Method (CEHFB/CZHTB)	71		
Sequential Access Method (CEHFC/CZHTC)	72		
Telecommunications Access Method (CEHFD/CZHTD)	73, 74		
I/O Editor (CEHFE/CZHTE)	75, 76		
DASDAM Error Recovery (CEHGA/CZHUA)	84, 85		
CAM Error Recovery (CEHGB/CZHUB)	86		
SAM Error Recovery (CEHGC/CZHUC)	87, 88		
TAM Error Recovery (CEHGD/CZHUD)	89		
Error Scan and Recovery (CEHGE/DAHUE)	81, 82, 83	---	CEHCM, CZHNM
RSS/VSS I/O Completion (CEHHA/CZHVA)	80	---	---
RSS/VSS AT SVC Processor (CEHJA/CZHZA)	36, 37	11, 27	CEHLC/CZHXC CEHKR/CZHYR CEHMS CEHCM, CZHNM CEHCB, CZHPB CEHCA

Module Name and ID	Chart ID	Appears as a Subroutine on	Modules Called
$AT/$PATCH Format (CEHJF/CZHZF)	61	58	CEHEA/CZHSA CEHCA, CZHPA CEHCB, CZHPB
RSS $STATUS/$TASK Format (CEHJH)	62	58	CEHCF, CEHCA CEHCB
AT Command Processor (CEHKA)	50	40	CEHCA, CZHPA CEHCB, CZHPB
COLLECT Command Processor (CEHKC/CZHYC)	55	40	CEHKS/CZHYS
DUMP/DISPLAY Commands Processor (CEHKD/CZHYD)	58, 59	40	CEHMM/CZHWM CEHJF/CZHZF CEHCA, CEHCB CZHPA, CZHPB CEHMS, CZHWS CEHEA/CZHSA CEHJH/CZHZH
DEFINE Command Processor (CEHKE/CZHYE)	52	40	None
CALL/END Commands Processor (CEHKL/CZHYL)	65	40	None
DISCONNECT Command Processor (CEHKM/CZHYM)	66	40	CEHKR/CZHYR
RUN Command Processor (CEHKN/CZHYN)	68	40	None
PATCH Command Processor (CEHKP/CZHYP)	57	40	CEHKS/CZHYS
QUALIFY Command Processor (CEHKQ/CZHYQ)	53	40	None
REMOVE Command Processor (CEHKR/CZHYR)	64	37, 40, 66	CEHCA, CEHCB CZHPA, CZHPB CEHEA/CZHSA
SET Command Processor (CEHKS/CZHYS)	56	40, 55, 57	CEHCA/CAHPA CEHCB/CZHPB CEHEA/CZHSA
STOP Command Processor (CEHKT/CZHYT)	67	40	None
RSS CONNECT Command Processor (CEHKW)	54	40	CEHCF, CEHCS CEHCQ
Operator Functions (CEHLA/CZHXA)	44, 45, 46, 47, 48	40	CEHMA/CZHWA CEHCA, CEHCB CZHPA, CZHPB CEHBT/CZHRT
Language Control (CEHLC/CZHXC)	38	01, 11, 27, 36	CEHEA/CZHSA CEHLP/CZHXP CEHLS/CZHXS CEHCM, CZHNM

Module Name and ID	Chart ID	Appears as a Subroutine on	Modules Called
Literal Resolution (CEHLL/CZHKL)	43	40	CEHMS, CZHWS
Source to Polish (CEHLP/CZHXP)	39	38	None
Scan Control (CEHLS/CZHXS)	40	38	CEHMS, CZHWS CEHLL/CZHXL CEHLA/CZHXA CEHKA/CZHYA CEHKE/CZHYE CEHKQ/CZHYQ CEHKW CEHKC/CZHYC CEHKS/CZHYS CEHKP/CZHYP CEHKD/CZHYD CEHKR/CZHYR CEHKL/CZHYL CEHKM/CZHYM CEHKN/CZHYN CEHKT/CZHYT CEHCM, CZHNM
Address to Symbol Resolution (CEHMA/CZHWA)	49	48	CEHCB, CZHPB
Memory Map Routine (CEHMM/CZHWM)	60	58	CEHCB, CZHPB
RSS Symbol Resolution (CEHMS)	41	40, 43, 58	CEHCB
VSS External Interrupt Processor (CZHNE)	30	---	CZHPR, CZHXC
VSS Message Writer (CZHNM)	34	36, 37, 38, 40, 82	CZHSA
VSS Program Interrupt Processor (CZHNP)	31	---	None
VSS Activate Interrupt Processor (CZHNV)	27, 28	---	CZHPS, CZHXC CZHZA, CZHPS, CZHMN
VSS Real Core Access (CZHPA)	32	56, 59, 64	None
VSS Virtual Memory Access (CZHPB)	33	36, 42, 49, 51, 56, 59, 64	None
VSS Restore Status (CZHPR)	35	---	CEAIS
VSS Status Save (CZHPS)	29	27	None
VSS I/O Initiation/Posting (CZHSB)	78, 79	---	None
VSS Symbol Resolution (CZHWS)	42	40, 43, 58	CZHPB
VSS AT Command Processor (CZHYA)	53	42	CZHPB
VSS $TASK Format (CZHZH)	63	60	CZHPA

APPENDIX B: THE RSS LOAD FUNCTION TABLES

The "load function tables" are those used by the RSS Loader (CEHBL) to write out pages of TSS/360, to read in pages of RSS, and to maintain a record of the transaction for use by the RSS Program Interrupt Processor (CEHAP) and the RSS Unloader (CEHBU).

The tables used for the "write out/read in" operation are built by the TSS/360 Startup procedures. They are:

Segment Table

Segment Two Page Table

Segment Two External Page Table

Segment Three Page Table

Segment Three External Page Table

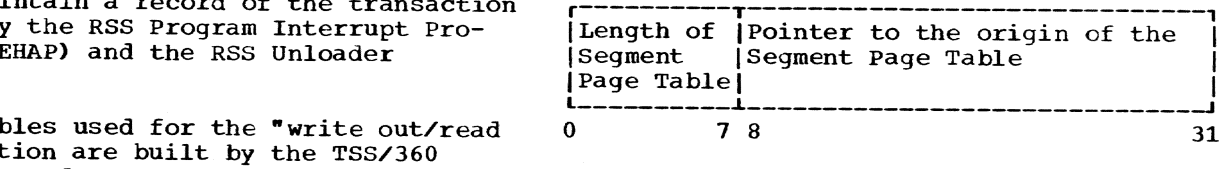
Segment Four External Page Table

The Segment Two Page Table, the Segment Two External Page Table, the Segment Three External Page Table, and the Segment Four External Page Table are sometimes referred to as the TSS Pageable Table, the RSS External Page Table, the Supervisor Symbol Dictionary Page Table, and the External Work Area Page Table. Startup also reserves space for the TSS External Page Table (CHAEXT).

To maintain a record of the load procedures, the RSS Loader builds another table -- the TSS External Page Table (CHAEXT) -- by placing entries in the table.

Segment Table

The Segment Table, consisting of five entries of four bytes each, provides a pointer to the Segment Page Table for each segment number. The RSS Loader (CEHBL) does not use the segment 0 and segment 1 entries. The segment 0 and 1 page tables are a means of mapping virtual addresses into real storage. When the system operates with DAT active, some of the modules have real storage addresses, some have virtual addresses. The hardware uses the segment 0 and 1 page tables to maintain address integrity while mapping TSSS modules into real storage. The Loader does use the third, fourth and fifth entries, which point to the Segment Two Page Table, the Segment Three Page Table and the Segment Four External Page Table, respectively. An entry in the Segment Table appears as follows:



The diagram shown in Figure 24 indicates the relationship between the Segment Table and the Segment Page Tables it points to, as this relationship is used by the RSS Loader.

There is a one-for-one correspondence between the entries in a given page table and the entries in the corresponding external page table.

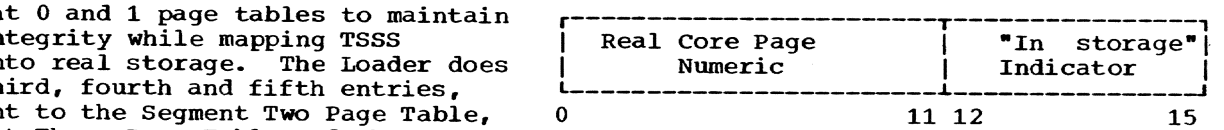
Segment Two Page Table (TSS Pageable Table)

The Page Table for segment two page addresses indicates those TSS/360 Supervisor read-only pages which may be written out of real storage onto an external device. Each vacated space provides a location in real storage for an RSS page. The addresses are in ascending order. It is 30 bytes long, each entry being 2 bytes in length.

The RSS Loader (CEHBL) refers to this table during dynamic loading. This table is later used by the RSS Unloader (CEHBU) and the RSS Program Interrupt Processor (CEHAP).

When a TSS/360 Supervisor page is replaced by a transient RSS page, the Loader flags the entry for that page as "in storage", in this table, in order to maintain a record of the page's location. During RSS deactivation, when the Supervisor page is read back into real storage, the RSS Unloader turns off the "in storage" flag. The Program Interrupt Processor uses an entry in this table to calculate the corresponding entry in the Segment Two External Page Table, which it uses as input to the RSS Loader.

An entry in this table is as follows:



bits twelve-fifteen are initialized to 1000

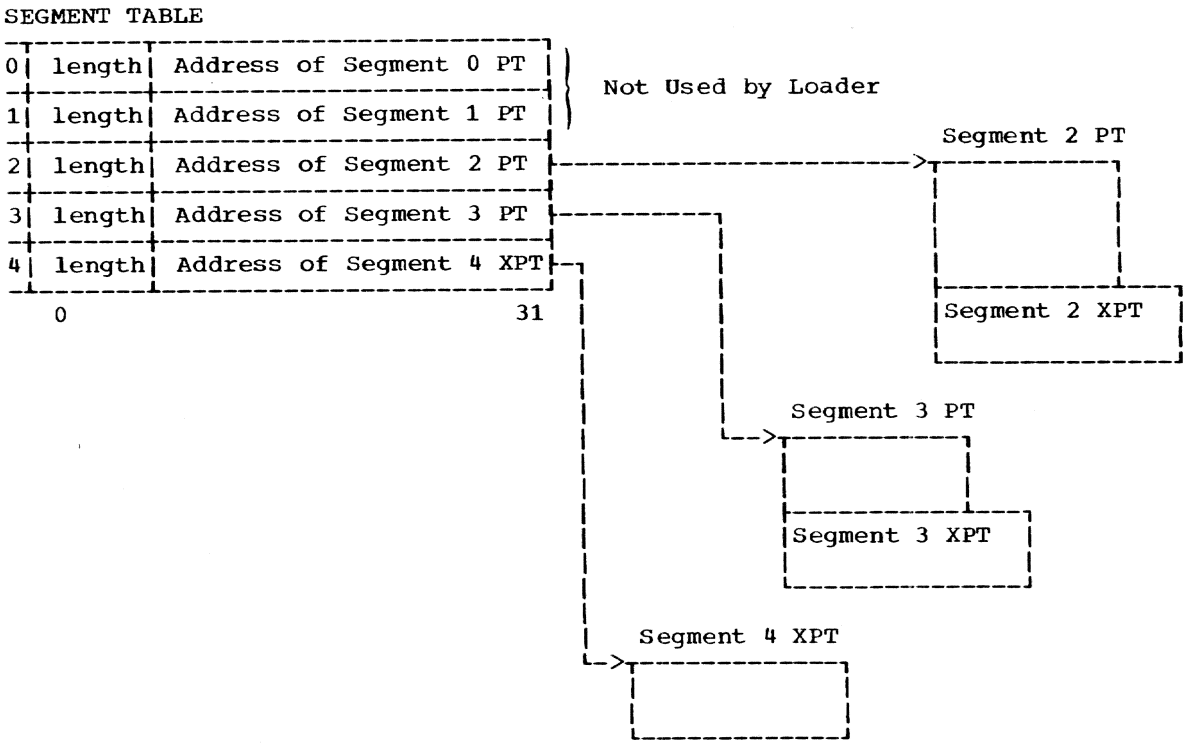


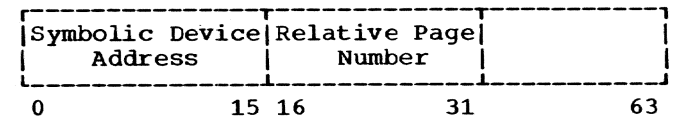
Figure 24. The Relationship between the Segment Table and the Segment Page Tables

Segment Two External Page Table (RSS External Page Table)

The entries in the External Page Table for segment two addresses have a one-to-one correspondence with the entries in the Segment Two Page Table. This table contains the addresses of the transient RSS pages which may be read into real storage at the locations vacated by TSS/360 Supervisor Pages listed in the Segment Two Page Table. The addresses are in ascending order. Each entry is 8 bytes in length.

The RSS Loader refers to this table during dynamic loading if such a page caused the paging exception. It is used later by the RSS Unloader. This table provides an address list for the direct access device upon which each transient RSS page resides, prior to loading.

An entry in this table is as follows:



Segment Three Page Table

The Page Table for segment three page addresses indicates those TSS/360 pages which may be written out of real storage onto an external device. Each vacated space provides a location in real storage for a TSS/360 Symbol Dictionary Page. The addresses are in ascending order. It is six bytes long, each entry being two bytes long, although this figure is dependent on the length of the Symbol Dictionary.

The RSS Loader (CEHBL) refers to this table in order to load a Symbol Dictionary if it causes a paging exception program check. This table is used later by the RSS Unloader (CEHBU).

When a Supervisor page is replaced by a Symbol Dictionary page, the Loader flags the entry in the Page Table as "in storage". During RSS deactivation, when the Supervisor page is read back into storage, the RSS Unloader turns off the "in storage" flag. The Program Interrupt Processor (CEHAP) uses an entry in this table to calculate the corresponding entry in the Segment Three External Page Table, which it uses as input to the RSS Loader.

An entry in this table has the same format as an entry in the Segment Two Page Table (see the diagram for that table).

Segment Three External Page Table (Symbol Dictionary Table)

The External Page Table for segment three addresses has a one-to-one correspon-

dence with the entries in the Segment Three Page Table. This table indicates the addresses of the Supervisor Symbol Dictionary pages that may be read into real storage at the locations vacated by the Supervisor pages listed in the Segment Three Page Table. The addresses are in ascending order. Each entry is 8 bytes long; the length of the table depends on the number of entries.

The RSS Loader refers to this table during dynamic loading. It is used later by the RSS Unloader. This table provides an address list for the direct access device upon which each Symbol Dictonary page resides prior to loading.

An entry in this table has the same format as an entry in the Segment Two External Page Table (see the diagram for that table).

Segment Four External Page Table (External Work Area Table)

The External Page Table for segment four is a list of available locations on a direct access device. This list provides the addresses of temporary storage locations in which the written-out, segments two and three Supervisor pages can reside while RSS is active.

This table is at least 72 bytes long; it contains as many entries as the sum of the entries in the Segment 2 and Segment 3 page tables. Each entry is eight bytes long. It is used by the RSS Loader (CEHBL) during the write operation. When, during deactivation, the RSS Unloader wishes to retrieve the Supervisor pages and read them back into real storage, it refers to this table for the page's current address.

An entry in this table has the same format as an entry in the Segment Two External Page Table.

TSS External Page Table (CHAEXT)

This table is used to define and correlate the real storage addresses and the external locations of TSS Supervisor pages that are written out during loading. The RSS Loader (CEHBL) builds and makes entries in this table during dynamic loading. 288 bytes of real storage are reserved for this table, starting on a double word boundary; it may have up to 18 entries, each 16 bytes long.

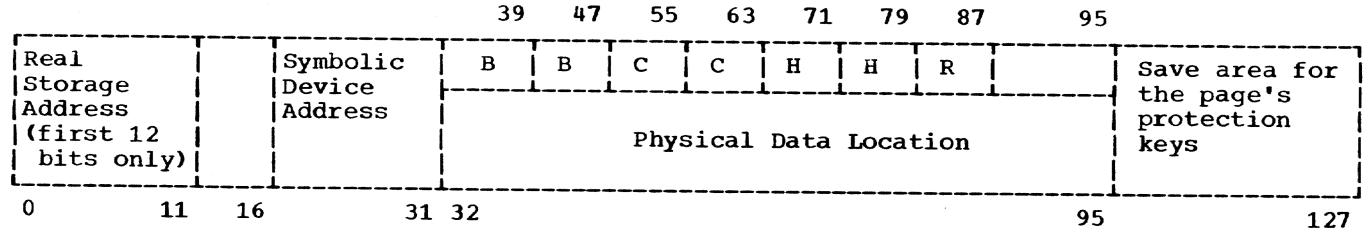
Additional entries or changes to this table occur when it is later used by:

The RSS Unloader (CEHBU) to restore the written-out Supervisor pages during RSS deactivation

RSS Real Core Access (CEHCA) to determine the status of a requested page

The RSS Disconnect module (CEHBD) or the RSS Exit module (CEHBE) reinitializes this table to X'FF's.

The format of an entry in the TSS External Page Table is shown in Figure 25.



B, C, H, and R, represents a data field of 8 bits, B is the bin number, C is the cylinder number, H is a head number, and R is the record ID.

Figure 25. An entry in the TSS External Page Table (CHAEXT)

The TSSS I/O System defines three tables:

- Support System Device Allocation Table (CHAECX)
- Support System Active Device Table (part of CHASYS)
- Support System Input/Output Request Control Block (CHAECW)

These tables are collectively called the Device Assignment Tables, and are referred to as SSDAT, SADT, and SIORCB, respectively. The SSDAT and the SADT are built by the TSS SYSGEN/STARTUP procedures, although the SADT is completed by TSSS.

The Device Allocation Table (CHAECX)

The TSSS Device Allocation Table or SSDAT defines certain information about the TSS/360 devices to TSSS I/O. The VSS copy of the SSDAT resides in IVM. The RSS copy is divided into a resident and a transient portion.

The resident portion of the SSDAT comprises a 12-byte header and four 12-byte entries. The first entry is contiguous with the header and defines the main operator's terminal. The second, third, and fourth device entries are contiguous with the first and define the RSS residence devices.

The remainder of SSDAT is nonresident and is loaded by the RSS Loader (CEHBL) when RSS is activated. This portion consists of one device entry for every device in the system, arranged in ascending order by symbolic device address. All entries are contiguous in the transient portion.

The SSDAT is created by TSS/360 SYSGEN/STARTUP from information contained in the SDAT (the TSS/360 Device Allocation Table) and the path-finding tables. The transient portion of this table is stored in a predefined location on the RSS residence device.

The following diagram is an example of the SSDAT header and a general device entry in the SSDAT--resident and transient.

SSDAT Header--12 bytes

Relative Location	Description of Field	Created By	Format at SYSGEN
0 to 3	Pointer to first non-resident device entry	SYSGEN/STARTUP	Virtual address pointer
4 to 7	Pointer to last non-resident device entry	SYSGEN/STARTUP	Virtual address pointer
8 to 11	Reserved for use by I/O system	SYSGEN/STARTUP	Initialized to 0

SSDAT General Device Entry--12 bytes

Relative Location	Description of Field	Created By	Format at SYSGEN
0 and 1	Symbolic device address	SYSGEN	SDASDA field from SDAT. If this is a resident entry and TSSS does not reside on the residence device specified, this field has all bits on.
2 and 3	Physical Path	SYSGEN	Path from path finding table. If this is a resident entry and TSSS does not reside on the residence device specified, this field has all bits on.
4 and 5	Alternate physical path	SYSGEN	Alternate path from path finding tables, or X'FFFF' if none exists.
6	Flag	SYSGEN	Byte 6 bit 0 = 1 if device is VAM-formatted. Byte 6 bit 1 = 1 if device can be called.
7	Reserved for future use		
8 to 11	device defining information	SYSGEN	SDADEV field from SDAT.

The Active Device Table

The Active Device Table or SADT, which exists as part of the TSS/360 System Table (CHASYS), is used by the TSS/360 interruption filter to determine which, if any, RSS I/O devices are active, as well as which interruptions belong to RSS. It is also used by the RSS I/O Interrupt Processor (CEHAD) to determine whether an interruption is synchronous or asynchronous. The VSS copy of the SADT resides in the TSS/VSS Status Save Area.

This table consists of two 24-byte entries, which are initially formatted by TSS/360 SYSGEN/STARTUP to all zeroes.

The format of the SADT appears below:

Relative Location	Description of Field	Created by
	ENTRY 1 (SYSRIO)*	LOCATION: CHASYS +188
0 and 1	Physical Path SP terminal (SYSRPP)	Startup, Environment Area, I/O Initiation
2 and 3	Flags: (SYSRFL) Bit 0: Interrupt expected (SYSII) 1: Asynchronous interruption expected (SYSAI) 2: Interrupt received (SYSIR) 3: Asynchronous interruption received (SYSAR) 4: CSW stored on SIO expected (SYSSE) 5: CSW stored on SIO received (SYSSR)	Startup, RSS I/O Initiation, RSS I/O Interrupt Processor, RSS I/O Completion
4 to 7	Pointer to SIORCB (SYSRCB)	I/O Initiation
8 to 15	CSW (SYSRCS)	I/O Initiation -- RSS I/O Interrupt Processor -- interruption.
15 to 23	PSW (SYSRPS)	RSS I/O Interrupt Processor -- interruption I/O Initiation
*In VSS, the SADT begins at EVSRIO in the VSS Status Save Area, and all names shown as SYSxxx will appear as EVSxxx.		
	ENTRY 2	
0 and 1	Physical path--device other than SP terminal	I/O Initiation
2 and 3	Flags (as in ENTRY)	I/O Initiation, RSS I/O Interrupt Processor, I/O Completion
4 to 23	Same as in ENTRY 1	Same as ENTRY 1

The I/O Request Control Block (CHAECW)

The SIORCB is a table used by RSS and VSS. It serves as the communications area between modules requesting I/O and the I/O system, as well as between the I/O system modules. All parameters and status indicators pertaining to a given I/O operation are passed in the SIORCB.

RSS SIORCB

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description	Field Set By	Field Used By
72	0	0	ECWASAVE	I/O System Save Area	I/O System	RSS Message Writer
4	+48	+72	ECWAPSCT	Pointer to SIORCB	Startup	
2	+4C	+76	ECWASDA	Symbolic Device Address	I/O User	I/O Control
2	+4E	+78	ECWAUFL1 ECWAUFL2	User flag bytes	I/O User	Access method I/O Control
4	+50	+80	ECWABFFR	Start Address of CCW and buffer area or of data field	I/O User	Access method
1	+54	+84	ECWAOPCD	Operation code as required by access methods	I/O User	Access method
1	+55	+85	ECWAACMD ECWAMODE	Actual Command Code EQU ECWAACMD, mode set for 7-track tape	I/O User	Access method
2	+56	+86	ECWALEN	Length in bytes of data area to be read	I/O User	Access method I/O Control
4	+58	+88	ECWALRCL	Logical Record Length	I/O User	Access method
2	+5C	+92	ECWASLEN	Number of double words to be skipped before reading data	I/O User	Access method
2	+5E	+94	ECWARES1	Unused	--	--
8	+60	+96	ECWASEEK	Seek Address	I/O User	Access method
64	+68	+104	ECWASCSV	Error Scan Save Area	I/O System	Error Scan
2	+A8	+168	ECWASDAT	Symbolic Device Address from SSDAT entry	I/O Control	Access method error routines
2	+AA	+170	ECWAPHP	Physical path from SSDAT entry	I/O Control	I/O Control
2	+AC	+172	ECWAPHP2	Alternate physical path	I/O Control	Access method error routines
2	+AE	+174	ECWAFL1 ECWAFL2	System flags from SSDAT	I/O Control	Access method error routines
4	+B0	+176	ECWADEV	Device defining information from SSDAT	I/O Control	Access method error routines

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description	Field Set By	Field Used By
4	+B4	+180	ECWACAW	Channel Address Word	Access Method Error routines	I/O Initiation
4	+B8	+184	ECWASAPT	Pointer to SADT entry	I/O Control	I/O Initiation I/O Completion
8	+BC	+188	ECWACSW	Channel Status Word	I/O Completion	Error routines
8	+C4	+196	ECWAPSW	Program Status Word	I/O Completion	RSS Message Writer
2	+CC	+204	ECWAIC01	Extended PSW interruption code		
2	+CE	+206	ECWAERCT	Error retry	Error routines	Error routines
8	+D0	+208	ECWASENS	Sense Data	Error Scan	Error routines
4	+D8	+216	ECWARTN	Error recovery return address	device-dependent Error Recovery	I/O Completion I/O Initiation Error Scan
4	+DC	+220	ECWARAM	Access method return address	Access method	I/O Initiation I/O Completion Error routines
8	+EO	+224	ECWAACSW	Save area for CSW	Error routines	Error Routines
8	+E8	+232	ECWAAPSW	Save area for PSW	Error routines	Error Routines
2	+F0	+240	ECWAIC02	Extended PSW interruption code save area		
2	+F2	+242	ECWALENV	ECWALEN save area for I/O Editor	--	I/O Editor
4	+F4	+244	ECWAACAW	Save area for CAW	Error routines	Error Routines and I/O Initiation
4	+F8	+248	ECWASFRS	ECWAFRST address save area	Error routines	Error Routines
4	+FC	+252	ECWASLST	ECWALAST address save area	Error routines	Error Routines
8	/100	+256	EWAREC	Work area		

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description	Field Set By	Field Used By
24	+108	+264	ECWACLOA	Channel Log Out Area	Error Scan	RSS Message Writer, VSS Message Writer
4	+120	+288	ECWABFFV	ECWABFFR save area for I/O Editor	--	Editor
1	+124	+292	ECWAAAOP	Save area for actual command code	Error routines	Error routines
1	+125	+293	ECWAAOP	Save area for op code	Error routines	Error routines
2	+126	+294	ECWAXSAV	Save area for residual count	I/O Completion	I/O Completion
4	+128	+296	ECWASFLA	System Flag Bytes	I/O System	I/O System
4	+12C	+300	ECWARES2	I/O Work Area	--	I/O System
256	+130	+304	ECWATRIN	Terminal Read In Area (Force doubleword boundary)	--	TAM, CAM, Editor, I/O User
80	+230	+560	ECWAIORF	IORCB Flags	VSS Initialization	TSS/360
64	+280	+640	ECWAPGLS	IORCB page list	VSS Initialization	TSS/360
80	+2C0	+704	ECWACCWS	Error recovery CCW list	Error routines	Error routines I/O Initiation
720	+310	+784	ECWACCWF	Access method CCW list	Access method	I/O Initiation Error routines
4	+5E0	+1504	ECWAFRST	Pointer to first active CCW	Access method error routines	Error routines
4	+5E4	+1508	ECWALAST	Pointer to last active CCW	Access method error routines	Error routines I/O Initiation
124	+5E8	+1512	--	The V-type Address Constants		
	+664	+1628	ECWALDB	End of Load Area		

Source to Polish (CEHLP/CZHSP) constructs a polish string through the following steps:

1. STEP. Source to Polish refers to the next item in the source string, deciphers what it is, and assigns it a classified type.

2. WORK. Source to Polish fetches the appropriate action code from the Action Code Matrix by multiplication, and performs the function requested by the action code.

3. HOOKUP. When it encounters an "end" character, Source to Polish makes the symbol string follow the name string to form what is now the polish string.

Source to Polish uses three tables in constructing the polish string: the Table of Delimiters and Allowable Characters, the Table of Codes, and the Action Code Matrix. It uses the first two tables during STEP to decipher each item and to classify it, respectively. These tables are shown in this appendix.

A finished polish string is formatted as follows:

1. The first fullword in the polish string contains the number of bytes in the length of the name string, plus 4 bytes.

2. The Name String: A series of halfwords that are either (1) pointers to entries in the symbol string or (2) representations of operators and keywords.

3. A fullword byte count for the symbol string.

4. The Symbol String: Each entry has a length and type attribute, and the EBCDIC representation of the symbol or literal.

Table of Delimiters and Allowable Characters in Hexadecimal

An entry in a row represents the first hexadecimal digit of a source item; an entry in a column represents the second hexadecimal digit.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
1	01	01	01	01	01	03	01	01	01	01	01	01	01	01	01	01
2	01	01	01	01	01	01	8E	01	01	01	01	01	01	01	01	01
3	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4	03	01	01	01	01	01	01	01	01	01	01	. 07	< 89	(85	+ 81	| 8C
5	& 8B	01	01	01	01	01	01	01	01	01	01	$ 05	* 83	) 86	; 8F	8D
6	- 82	/ 84	01	01	01	01	01	01	01	01	01	' 91	% 94	01	> 88	01
7	01	01	01	01	01	01	01	01	01	01	: 8A	01	01	' 04	= 87	01
8	01	a 02	b 02	c 02	d 02	e 02	f 02	g 02	h 02	i 02	01	01	01	01	01	01
9	01	j 02	k 02	l 02	m 02	n 02	o 02	p 02	q 02	r 02	01	01	01	01	01	01
A	01	01	s 02	t 02	u 02	v 02	w 02	x 02	y 02	z 02	01	01	01	01	01	01
B	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
C	01	A 00	B 00	C 00	D 00	E 00	F 00	G 00	H 00	I 00	01	01	01	01	01	01
D	01	J 00	K 00	L 00	M 00	N 00	O 00	P 00	Q 00	R 00	01	01	01	01	01	01
E	01	01	S 00	T 00	U 00	V 00	W 00	X 00	Y 00	Z 00	01	01	01	01	01	01
F	0 00	1 00	2 00	3 00	4 00	5 00	6 00	7 00	8 00	9 00	01	01	01	01	01	01

Table of Codes

Operators		Keywords		Operands	
0301	+	0C10	IF	03	external
0302	-	0C01	AT	04	hex
0203	*	0C02	CONNECT	05	integer
0204	/	0C03	RUN	06	location
0705	(	0C04	STOP	07	adcon
0806	)	0C05	DISPLAY	08	character
0607	=	0C06	DUMP	09	null
0608	>	0C07	SET	0A	$ Symbol
0609	<	0C08	COLLECT		
0A0A	:	0C09	PATCH		
040B	&	0C0A	QUALIFY		
040C	|	0C0B	DEFINE		
050D	¬	0C0C	CALL		
0D0E	Δ	0C0D	REMOVE		
0D0F	;	0C0E	DISCONNECT		
0C10	IF	0C0F	END		
0B11	,				
0912	. (offset)				
0913	$ID				
0914	%				
0915	$P				
0916	$L				
0917	$S				
0918	$T				
0919	subscript or ¢				
091A	$B				
091B	. (explicit qual.)				
091C	$PATCH.				
091D	$AT.				
091E	-(unary)				

Action Code Matrix

This publication uses the following letters to represent the arguments of the Action Code Matrix.

a = NAME
b = / *
c = + -
d = | &
e = ¬
f = <=>.
g = (
h =)
i = $OPERATORS
j = : %
k = ,
l = KEYWORDS
m = ;

A row represents the incoming source item; a column represents the last operator in the "push-down stack."

	a	b	c	d	e	f	g	h	i	j	k	l	m
a	4	1	1	1	1	1	1	0	1	1	0	1	1
b	4	4	2	2	2	2	2	0	4	2	0	2	2
c	4	4	4	2	2	2	2	0	4	2	0	2	2
d	4	4	4	2	4	4	2	0	4	4	0	2	2
e	4	4	4	2	0	0	2	0	0	0	0	2	2
f	4	4	4	2	2	4	2	0	4	4	0	9	2
g	0	2	5	5	5	5	5	0	5	5	0	5	2
h	4	6	6	6	6	6	6	0	6	6	0	0	2
i	4	2	2	2	2	2	2	0	4	2	0	2	2
j	4	2	2	2	2	2	2	0	2	4	0	2	2
k	0	8	8	8	8	8	7	0	8	8	8	8	2
l	4	4	4	4	4	4	0	0	4	4	0	4	2
m	4	4	4	4	4	4	0	0	4	4	0	4	3

(All number codes are in hexadecimal.)

where:

Code 0 indicates a syntax error.

Code 1 indicates that the source item operand is to be entered in the name string; STEP; WORK.

Code 2 indicates that the source item is an operator and is to be entered on the top of the operand stack; STEP; WORK.

Code 3 indicates that the source item is an "end" character. It is to be entered as the last item in the string; HOOKUP. Exit to Scan Control if an EOB; continue polish string construction if a semi-colon.

Code 4 indicates that the last operator on the top of the stack is to be unstacked and entered in the name string; WORK.

Code 5 indicates that the source item is a left parenthesis. It is compared with what preceded it in the source string. On the basis of this comparison a decision is made which will affect how far the reverse scan will come when a right parenthesis is found.

Code 6 indicates that the source item is a right parenthesis. Operators are removed from the operator stack and entered into the name string until a left parenthesis is encountered. If in "prefix mode," the preceding operator ("prefix"), which is either ".", "$L", "$S", "$T", "$ID", "C" (subscript), or "$P", is also removed from the operator stack and placed in the name string. Both parentheses are removed.

Code 7 indicates that the source item is a comma; the operator is a left parenthesis; STEP; WORK. This effectively ignores the source item.

Code 8 indicates that the source item is a comma; the operator is a keyword. Unstack all operators until a keyword is reached. If the keyword is AT:STEP; WORK. Otherwise, move the keyword into the name string; STEP; WORK.

Code 9 indicates that the source item may be an equal sign; the operator is a keyword. If the keyword is other than an IF, disregard the equal sign; STEP; WORK. Otherwise go to Code 2 routine.

For a detailed explanation of the construction and format of the polish string see the description of the Source to Polish module.

This appendix describes those TSSS control blocks, buffers, and tables either built by or used and maintained by the Language routines. These tables include the

Input Device Table,

Qualify Table,

Symbol Control Block,

AT Control Block,

Patch Control Block,

Symbol Tables,

AT Tables,

PATCH Table, and the

SP Symbol Table.

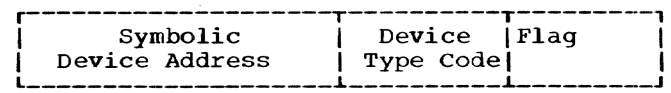
The table name in parentheses is the DSECT name. Generally, all the Language tables have two CSECTs for each DSECT, one each in real and virtual storage (see "Introduction" to the Appendixes).

Input Device Table (CHALCR)

When control passes from TSS/360 to TSSS via the Loader module (CEHBL), two contiguous words in storage having the same format are updated. These two words constitute the Input Device Table, which indicates to Language Control (CEHLC/CZHXC) the device type from which it is to receive input. A copy exists in both real and virtual storage, having different CSECT names.

TSSS may receive input from magnetic tape, cards, or a terminal. Language Control invites input from a terminal by causing a $ to be printed out, before it reads the terminal. A card reader or tape drive is merely read.

The format of the Input Device Table is as follows:



where:

Device Type Code may be X'00' for a terminal (1050, 2741, KSR35), X'01' for a 2540 card reader, or X'02' for a 2400 tape drive.

The Flag byte may be set:

Bit 0. When it is zero, the device is an SP terminal. When it is one, the device is a called device.

Bit 1. When it is zero, input must be read in. When it is one, the input is already in the buffer. (Register 1 points to the address of the input and contains the byte count.)

Bit 2. When it is zero there is no connected SP. When it is one there is an SP connected.

The construction and content of the second fullword of the Input Device Table is identical to the first fullword. It is used to store the primary system input device information when a CALL command is issued. The CALL Command Processor constructs a new first word for the table.

The modules which refer to this table are:

The RSS Loader (CEHBL) and the VSS Activate Interrupt Processor (CZHNV) which are responsible for initializing the table.

Language Control (CEHLC/CZHXC), which uses it and may restore it after a CALL command.

The CALL Command Processor (CEHKL/CZHYL), which designates a sequential device (called device) to replace the first word of the table.

The END, RUN, STOP, and DISCONNECT Commands Processors, which restore the first word of the table by referring to the second word, following a CALL command.

Buffers

There are two language buffers:

The language input buffer (CEHLCT/CZHXCT) is 256 bytes long, and is assembled with Language Control (CEHLC/CZHXC). Data from the input device is placed into CEHLCT/CZHXCT by the I/O System to form the input string.

The language work area buffer (CEHCAS/CZHPAS) is 4096 bytes long, and is assembled with the DISPLAY Command Processor (CEHKD/CZHYD). It is used by the language routines to manipulate data fields.

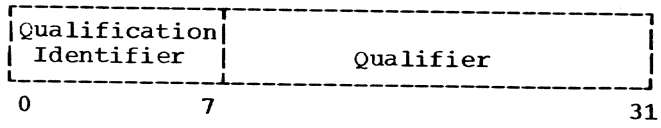
In addition, Real Core Access (CEHCA, CZHPA) uses a get/put buffer (CEHCAR, CZHPAR) on request of the language routines.

Qualify Table (CHAKQD)

This table establishes uniqueness for all references to elements of TSSS by the system programmer. For every symbol, some qualification is appended, either supplied explicitly with the symbol or taken from this table.

The Qualify Table is organized in two parts -- a Qualification Identifier (1 byte) and a Qualifier (2 or 3 bytes). The Identifier determines what interpretation should be placed on the remainder of the table.

The Qualify Table is one fullword and is formatted as follows:



where:

1. If the identifier is X'00', the table specifies real storage, using the Prefix Storage Area (PSA) addressable by the active CPU through prefixing hardware; and the qualifier is all zeros. These are the defaulted values for the Resident Support System.

2. If the identifier is X'01', the table specifies real storage, using the PSA of the designated CPU and bypassing the prefixing hardware; and the qualifier is the appropriate prefix value (or base) for the PSA.

3. If the identifier is X'02' with X'00' task ID, the table specifies virtual storage, and the qualifier is zero. This qualification is global and pertains to the shared virtual storage of all tasks. These are the defaulted values for the Virtual Support System. The defaulted option for the Resident Support System is the current task, as "global" has no real meaning in RSS.

4. If the identifier is X'03', the table specifies the private virtual storage of a designated task, and the qualifier is the designated task ID, which may be a number from X'01' to X'FFFF', as long as the task is attached to the system.

The Qualify Table is maintained by the Qualify Command Processor, and referred to by the majority of the Language Area modules, in particular Symbol Resolution (CEHMS, CZHWS).

The Symbol Control Block (CHAMSW)

The SCB is used to define a symbol and contains all the symbol attributes. It may also be used to resolve a literal. During Language Area processing, one SCB exists for each symbol or literal in the polish string. The SCB is built, maintained, and used by the entire Language Area.

The format of the SCB is shown in Figure 26. The SCB is aligned on a doubleword boundary.

The fields of the SCB represented in Figure 26 have the following meanings:

Length (MSWLEN, 4 bytes): - the number of bytes needed to accommodate an item.

Size (MSWSIZE, 4 bytes): - the total number of the storage elements associated with the symbol.

Work Area (MSWUNUS, 4 bytes): - a work area for various language routines.

Type (MSWTYPE, 1 byte) - a code designating the value as integer, hexadecimal, or character.

Type Attribute	Code
Hexadecimal	X'01'
Character	X'02'
Integer	X'03'

Classification - (1 byte): - a code designating a system symbol, SP symbol, external symbol, or literal.

Classification (MSWCLASS)	Code	Name
$	X'01'	MSWSYS
SP	X'02'	MSWSP
External	X'03'	MSWEXT
Literal	X'04'	MSWLIT
Immediate data	X'05'	MSWNAD

Flags (MSWFLAGS, 1 byte):
Bit 0 Undefined Symbol
Bit 1 Null SCB
Bit 2 Subject SCB
Bit 3 Data in work area
Bit 4 Physical Path
Bit 5 Cylinder # No
Bit 6 Track # desig. bit
Bit 7 Record # is on.

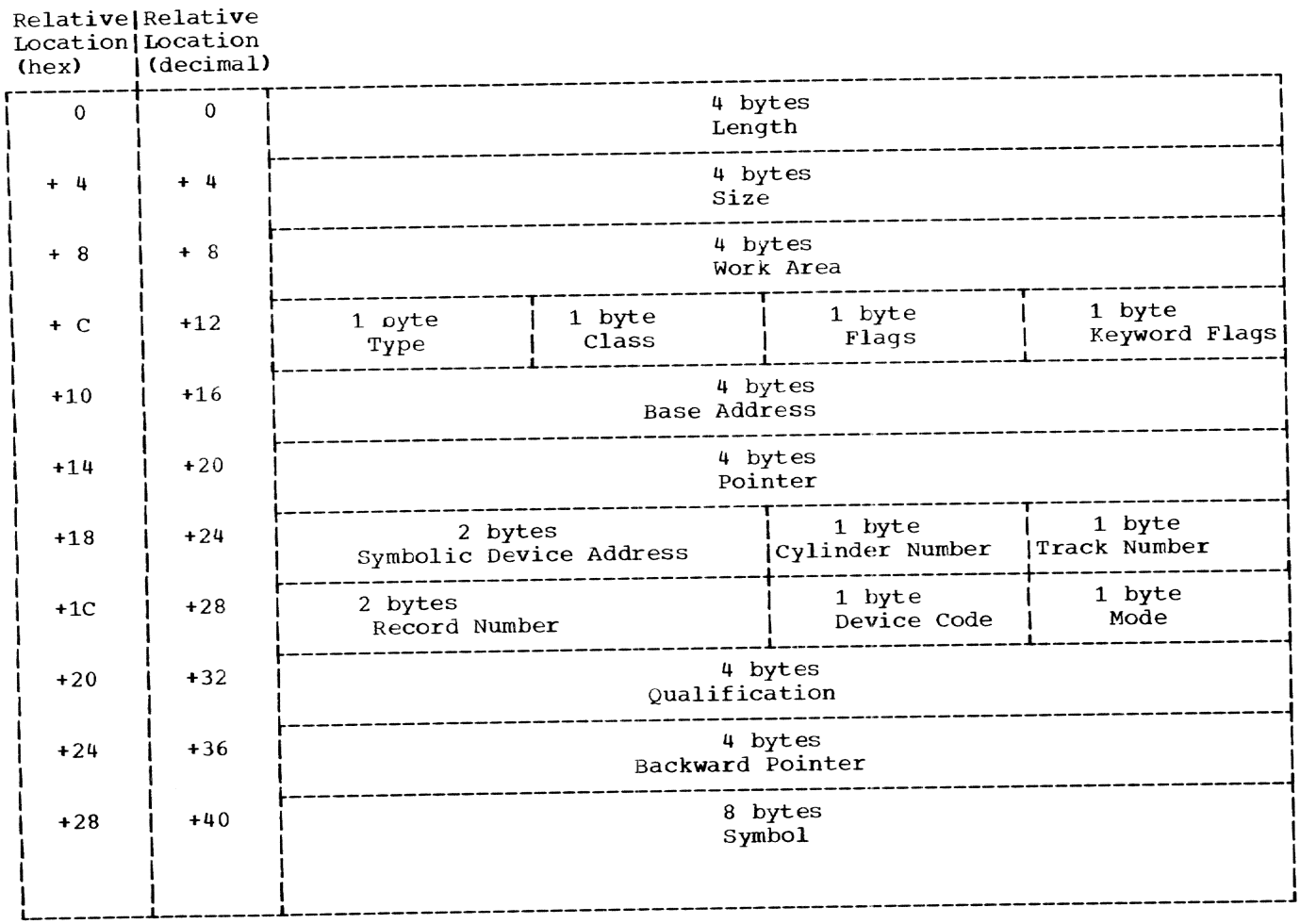


Figure 26. The Symbol Control Block

Keyword Flags (MSWBLNK2): - (1 byte):

Bit 0	More data to format dump
Bit 1	All of AT or Patch Table
Bit 2	Record over 4096 bytes
Bit 3	Set overflow condition
Bit 4	End of file condition from I/O
Bit 5	One byte of instruction in ACB
Bit 6	Two bytes of instruction in ACB
Bit 7	Same track indicator for DUMP/DISPLAY

Base Address (MSWBASE, 4 bytes): - the high order byte address of the storage area represented by the symbol.

Pointer (MSWPTR, 4 bytes): - an additive constant, used when computing an address, to indicate an element of an array.

Symbolic Device Address (MSWSDEV, 2 bytes): address of the device.

Cylinder # (MSWCYL, 1 byte)

Track # (MSWTRK, 1 byte)

Record # (MSWREC, 2 bytes)

Device Code in Hex (MSWDEVC, 1 byte): codes for devices used

X'00'	terminal
X'01'	2540 card reader
X'02'	2400 magnetic tape
X'03'	1403 printer
X'04'	2311 Disk Storage Drive
X'05'	2314 Storage Facility
X'06'	2301 Parallel Drum

Mode (MSWMODE, 1 byte) the channel command code, which determines density, parity, etc., of a tape. See Principle of Operations Manual for IBM 2400 Series Tape Drives, GA22-6866.

Qualification (MSWQUAL, 4 bytes): designation as to whether the data field resides in real storage, virtual storage, or exter-

nal storage; of which 1 byte = qualification code:

Qualification	Code
Real Core Unqualified	X'00'
Real Core Qualified	X'01'
Virtual Memory Unqualified	X'02'
Virtual Memory Qualified	X'03'
External Storage	X'04'

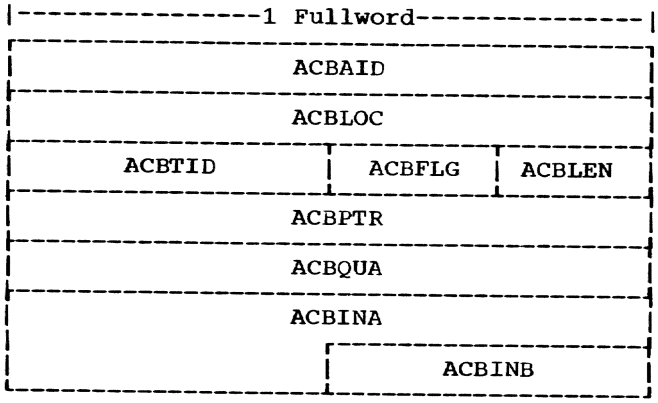
and 3 bytes = a field that contains either a prefix for real storage on a TASKID for virtual storage.

Backward Pointer (MSWBKPT, 4 bytes): used for system and SP symbols to point to the original SCB.

Symbol (MSWSYMB, 8 bytes): a string of alphabetic or numeric characters. The actual symbol.

Two special types of control blocks are used by TSSS Language -- the AT Control Block (ACB) and the PATCH Control Block (PCB). The Language Area uses these control blocks to maintain a record of information surrounding the implanted ATs and PATCHes. They are built by the AT Command Processor (CEHKA/CZHYA) and the PATCH Command Processor (CEHKP/CZHYP) and used by the Remove Command Processor (CEHKR/CZHYR), the AT SVC Processor (CEHJA/CZHZA), and other language modules.

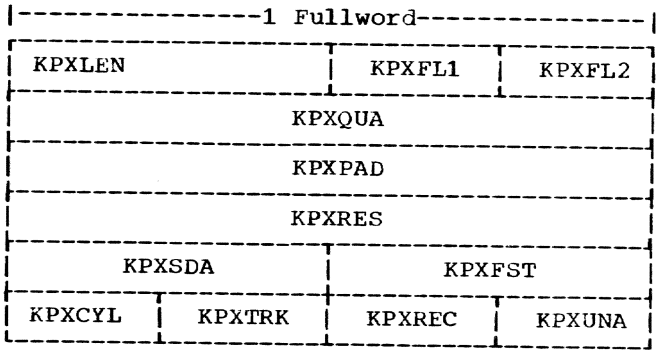
The ACB (CHAACB) is formatted as follows, aligned on a fullword boundary.



where:

ACBAID is a unique AT identifier
ACBLOC is the SVC location
ACBTID is the TSP task ID, if in VSS
ACBFLG is the flag byte
ACBLEN is the length of the text minus one
ACBPTR is the pointer to the command text
ACBQUA is the QUALIFY Table at the point of implantation
ACBINA is up to six bytes for the saved instruction;
ACBINB is first 2 bytes of the Qualify Table

The PCB (CHAKPX) is formatted as follows, aligned on a fullword boundary.



where:

KPXLEN is the length of the patch.
KPXFL1, KPXFL2 are unused.
KPXQUA is the Qualify Table at the time of implantation.
KPXPAD is the patched address.
KPXRES is the address of the saved restore data.
KPXSDA is the symbolic device address.
KPXFST is the offset of patch from beginning of record (2 bytes).
KPXCYL is the cylinder number (1 byte).
KPXTRK is the track number (1 byte).
KPXREC is the record number (1 byte).
KPXUNA is unused (1 byte).

The PCB and the ACB exist in both real and virtual storage, with one CSECT each (see "Introduction" to the appendixes).

The ACBs and PCBs are kept in tables as records. The remaining information in the AT and PATCH Tables completes the record of the implantation.

The AT Tables (CHAATB)

The ACB completely identifies the AT and contains the length of a pointer to the command text to be executed as a result of AT SVC execution. The AT Tables contain ACBs, as well as this command text. To conserve space the AT Tables are filled from lowest to highest address by ACBs, and from highest to lowest address by command text. Each of the AT Tables is one page long.

Each AT Table has a header of 2 doublewords, the first word of which is composed of a lock byte, an unused byte, and a half-

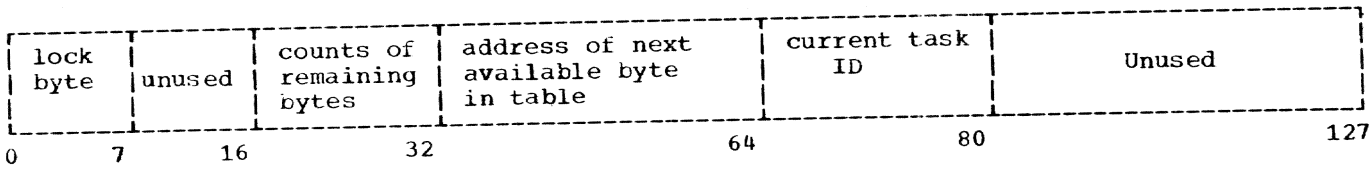
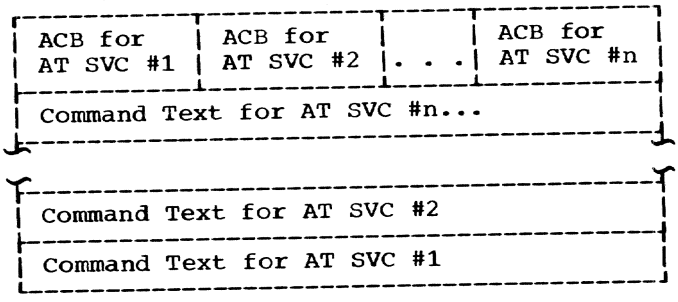


Figure 27. The AT Table Header

word containing the count of the remaining unused bytes in the table. The second word of the header is the address of the next available byte into which an ACB may be placed. The first two bytes of the second doubleword contain the task ID of the task whose TSSS routines are currently using the table. The remaining six bytes of the second doubleword are unused. The AT Table Header is shown in Figure 27.

The use of a lock byte in the AT Table is to guard against concurrent use by several VSS routines. If the table under consideration is locked, the VSS routine using the table forces time slice end for the task. If a VSS REMOVE Command Processor finds a locked table in which the task ID matches its own task ID, it recognizes the ABEND situation, ignores the lock byte, and removes the ACBs for the ATs implanted by the VSS routines for that task.

Overflow causes the current and subsequent AT commands to be ignored; an error message is written to the system programmer. The AT Tables are formatted as follows, exclusive of the header:



In RSS there is only one CSECT for CHAATB -- the MSP's AT Table (CHBATBR). In VSS there are two distinct CSECTs -- the TSP's AT Table (CHBATBVA), for which there exists a copy in every task, and the Shared Global AT Table (CHBATBVB), to which all tasks have access. The format for the ACB and the AT Table itself is identical for all three tables.

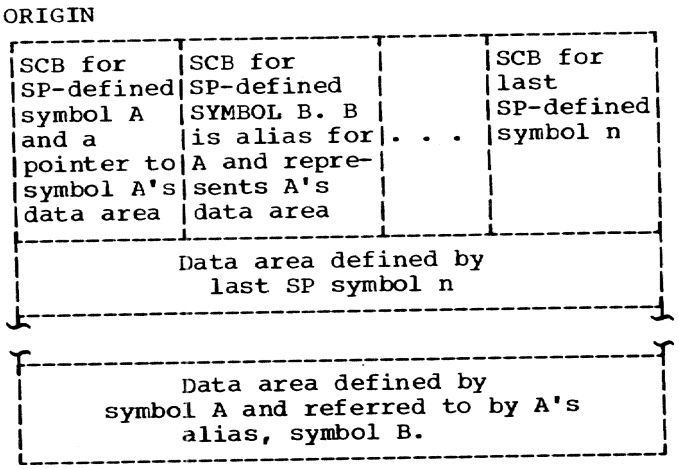
The Patch Table (CHAKPW)

The Patch Table serves the same function for the PCB that the AT Tables do for the ACB. The Patch Table is two pages long, but the placement and format of the PCBs and the patch restore data corresponds exactly to the ACBs and the AT Command text, respectively. The Patch Table has the same type of header that the AT Tables have. It has two CSECTs, one each in both real and virtual storage.

The SP Symbol Tables (CHASPM)

The SP Symbol Tables record each SP's privately defined symbols, both independent definition and alias establishment (see the DEFINE Command Processor). There are three distinct SP symbol tables, but they have identical formats. They are (1) the MSP Symbol Table (CHBSPMR), (2) the TSP Symbol Table (CHBSPMVA), and (3) the Global Shared Symbol Table (CHBSPMVB). Each symbol table is two pages (8192 bytes) long.

The Symbol Control Blocks (SCBs) that define the symbols are entered sequentially from the origin of a given Symbol Table. The data areas that the symbol defines are entered in reverse sequence from the end of the same tables. In alias establishment, an SCB is entered in the Symbol Table without allocating an additional data area for it. The data area which it defines has already been moved into the SP Symbol Table when the initial independent definition took place. The SP symbol tables have the same type of header as the AT and Patch tables. A sample SP Symbol Table is shown as follows:



APPENDIX F: THE SAVE AREAS

The tables included here, representing the status save areas, give the names of the major fields. For greater details see the program listings.

TSS/RSS Status Save Area (CHAESV)

This area is used by RSS to save the status of TSS/360 during RSS activation. It also contains certain control information necessary for RSS execution. TSS/360 status is restored from CHAESV at RSS exit.

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description
8	0	0	ESVCPSW	Current (Exit) PSW
8	+8	+8	ESVEPSW	External Old PSW
2	+10	+16	ESVEXCD	External Interrupt Code
8	+18	+24	ESVSPSW	SVC Old PSW (aligned on a doubleword boundary)
2	+20	+32	ESVSVCD	SVC Interrupt Code
8	+28	+40	ESVPPSW	Program Old PSW (aligned on a doubleword boundary)
2	+30	+48	ESVPMCD	Program Interrupt Code
8	+38	+56	ESVMPSW	Machine Check PSW (aligned on a double-word boundary)
2	+40	+64	ESVMKCD	Machine Check Interrupt Code
8	+48	+72	ESVYPSW	I/O Old PSW (aligned on a doubleword boundary)
2	+50	+80	ESVYYCD	I/O Interrupt Code
40	+58	+88	ESVxNPSW	New PSWs, where x= E=External S=SVC P=Program M=Machine Check Y=I/O
4	+80	+128	ESVCAW	Channel Address Word
8	+88	+136	ESVCSW	Channel Status Word (aligned on a double-word boundary)
4	+90	+144	ESVTSX	Pointer to Currently Active TSI
1	+94	+148	ESVNVRUN	Intervening RUN Flag Byte
3	+95	+149	ESVDVTSI	TSI Pointer from the Device Group Table (CHADEV)
2	+98	+152	ESVTSKID	Sending Task ID

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description
1	+9A	+154	ESVMASK	System mask to indicate 32 or 24 bit mode
1	+9B	+155	ESVLOCK2	Save area for Inter-CPU Communications lock byte
4	+9C	+156	ESVEWPGE	External Work Area Page Table entry pointer
4	+A0	+160	ESVTEPE	TSS External Page Table entry pointer
4	+A4	+164	ESVHTRTN	Return address for Halt and Transfer
4	+A8	+168	ESVMSPM	SP terminal information if RSS is activated by manual interruption
4	+AC	+172	ESVMSPCN	SP terminal information if connected
64	+B0	+176	ESVGPRS	General Purpose Registers (16)
32	+F0	+240	ESVFPRS	Floating Point Registers (4)
64	+110	+272	ESVCTRS	TSS Control Registers (15)
1	+150	+336	ESVACT	RSS Activation Flags
			ESVEXTR	External Interrupt (X'80')
			ESVTASP	TSSS Activation Sequence in Progress (X'40')
			ESVMSPA	MSP Attention (X'20')
			ESVSTUN	Symbol Table Unload in Progress (X'10')
			ESVRCAP	RSS Real Core Access in Progress (X'08')
			ESVOUHB	Other CPU halted bit (X'04')
			ESVITSI	Inactive TSI scan bit (X'02')
			ESVATXM	At Execution mode (X'01')
1	+151	+337	ESVSTAT	RSS Status Indicators
			ESVSTUP	Startup in Progress (X'80')
			ESVXITP	Exit in Progress (X'40')
			ESVLUNL	Load/Unload in Progress (X'20')
			ESVPXPR	Paging Exception Progressing (X'10')
			ESVDPCK	Deliver Program Check (X'08')
			ESVWRIT	Write Direct Received (Inter-CPU) (X'04')
			ESVEWTB	Error Wait Bit -- Inter-CPU (X'02')
			ESVTLIP	Symbol Table Load in Progress (X'01')
1	+152	+338	ESVSTAT1	RSS Status Indicator Byte 2
			ESVPROI	Loader indicator to read only (X'80')
			ESVRNAT	RUN with operand Indicator (X'40')
			ESVVMAP	RSS VM Access in Progress Indicator (X'20')
			ESVTRWI	TAM Read/Write Indicator (X'10')
			ESVQXT	External interruption pending (X'08')
1	+153	+339	ESVCPU	CPU Identification
24	+154	+340	ESVSADT	Save Area for Second Entry of SADT for RSS Dynamic Load Procedures

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description
28	+16C	+364	ESVUSRIO	Save Area for User Portion of SIORCB during Intervention Required
4	+188	+392	ESVFLGIO	I/O Flags stored on Intervention Required
4	+18C	+396	ESVRAMIO	Saves the SADT Entry Pointer on Intervention Required
12	+190	+400	ESVRECIO	Saves the SSDAT on Intervention Required
8	+19C	+412	ESVSAPIO	Save Area Pointers
8	+1A4	+420	ESVCSWIO	CSW save area during Intervention Required
8	+1AC	+428	ESVCAWIO	CAW save area during Intervention Required
	+1B4	+436	END CHAESV	

TSS/VSS Status Save Area (CHAEVS)

This area is used to save TSS status upon activation of VSS. It is also used as an internal communications area by VSS. (CHAISA is saved in a special PSECT by itself -- CZHPSR.)

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description
8	0	0	EVSCVPSW	Current VPSW
56	+8	+8	EVSxVPSW	Old VPSWs, where x = S (SVC) E (External) A (Asynchronous I/O) T (Timer) N (Synchronous I/O) P (Program-PPSW) D (Recoverable Data Set Paging Error VPSW-PPSW)
8	+40	+64	EVSVVPSW	Old VPSW
8	+48	+12	EVSPNPSW	New Program VPSW
56	+50	+80	EVSxNPSW	New VPSWs, where x = S (SVC) E (External) A (Asynchronous I/O) T (Timer) N (Synchronous I/O) R (Recover Data Set Paging) V (VSS)
64	+88	+136	EVSGPRn	General Purpose Registers, where n = 0 through 15
32	+C8	+200	EVSFPRS	Floating Point Registers
64	+E8	+232	EVSCTRS	Control Registers
1	+128	+296	EVSMODE	VSS Status Indicators
			X'80'	VSS Active (EVSACT)
			X'40'	TSP Attention Received (EVSTSPA)
			X'20'	TSP Connected (EVSTSPC)
			X'10'	One Terminal Case (EVSTRM1)
			X'08'	Two Terminal Case (EVSTRM2)
			X'04'	Dynamic Mode (EVSDYNA)
			X'02'	Conversational Mode (EVSCONV)
			X'01'	RUN with operand indicator (EVSRNAT)
1	+129	+297	EVSMOD1	VSS State Indicators
			X'80'	VSS in activation sequence (EVSASEQ)
			X'40'	VSS Virtual Memory in process (EVSVMAP)
			X'20'	VSS TAM Read/Write Indicator (EVSTRWI)
			X'10'	
			X'08'	
			X'04'	
			X'02'	
			X'01'	

Field Length (bytes)	Relative Location (hex)	Relative Location (decimal)	Field Name	Field Description
The following is a VSS SADT compatible with the SADT in CHASYS				
2	+130	+304	EVSRPP	Physical path
2	+132	+306	EVSRFL	Flags
4	+134	+308	EVSRCB	Pointer to SIORCB
8	+138	+312	EVSRCS	Channel Status Word (CSW)
8	+140	+320	EVSRPS	Program Status Word (PSW)
24	+148	+328	--	Second RSS I/O Device Entry
28	+160	+356	EVSUSRIO	Save Area for User portion of SIORCB during Intervention Required
4	+17C	+380	EVSFLGIO	I/O flags stored on Intervention Required
4	+180	+384	EVSRAMIO	Saves SADT entry pointer on Intervention Required
12	+184	+388	EVSRECIO	Saves SSDAT on Intervention Required
8	+190	+400	EVSSAPIO	Save area pointers
8	+198	+408	EVSCSWIO	Saves CSW on Intervention Required
8	+1A0	+416	EVSCAWIO	Saves CAW ON Intervention Required
The following defines the save area for the MCB passed during VSS activation				
1	+1A8	+424	EVSLNG	Message length in doublewords
1	+1A9	+425	EVSCOD	Flag byte
1	+1AA	+426	EVSRCD	Return code
1	+1AB	+427	EVSCD1	MCB message code
2	+1AC	+428	EVSSVC	XSEND SVC
2	+1AE	+430	EVSSPR	Spare space
2	+1B0	+432	EVSSND	Our task ID (EVSTSKID EQU EVSSND)
2	+1B2	+432	EVSRCV	Receiving task ID
4	+1B4	+434	EVSECB	Address of Event Control Block
0	+1B8	+438	EVSTXT	Message text
2	+1B8	+438	EVSSDA	Symbolic Device Address
2	+1BA	+440	EVSSRSS	RSS/VSS Indicator. RSS=X'FF' VSS=X'00'
	+1BC	+442	ENDCHAEVS	

Each of the SVC codes processed by TSSS is shown below, accompanied by (1) the module responsible for implanting or remotely executing it, or both; (2) the request or operation it represents, (3) the module or modules that processes it after the SVC Interrupt Processor has completed its processing; and (4) the result. (SVC codes 74-79 and 86-94 are reserved.)

SVC code	Implanted/ Executed By	Indicated Request or Operation	Processed By (sequentially)	Result
65	CZHPA (I,E)	Get real storage for VSS	1. CEHDR 2. CEHCA	Requested page of real storage in VSS paging buffer
66	CZHPA (I,E)	Put real storage for VSS	1. CEHDR 2. CEHCA	Contents of VSS paging buffer are stored on external device of real page
67	CEHJA (I only)	AT SVC execution in real storage completed	1. CEHDR 2. CEHJA	Control returned to the program interrupted by AT SVC execution
68	CEHJA (I only)	RSS AT SVC execution in virtual storage complete	1. CEHDR 2. CEHJA	Control returned to the program interrupted by AT SVC execution
69	CEHJA (I only)	Execute AT SVC in real storage for RSS or VSS	1. CEHDR 2. CEHJA	Command string for AT SVC becomes input to CEHLC
70	CZHYA,CZHYR (I,E)	Process VSS-supplied command string	1. CEHDR 2. CEHLC	SVC 69 implanted in real storage for VSS or specified $AT removed.
71	CEHKA (I only)	Execute AT in private virtual storage for RSS	1. CEHDR 2. CEHJA	Command string for AT SVC becomes input to CEHLC
72	CEHKA (I only)	Execute AT in shared virtual storage for RSS	1. CEHDR 2. CEHJA	Command string for AT SVC becomes input to CEHLC
73	CZHPB (I,E)	Determine if input VM page is shared	1. CEHDR 2. CEHCB	Shared indication in routine's registers.
80	CZHYA (I only)	Execute AT in private virtual storage for VSS	1. CEHDA 2. CZHNV 3. CZHZA	VSS is activated, code 2; AT command string becomes input to CZHXC
81	LOGON1 (I,E)	LOGON MSP or TSP as indicated in MCB	CEHDL	If valid LOGON attempt, SP is connected, logged-on
82	CZHPR (I,E)	Deactivate VSS	CEHDE	VSS deactivated as requested; control returned to TSS/360
83	VSS Command	Activate VSS	CEHDV	VSS activated, code 1;TSP connected: one terminal case
84	CZHZA (I only)	VSS AT SVC execution in virtual storage completed	1. CEHDA 2. CZHNV 3. CZHZA	VSS is activated, code 4; control is then returned to program interrupted by AT execution
85	CZHYA (I only)	Execute AT is shared virtual storage for VSS	1. CEHDA 2. CZHNV 3. CZHZA	VSS is activated, code 6; AT Command string becomes input to CZHXC

APPENDIX H: TSSS FIELDS IN TSS/360 TABLES

Five TSS/360 tables are used in connection with TSSS operation:

- The Interrupt Storage General Queue Entry (CHAGQE)
- Prefix Storage Area (CHAPSA)
- Interrupt Storage Area (CHAISA)
- System Table (CHASYS)
- Task Status Index (CHATSI).

These tables, initialized and maintained by TSS/360, are used by TSS/360 during TSSS activation or by TSSS during its operation. Three of the tables have special fields used only by or for TSSS, but initialized by TSS/360. This appendix lists the fields in CHAISA, CHASYS, and CHATSI that are created for TSSS.

CHAISA

Field Name	Size (bytes)	Purpose	Rel. Loc. (hex)
ISANV	8	New VSS VPSW	+838
ISAEF		VSS Active Flag	+846
ISAVSC		VSS Connected Flag	+844

CHASYS

Field Name	Size (bytes)	Purpose	Rel. Loc. (hex)
SYSRSS	1	RSS Active.	+14C
SYSRSM	EQU	X'80' RSS Active Mask	
SYSRCT	4	RSS Communication Table Address	+150
SYSRS1	8	LPSW to Enter RSS via Program Interrupt	+158
SYSRS2	8	LPSW to Enter RSS via SVC Interrupt	+160
SYSRS3	8	LPSW to Enter RSS via I/O Interrupt	+168
SYSRS4	8	LPSW to Enter RSS via External Interrupt Key	+170
SYSRS5	8	LPSW to Enter RSS via Channel Interrupt Processor	+178
SYSRS6	8	LPSW to Enter RSS via Queue GQE on TSI	+180
SYSRIO	48	RSS System Active Device Table (SADT) -- two entries	+188

CHATSI

Field Name	Size (bytes)	Purpose	Rel. Loc. (hex)
TSIVSS	1	Flags to maintain a record of VSS use of the TSI	+86
TSIVS	EQU	TSIVSS VSS Active Flag. X'80' is mask	
TSIVT	EQU	TSIVSS TSP Connected Flag. X'40' is mask	
TSIVU	EQU	TSIVSS Separate TSP Terminal Flag X'20' is mask	
TSIVTP	4	VSS Alternate TSI Pointer	+80
TSISDA	2	Symbolic device address of the TSP terminal	+84

APPENDIX I: MESSAGES BY MODULE USAGE

This appendix lists the TSSS modules with the message class codes and message numbers that each module issues. The modules are arranged in alphabetic order.

The input parameter to the message routines, the Message Control Word, is formatted as ccyyxx, where cc is a unique two-character module identifier in EBCDIC, yy is the message number in hexadecimal, xx is the message class code in hexadecimal. However, the output format for a message is CEHccxyy, where CEH is a constant (CZH in VSS), cc is the two character module identification, x is the message class code, and yy is the message number. The message class codes and message numbers are listed here in the external output format (for example, message class 02, message number 04, appears as 204).

Module ID	Message Class and Number
CEHAC	none
CEHAD	none
CEHAE	306, 307
CEHAP	118, 10A, 20D, 308
CEHAQ	none
CEHAS	none
CEHBD	none
CEHBE	none
CEHBL	303, 304, 305
CEHBT/CZHRT	202, 209, 20A, 20B, 20C
CEHBU	300, 301, 302
CEHCA	117, 118
CEHCB	10A, 204
CEHCC	none
CEHCF	109
CEHCH	none
CEHCM	none
CEHCQ	none
CEHCS	none
CEHDA	none
CEHDE	none
CEHDL	none
CEHDR	201

Module ID	Message Class and Number
CEHDV	none
CEHEA/CZHSA	000, 001, 002, 004, 005, 00C, 00F
CEHEB	003, 009
CEHFA/CZHTA	000, 001, 002
CEHFB/CZHTB	000, 002, 010
CEHFC/CZHTC	000, 001, 002, 00A
CEHFD/CZHTD	000, 001, 002, 010
CEHFE/CZHTE	none
CEHGA/CZHUA	006
CEHGB/CZHUB	006
CEHGC/CZHUC	006
CEHGD/CZHUD	006
CEHGE/CZHUE	006, 007
CEHHA/CZHVA	none
CEHJA/CZHZA	200, 201, 208, 20E
CEHJF/CZHZF	101, 102, 10A, 10E
CEHKA/CZHYA	106, 107, 10E, 114, 119, 11A
CEHKC/CZHYC	106, 107, 122, 124
CEHKD/CZHYD	105, 106, 107, 10D, 10E, 11C, 121, 128, 206
CEHKE/CZHYE	103, 107, 108
CEHKL/CZHYL	105, 11D, 11F
CEHKM/CZHYM	107

Module ID	Message Class and Number
CEHKN/CZHYN	106, 107
CEHKP/CZHYP	106, 107, 115, 116
CEHKQ/CZHYQ	107, 109, 10C, 10E
CEHKR/CZHYR	106, 107, 10E, 118, 11E
CEHKS/CZHYS	000, 106, 107, 10D, 10E, 123, 124, 206
CEHKT/CZHYT	107
CEHKW	104, 205
CEHLA/CZHXA	100, 105, 106, 107, 109, 10B, 10D, 10E, 10F, 118, 120, 124, 125, 126, 127, 20D
CEHLC/CZHXC	106, 114, 121
CEHLL/CZHXL	111, 124, 106
CEHLP/CZHXP	111, 112, 113, 11B
CEHLS/CZHXS	205,
CEHMA/CZHWA	10B
CEHMM/CZHWM	121
CEHMS	112
CZHNE	none
CZHNM	none
CZHNP	10A
CZHNV	none
CZHPB	none
CZHPR	none
CZHPS	none
CZHSB	006,007,008,00B
CZHWS	112

GLOSSARY

For time-sharing terms not defined here, see the glossary in IBM System/360 Time Sharing System: Concepts and Facilities, GC28-2003.

abort case: When TSS/360 is considered to be aborted, as the result of the Inter-CPU error wait state, the abort case allows the use of a special subset of RSS capabilities only, restricting the MSP from the AT, CONNECT, DISCONNECT and RUN functions. (See the RSS External Interrupt Processor module description.)

activation: Activation is a series of operations that make RSS or VSS capabilities logically available to the Systems Programmer, either via his intervention or via execution of an AT he previously caused to be implanted. Activation includes saving all status pertinent to the interrupted program; suspending execution of TSS/360 or, in VSS, the subject task, and entering RSS or VSS execution mode. Initial activation includes connecting the SP.

AT mode: AT mode is established by setting a flag in the RSS Status Save Area (ESVARXM), which occurs when an AT SVC is executed (causing activation of TSSS). During AT mode processing, Language Control initiates processing of a stored dynamic statement instead of reading an input device.

call mode: Call mode is established through the use of the CALL command, which causes the active entry in the Input Device Table to be changed to contain the address of a card reader or tape drive instead of the system programmer's terminal. Language Control initiates the reading of a sequential data set from this called device, until a termination command is reached or the SP terminates the data set. Call mode can be concurrent with AT mode.

chained ATs: A chained AT is one whose AT Control Block (ACB) shares the same implanted AT SVC with one or more other ACBs. Execution of the SVC causes the chained AT command strings to be executed sequentially. An X'80' setting in the flag byte (ACBLOC) of an ACB indicates that chaining has been activated for that ACB.

command: A TSSS command is the syntactical unit that specifies a TSSS operation. While it may be synonymous with keyword, command generally includes the operand for a given keyword.

command statement: A command statement generally includes more than one command; it is used synonymously with the terms input string and command string.

connect, connection: Connection denotes MSP or TSP capability at a terminal. It implies that RSS or VSS was successfully invoked and that the disconnect function has not been performed for the connected user. Connection is the result of initial activation. This condition is represented by various bit settings, which are tested by the appropriate routines.

conversational mode: Conversational mode is established when input is solicited from the system programmer's terminal. The SP terminal is the primary input device; its symbolic device address is initialized in the Input Device Table by the Environment area module responsible for RSS or VSS activation.

control nucleus: The control nucleus is the resident portion of TSSS and is responsible for processing interruptions and activating or deactivating either RSS or VSS. It is loaded with the TSS/360 Supervisor at TSS/360 Startup.

current PSW: The current (or exit) PSW is the TSS/360 old PSW, which was stored when TSSS interrupted the system, and which designates the address where control is to be returned when TSSS is deactivated. This PSW is stored by TSSS in the first eight bytes (ESVCPSW) of the TSS/RSS Status Save Area.

deactivation: TSSS deactivation is a series of operations that restore control to TSS/360 when TSSS is active. Deactivation is initiated by a RUN or DISCONNECT command; it includes restoring all saved TSS/360 or task status, disconnecting the SP, and transferring control to TSS/360. Deactivation in RSS also includes restarting the halted CPU.

dedicate, dedicated: A terminal at which an SP is connected is dedicated to RSS or VSS when RSS or VSS, respectively, is executing, in that no other program can then be executed at the terminal.

disconnection, disconnect function: An SP signals that he wishes to end his terminal session by the DISCONNECT command. This results in deactivation, as well as restoring the TSSS load tables to their status prior to TSSS activation. All ATs belong-

ing to the SP are removed as a result of the disconnect function; all SP-implanted patches remain, although the record of them in the Patch Table is destroyed when the table is reinitialized.

dynamic statement: A TSSS dynamic statement is the command or command statement accompanying an AT. It is stored in the AT Table with the ACB, which contains a pointer to the statement.

"get": In TSSS a "get" is moving the contents of a designated page from an address in real storage, or from a location on an external device, into a buffer. Opposite of "put."

global qualification: TSSS global qualification is specified by the SP and exists when the Identifier field of the Qualify Table is X'02', and the Qualifier field of the Qualify Table is X'000000'. If an AT with global qualification (recorded in the ACB) is implanted in a way that causes a task other than the parent task to encounter and execute the SVC, every encountering task will also execute the stored dynamic statement. TSSS need not have been previously activated within the encountering task.

Inter-CPU Error Wait State: The Inter-CPU Error Wait State exists when the Inter-CPU Error Wait bit (ESVEWTB) is set on (X'02') in the TSS/RSS Status Save Area. This error wait state occurs if the CPU that is subject to a Halt and Transfer order (via Write Direct) does not respond within one second, whereupon the timer causes an external interruption. After the Inter-CPU Error Wait State has been established, it may be terminated only by a manual key external interrupt, at which time the abort case is established.

keyword: Each TSSS command name except IF constitutes a TSSS keyword, which is the internal designation for an executable operator portion of a command. TSSS recognizes 16 keywords, each of which invokes a keyword execution subroutine.

local MSP: If an MSP connects to RSS by pressing a CPU interruption key, which preempts the operator's terminal and temporarily dedicates it to RSS, he is called a local MSP.

parent task: If VSS, activated within a given task, implants an AT SVC during VSS execution, this task is called the parent task with regard to the implanted AT SVC.

polish, polish string: TSSS converts the input string into an internal format known as polish notation; the converted data is called a polish string. TSSS uses Early Operator Reverse Polish.

private qualification: Private qualification is used to denote non-global qualification for an AT. If an AT SVC with private qualification is implanted in shared virtual storage, other tasks will encounter and execute it. Only the parent task will execute the stored dynamic statement.

"put": In TSSS "put" is returning the contents of a given buffer to the location from which it was fetched with a "get" function. Opposite of "get."

reactivation: Reactivation is the activation of either RSS or VSS when the SP is already connected, but has relinquished control to TSS/360 or to the subject task with the RUN command.

restart: Restart is two sequential manual key external interruptions from the CPU executing RSS. This manual intervention may be the result of an apparent error condition in RSS. Restart results in dumping TSS/360 real storage, and unloading and completing reloading of transient RSS. Processing then may continue as for any activation of RSS.

run mode: Run mode is the state of TSSS when TSS/360 (for VSS, the task) is executing but a system programmer is still connected. Run mode is the alternative to the TSSS execution modes -- conversational, call, and AT modes.

SADT: The TSSS Active Device Table (SADT) is part of the TSS/360 System Table (CHASYS) for RSS, and the TSS/VSS Status Save Area (CHAEVS) for VSS.

SCB: A Symbol Control Block (SCB) is (1) a 48-byte data field in the SP's working storage containing an SP symbol or a system symbol, symbol-defining information, and other related data; (2) a data field used to pass information between language area routines, in the same format as (1) and often a copy of such an SCB.

SP symbol: An SP symbol is a symbol created by the system programmer with the DEFINE command. The definition is recorded in an SCB, which is stored in the SP's Symbol Table or, for a TSP, in the Global Symbol Table.

system symbols: TSSS system symbols are a group of 23 fixed symbols which begin with $. These system symbols allow symbolic reference to certain data fields or perform certain functions when used as command operands or within operands. The system symbols $B, $P, $L, $S, $T, and $ID are

defined as system symbols externally; they are treated as operators internally.

VCA: A TSSS Virtual Communications Area (VCA) is a concept used primarily for convenience on the TSSS flowcharts. A VCA is any area that is used to pass information from VSS to RSS, via a Load Real Address instruction. For example, the VSS Status Save Area becomes a VCA when VSS restores task status prior to exiting.

void command: The void command consists of an input string containing only end-of-block characters (VSS only). Its execution results in deactivation of VSS with an Attention interruption queued to the task. Its purpose is to permit a TSP at the task's SYSIN terminal to communicate with the TSS/360 Command Language Interpreter; in run mode (when TSS/360 is active) he presses Attention (activating VSS) followed by the void command (RETURN).

VSS command: The VSS command is a TSS/360 command that causes an SVC 83 to be issued; VSS is activated within the current task, preempting the SYSIN terminal and establishing the one-terminal case.

Where more than one page number is given, the major reference is first. An asterisk indicates that this entry is also a glossary entry.

Time Sharing Support System Printed in U.S.A. GY28-2022-2

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]