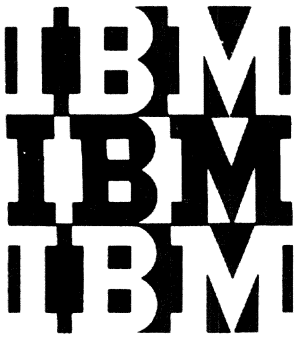


File No. S360-36
GY28-2013-6

Program Logic

Version 8.1

IBM System/360 Time Sharing System

Command System

Provides descriptions of the internal logic of each Command System module for persons responsible for program maintenance and modification. The Command System consists of modules that process the commands available to the user, operator, manager, and administrator of the System/360 Time Sharing System.

Provides the following information for each module: basic function, entries, modules called, method of operation, exits, error conditions, system control blocks used, and a flowchart.

Before using, be familiar with the contents of:

IBM System/360 Time Sharing System:

Concepts and Facilities, GC28-2003

Command System User's Guide, GC28-2001

Manager's and Administrator's Guide, GC28-2024

Operator's Guide, GC28-2033

Have available for reference:

IBM System/360 Time Sharing System: System Messages, GC28-2037

Seventh Edition (September 1971)

This is a major revision of, and makes obsolete, GY28-2013-5 and Technical Newsletter GN28-3164. This edition documents the new KEYWORD, CHGPASS, FLOW, MCAST, and MCASTAB commands, as well as a set of attention response commands -- EXIT, PUSH, RTRN, STACK, and STRING. Significant changes will be indicated by a vertical line beside the changed text.

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.

Requests 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 IBM Corporation, Time Sharing System/360 Programming Publications, Department 643, Neighborhood Road, Kingston, New York 12401

This manual, a guide to the logical structure and functions of the TSS/360 Command System, is designed to be used with the individual assembly listings; it does not discuss the machine-language module structure. The symbols that appear here correlate with those used in the listings.

This manual is intended for persons involved in program maintenance and modification.

The manual has 8 sections. Section 1 introduces the Command System. Sections 2 through 7 describe the logical divisions of the system; Command Controller, Text Editor, command routines, macro instructions, source language processing, and interruption processing; Section 8 contains flowcharts for the command system modules. The manual also includes two appendixes: System control block usage and a copy of SYSPRO.

Each description of a command system module includes:

- Common and symbolic module names.
- All entry points to the module, and the use of each entry point.
- Basic module functions. (Unless otherwise noted, modules will operate in conversational and nonconversational modes.)
- All module exits. In most cases, the standard exit is a return to the calling routine, via the RETURN macro instruction.
- All modules called and the reason for each call. If the called module is within the command system, its common and symbolic module names will be listed; for other modules, the common name and symbolic name, if applicable, are given.
- An explanation of module operation, with a discussion of the principal logic paths.
- All error conditions recognized by a module, and the actions taken for each condition.
- All system control blocks referred to by a module.
- Flowcharts.

To use this publication you need the information in:

IBM System/360 Time Sharing System:

Concepts and Facilities, GC28-2003

Command System User's Guide, GC28-2001

Manager's and Administartor's Guide, GC28-2024

Operator's Guide, GC28-2033

If you intend to use this book to debug programs, you need the following reference publication:

IBM System/360 Time Sharing System: System Messages, GC28-2037

ILLUSTRATIONS

COMMAND SYSTEM FUNCTIONS

The command system has three basic functions:

1. Receive and interpret commands submitted by users of the time-sharing sharing system.
2. Carry out the actions requested by each valid command.
3. Inform the user of the results of the actions performed or, if a command is invalid, inform him of his error so that he may correct it.

The command system has been designed to meet a number of major objectives that include facilities to:

- Provide the user with the ability to define his own commands.
- Permit the user to call his programs or procedures by direct commands.
- Standardize command syntax.
- Provide default and profile facilities.
- Provide a general editing capability.

The command system accepts all commands described in Command System User's Guide, Manager's and Administrator's Guide, and Operator's Guide (Full titles and order numbers of these publications are shown in the preface.) Commands may be accepted directly from a user at his terminal, or indirectly from a set of commands previously stored in the time-sharing system. The command system also processes a number of macro instructions that may be employed in user object programs; the expansion of these macro instructions results in object code that will link to the appropriate command system module at execution time.

The command system also contains a collection of service routines, almost all of them privileged, which operate in virtual storage under the control of the task monitor.

Each TSS/360 task includes a task monitor and some portion of the command system. The specific modules that appear in the task depend, of course, on the actions requested in that task. The task monitor handles all interruptions, and calls upon the command system as needed. In turn, the command system calls upon the TSS/360 general service, catalog service, and access method routines; it also requests system functions through supervisor calls, such as VSEND and TSEND. All actions performed by the resident supervisor, such as paging and time slicing, do not influence the command system operation.

COMMAND SYSTEM ORGANIZATION

Command Controller

The command controller consists of the Command Analyzer and Executor as the highest control routine; to this routine is added other routines to handle procedures, defaults, synonyms, and comments. Also, the lan-

guage processor controllers and text editor are driven by the command controller.

The basic control of the command controller is done by the Command Analyzer and Executor. The Command Analyzer and Executor is invoked upon initial task entry. It may also be invoked recursively by a user, upon intervention of an object program, or during language processor control. This routine calls the Verb Scanner to determine the verb type from the control dictionaries. The verb operands are translated and a command routine is called to execute the verb.

Normal procedures are handled by the Procedure Expander, which scans for procedure parameters, retrieves the procedure from the procedure library, and performs character substitution of the parameter specification for each occurrence of the dummy parameter in the procedure model.

Text Editor

The Text Editor commands help make it possible for data sets to be created and maintained. Lines of data may be edited within an existing VISAM data set or as they are entered into a VISAM data set. The User Control routine provides the principal interface between the Text Editor routines and the remainder of the command system, as most Text Editor routines operate in a nonprivileged state. The Edit Controller, which is external to the text editor (but also nonprivileged), provides an interface with the other modules in the system. All Text Editor commands are BUILTIN-created; each BPKD points to a common entry point located in the User Controller. After locating the BPKD, User Control supervises the execution of the command.

Command Routines

The largest part of the command system consists of command routines, almost all of which are privileged BUILTINs. Typically, each of these processes a single command or macro instruction by calling general service routines and other command routines, in a prescribed order, to carry out the requested actions. The operation of the command routines is basically the same: They are called by the Command Analyzer and Executor (or by another command routine or macro instruction); they make use of routines in the command controller to get input and issue output; they call general service routines as needed; and, when finished, they return to their calling routine. For an external description of the commands, consult the appropriate guide (listed in the preface). The command routines are contained in Section 4 with the exception of the PUNCH, PRINT, WT, RT, ASNBD, FORCE, HOLD/DROP, MSG/BCST, REPLY, SHUTDOWN, and XWTO routines. These routines are described in the Operator Task and Bulk I/O PLM, GY28-2047.

Several modules that perform support functions for the command routines are also described in Section 4.

Macro Instruction Processing

The command system handles many macro instructions. In general, expansion of those macro instructions produces object coding that links to a command system routine. Usually, the linkage is to a special macro instruction entry point in the routine so that the routine will not issue messages, but instead will return a code to show the processing results. Some command system modules (for example, those used to access SYSIN and SYSOUT) are linked to only by macro instructions. Other command system routines may be called by macro instructions and commands. See Section 5.

Source Language Processing

Three command system modules, LPCMAIN, GETLINE, and PUTDIAG, serve as an interface between the language processors (FORTRAN compiler, assembler, and linkage editor) and the user, or his prestored source program. These modules get parameters for the language processors, get source lines, and pass language processor diagnostics to SYSOUT. See Section 6.

Interruption Handling

The command system includes several modules that are entered by the task monitor for the following types of interruptions: the external interruption caused by intertask messages; some program interruptions; and the initial attention interruption of a conversational task. See Section 7.

COMMAND SYSTEM MODULE INTERACTIONS

Command system modular interaction is most easily understood as a function of time. Thus, the figures shown in this section are developmental, tracing command system interaction during the execution of specific functions. The first two are initialization functions:

Figure 1 - Explanation of the LOGON Procedure
Figure 2 - Initial Command Entry Procedure

The command entered may be a BUILTIN call (such as CDS or EDIT), a textual procedure, or a "direct call." (This last could be a dynamic statement or a LOAD, UNLOAD, or parameterless CALL command.) Figures 3, 4, and 5 demonstrate the command system's general operation in these cases. Figure 6, the last figure in this section, presents the overall linkage of command system modules.

The command system contains two centers of information, the source list (SL) and the combined dictionary. All commands in the system are read from the source list. The combined dictionary is the source of the names of all procedures, synonyms, defaults, and command variables.

The source list is a list of pending transactions. When the procedure capability was added it became possible to stack commands. The use of a procedure verb will result in the procedure expander placing more verbs in the source list. Execution continues until all of the verbs in the source list have been processed.

COMMAND SYSTEM OPERATION

The following paragraphs describe the general operation of the command system. The description begins with the command system operation at system startup, and continues to system shutdown.

System Startup

At system startup, command system initiates the main operator's task, and peforms the following prefatory functions: opens the operator's log, initiates the batch monitor task so that it can start operating, and readies the system for normal time-sharing operations.

Conversational Task Initiation

When a user causes an initial ATTENTION at his terminal, by pressing tne attention key or by dialing the telephone number to initiate conver-

sational operations, the resident supervisor establishes a task for him. The task monitor links to the command system's Initial Attention Interrupt Processor, which performs a variety of task initialization functions; these include definition of the system library and other system data sets, and identification of the user's terminal as the SYSIN and SYSOUT for the task. The LOGON command routine is then called to start conversational processing by prompting the user for his credentials.

Nonconversational Task Initiation

The user has three ways to start nonconversational tasks: he may issue an EXECUTE or BACK command from his terminal after he has logged on; he may issue a bulk input/output command (PUNCH, WT, or PRINT); he may submit a punched sequence of commands to the operator who will input them via a high-speed card reader or the RT command.

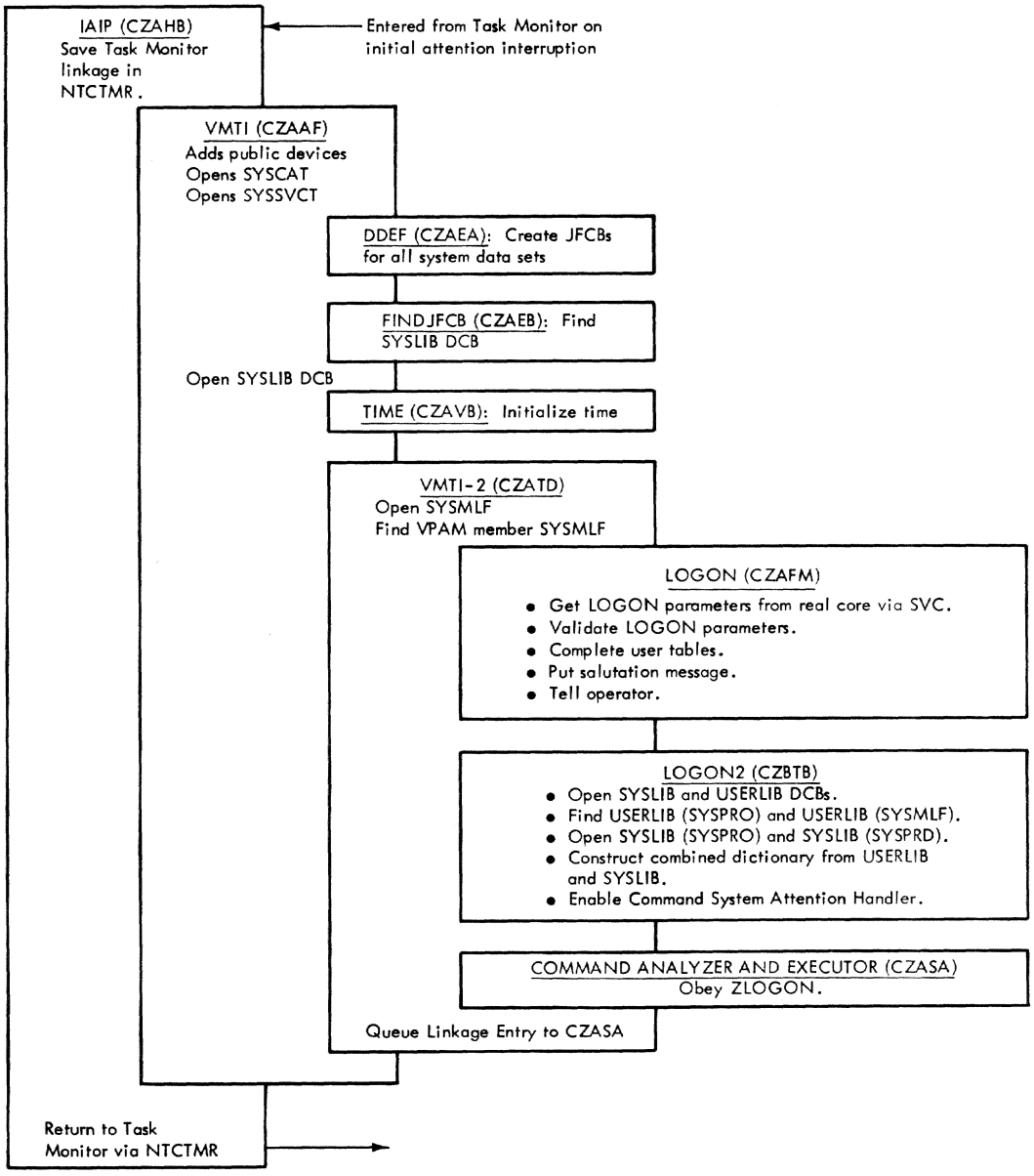


Figure 1. Explanation of LOGON procedure

As a result of the QLE Set up by VMTI-2 during LOGON, the Command Analyzer and Executor is called to service the user's commands ...

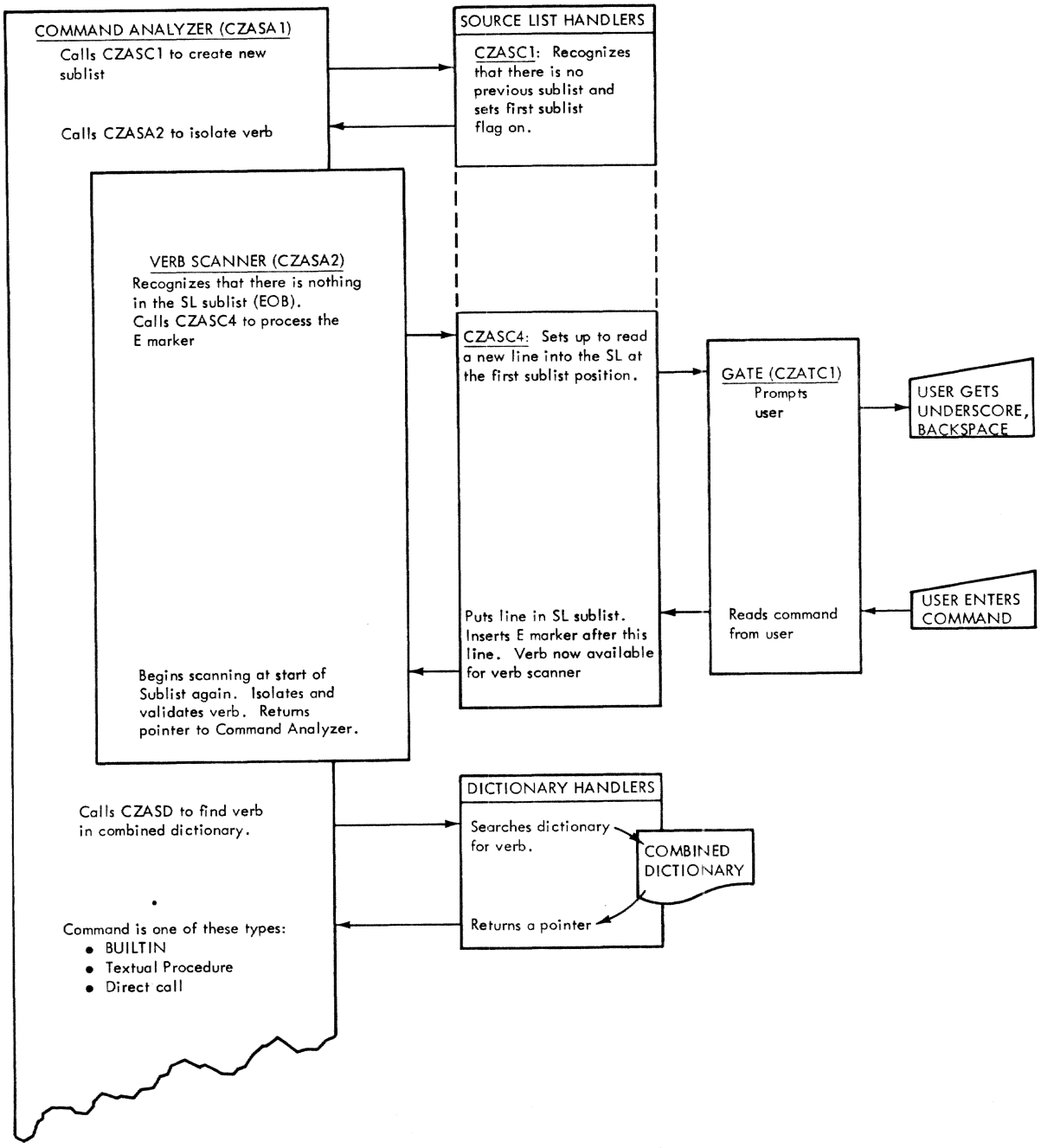


Figure 2. Initial command entry procedure

For an EXECUTE or BACK command, the Command Analyzer and Executor reads the command entered at the terminal and, after checking the command name, calls the EXECUTE or BACK command routine. EXECUTE validates the command operand, and sends a request to the batch monitor to start a nonconversational task. The batch monitor requests a task status index (TSI) for the new task, and initiates the task by sending a message to it; the resulting interruption acts in the same manner as the initial attention interruption of a conversational task. The task monitor and the command system of the new task respond to the interruption by performing the necessary task initialization, and the command system then

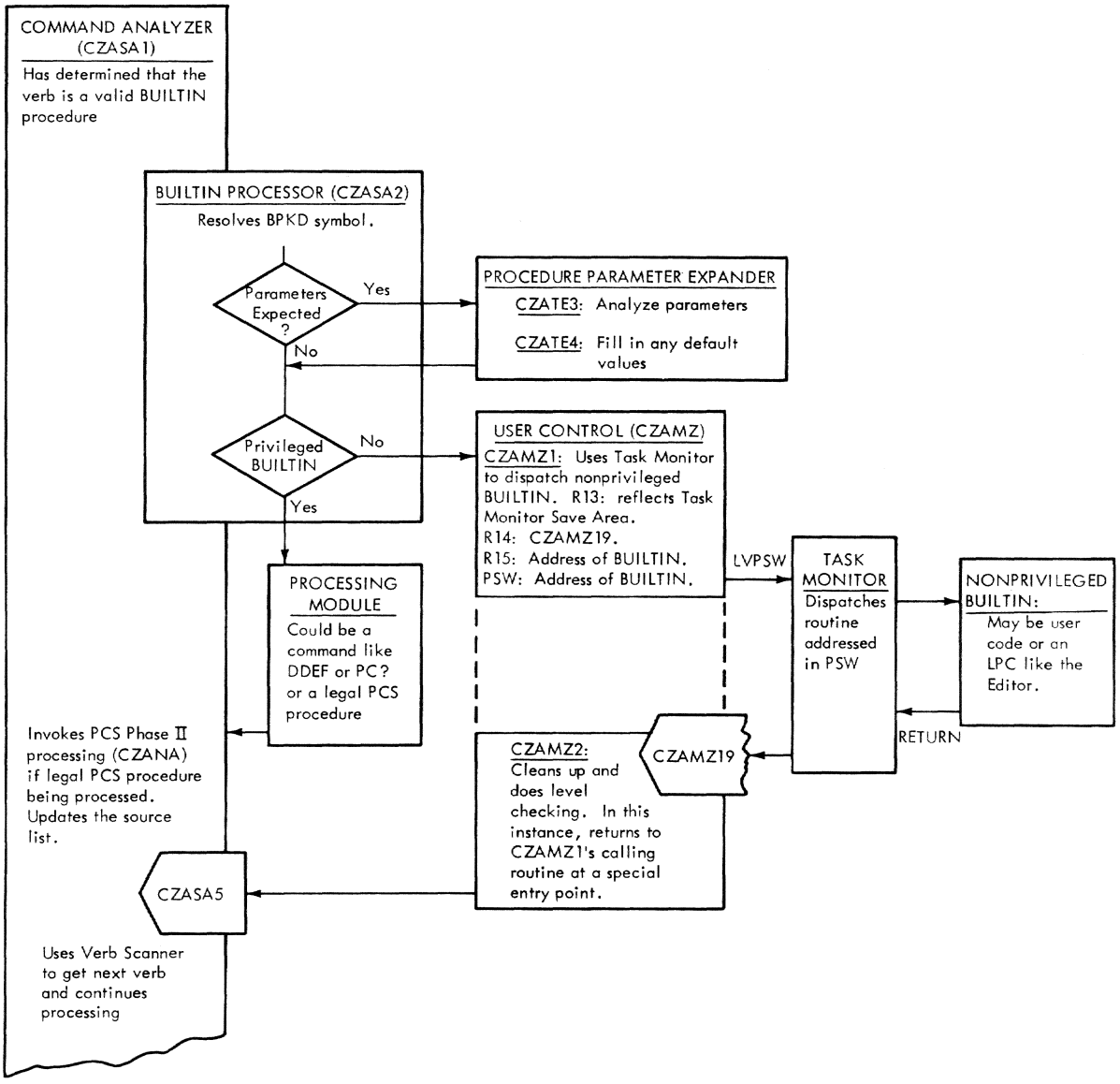


Figure 3. Execution of BUILTIN procedure

fetches its commands from the prestored data set that serves as SYSIN for the nonconversational task. BACK changes the mode of a task from conversational to nonconversational; it uses the batch monitor only to ensure that a new background task is permissible at this time and, if it is, sets the TSI to indicate nonconversational mode. Commands will subsequently be read from the data set specified in the BACK command.

PUNCH, PRINT, and WT commands may appear in conversational and nonconversational tasks; RT commands may be issued only in the operator's conversational task. For each of these commands, the Command Analyzer and executor validates the command name, and calls the BULKIO preprocessor. The preprocessor ensures that the command operands are acceptable; the user will be prompted, if necessary, and the preprocessor concludes with a request to the batch monitor for task initiation. The batch monitor operates in the same way as it would for an EXECUTE. There is no actual SYSIN for bulk input/output tasks. The SYSIN is a sequence of object code supplied as part of the command system; it is not the usual prestored series of commands.

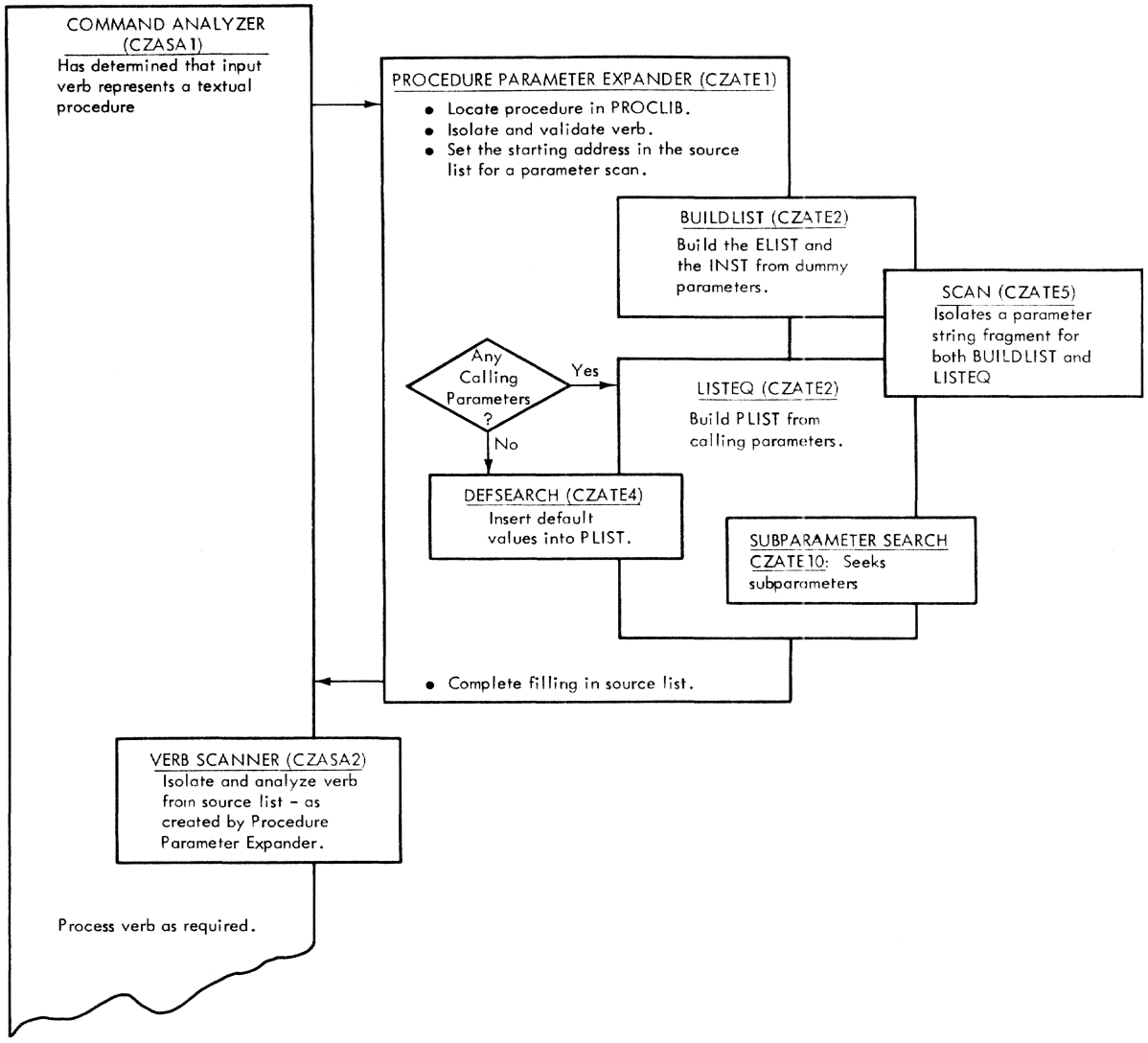


Figure 4. Execution of a textual procedure

Processing A Command

The Command Analyzer and Executor starts command processing by obtaining the next command to be processed from the tasks' SYSIN. The routine checks the validity of the operation before calling the appropriate command routine. The command routine, after verifying the correctness of the command operands, calls on TSS/360 general service routines, and other command routines, to carry out the requested actions, and then returns control to the Command Analyzer and Executor. This routine fetches the next command from SYSIN, and the entire processing cycle is repeated.

The verification performed by the command routine makes use of checking routines included as part of the SCAN package. If errors or omissions are found, the command routine will issue messages to the task's SYSOUT. In the conversational mode, the user may make corrections when responding to system messages, or he may reenter the command. Since the user is not present for a nonconversational task, any message that requires a user response results in termination of the nonconversational task.

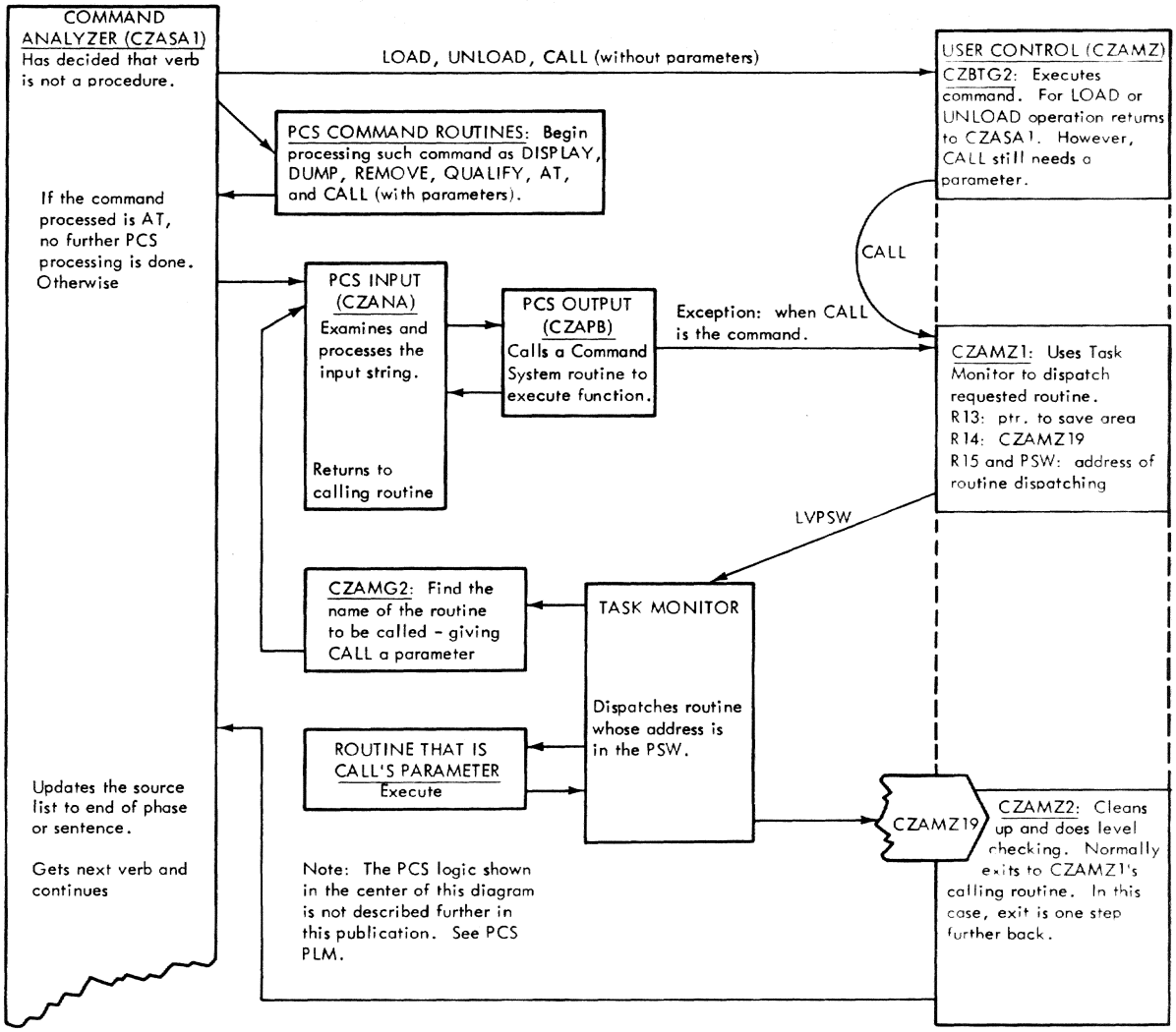


Figure 5. Execution of PCS commands with Direct Call

Terminating A Conversational Task

Since conversational users are available to handle error conditions, conversational tasks are terminated only at the user's request, except in cases of system shutdown, major system errors, or terminal failure. The LOGOFF command routine handles the normal termination; this ensures an orderly conclusion to the task. When LOGOFF has finished, it informs the task monitor which eliminates the task by deleting its task status index and by making its storage available to other tasks. The only trace of the task remaining after this operation is the cataloged data sets, if any, that have been retained.

Terminating A Nonconversational Task

Nonconversational tasks may be terminated either normally or abnormally. Normal termination is the same as a conversational task termination except that the SYSOUT data set, which contains all system communications to a task, is printed. The SYSOUT data set will be listed at the computer center. Another difference is that LOGOFF, when processing a nonconversational task, notifies the batch monitor that the task has ended. This notification is required to permit the batch monitor to update its records.

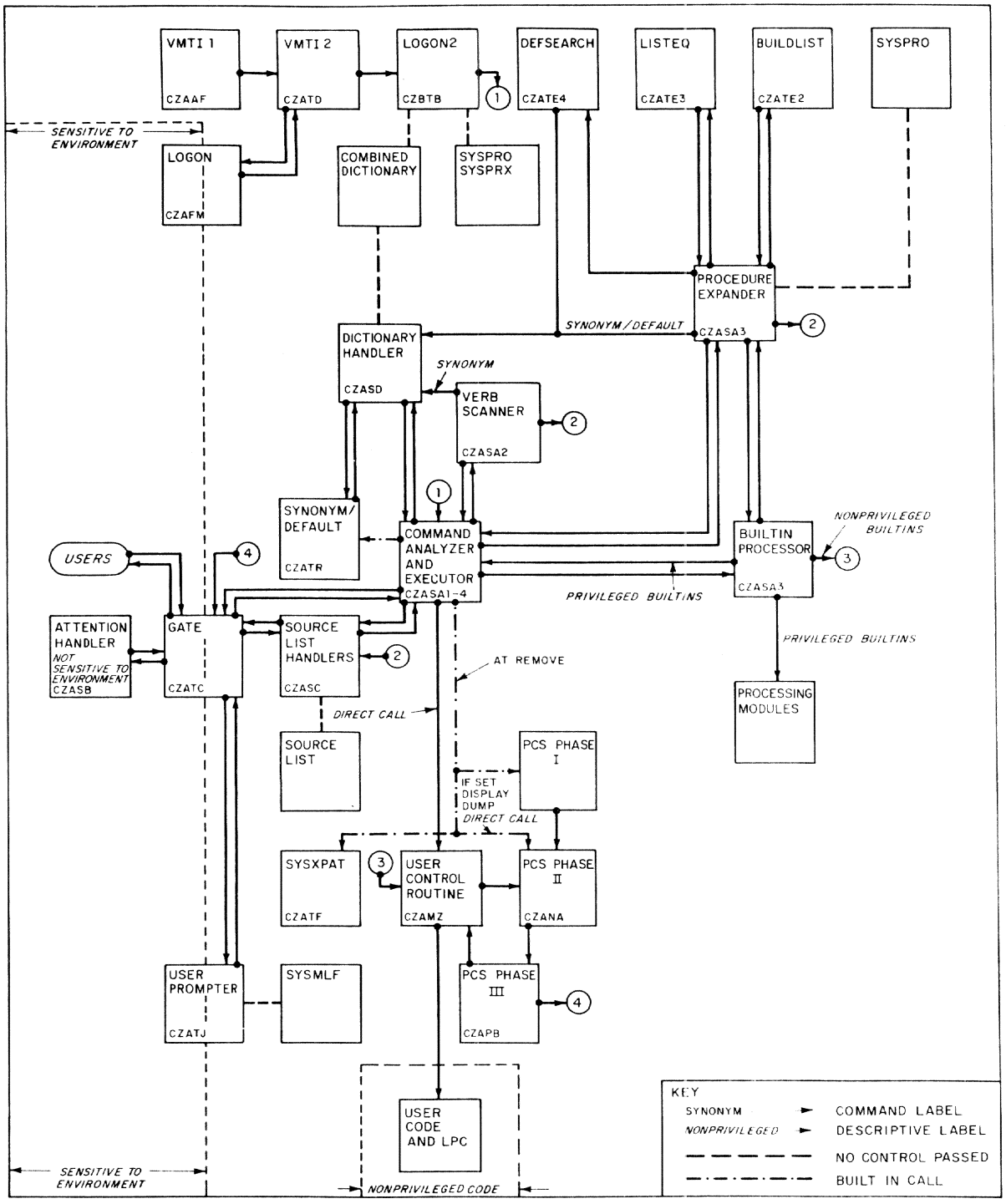


Figure 6. Modular relationship of the command system

A nonconversational task will be abnormally terminated whenever an uncorrectable error is detected. In this case, the command system forces a premature logoff. The SYSOUT of the task will be printed, and it will include a statement that reveals the cause of the abnormal termination.

Processing Macro Instructions

Command system makes no distinction between processing a command or processing a macro instruction, except in one situation. When some command routines are handling a macro instruction, they set indicators so that the macro instruction will not issue messages, but will return to the calling routine a code that reveals the success or failure of the routine. In all other aspects, the command routine operation functions in the same manner as if a command were being processed.

Some command system routines are accessed only by macro instructions. These routines do not ordinarily issue messages, but they do return a code to the calling routine.

When a macro instruction is being processed, there is a direct link between the user's object program and the command system routine; the Command Analyzer and Executor is not involved. This means that the calling routine must perform any error recovery procedures that may be required.

System Shutdown

When the main operator issues a SHUTDOWN command, every active task, except the operator's, is terminated. Conversational users will receive messages that inform them that their tasks have been terminated. Nonconversational tasks will be terminated by the batch monitor, which issues a message to the SYSOUT of each task to explain the reason for termination. After all tasks have been terminated, the operator is informed, and his own task is terminated. He may then physically shut down the system.

The Command Controller consists of the Command Analyzer and Executor routine, and six system support routines. The Command Analyzer and Executor (hereafter: Command Analyzer) obtains the next command to be processed, and determines the verb type from the control dictionaries. The verb type dictates the remaining scan to be used for each verb. The names and functions of the six system support routines are:

Routine Name	Function
GATE	Reads system input (SYSIN) and writes system output (SYSOUT).
SCAN	Fetches and validates command parameters for both privileged and nonprivileged routines.
User Prompter	Communicates with system users.
Attention Handler	Accepts and interprets the synchronous attention interrupts from the task monitor.
Source List Handler	Services the source list.
Control Dictionary Handler	Provides access to the combined dictionary.

The Command Controller provides an interface between the remainder of the command system, and the SYSIN and SYSOUT data streams. Command processing is initiated by the Command Analyzer, which is the basic control routine. When each command has been completed or terminated, control returns to the Command Analyzer. The six support routines are employed as needed. The following paragraphs describe the overall operation of the Command Analyzer, and the support routines.

COMMAND ANALYSIS AND EXECUTION

Command analysis and execution is generally done in three steps: (1) the verb is recognized, (2) the verb operands are translated, and (3) the verb is executed. The Command Analyzer is invoked upon initial task entry. In addition, it may be invoked recursively by a user, upon intervention of an object program, or during control of language processors. Verb resolution is performed by the Command Controller after calling the Verb Scanner (a part of the Command Analyzer). A transient command that is designed to be passed to a language processor at a specific place will cause the remainder of the line to be built into a single string, and passed to the invoked language processor, if a processor is defined.

When the verb is a user or system procedure verb, command analysis calls the Procedure Expander to analyze the parameters, and to expand the procedure. The output from the procedure expansion is added to the source lists.

When command analysis uncovers a conditional statement, the PCS Phase 1 IF routine is called to expand the condition, and the object executor to determine the value of the IF condition. If the condition was not true, a special scanner is invoked to find the end of the statement. Otherwise, the text of the statement is begun by the Command Analyzer.

An AT verb invokes the AT Analyzer and enables the dynamic statement switch. Thus, the remaining statements in the line are analyzed, but they are not executed until the AT is satisfied. In this case, statements which are not executable dynamically are recognized, and appropriate diagnostics are produced.

If the verb was found to be DISPLAY, DUMP, SET, or a program call, then the program parameter list analyzer (PCS FORMLIST and PCS Phase 2 parameter processing) is called to analyze the parameters; the commands are invoked by the appropriate execute routine: DISPLAY/DUMP, SET, or LINK.

Recognition of a DEFAULT or SYNONYM verb results in a call to the DEFAULT/SYN processor to make the appropriate changes to the primary control dictionary.

If the verb was GO, then the last uncompleted program will be resumed. Any commands that were still unexecuted in the source list at the time of intervention will become eligible for execution upon completion of the uncompleted routine. Any commands in the source list following GO will be discarded.

When a program issues an LPCINIT macro instruction, it causes the Edit Controller to be initialized, and the calling program is treated as a language processor controller. A subsequent LPCEDIT macro instruction issued by the language processor controller will cause the LPC to receive input. Any subsequent EDIT commands recognized by the Command Analyzer will cause the EDIT command processor (a part of the User Controller) to be called. The EDIT command processor in turn calls the appropriate Edit routine, which, on completion, exits back through the EDIT processor to the Command Analyzer.

GATE ROUTINE

Normally, GATE either reads from SYSIN and/or writes on SYSOUT, as requested by the calling routine. Four types of writing capability are provided:

- Write with available response (fetch the next record from SYSIN after writing)
- Write with spontaneous response (read the next record from the keyboard after writing)
- Write with no response
- Write with carriage control and no response

SCAN PACKAGE

This package is used by command routines to validate command parameters as well as to fetch those parameters. The SCAN package consists of six routines, whose names and functions are:

Routine Name	Function
NEXTPAR	Locates the next parameter field, and inspects it for invalid characters.
CHEKDS	Checks a data set name for valid form and characters.
ALFNUM	Checks for a valid symbol.

NUMSTG	Checks for a string of valid numeric characters.
CHKNUM	Checks a string of numeric characters, and converts them to a binary value.
ALFBET	Checks for a string of valid alphabetic characters.

USER PROMPTING

System programs use a centralized message communication facility to communicate with external users of TSS/360 Messages, responses, and explanations for messages are defined by system users. All of this information is cataloged in a central file; a call to the User Prompter retrieves it. When the User Prompter is called, via the PRMPT macro instruction or the PRMPT command, it converses directly with the external user by displaying messages and explanations, and by reading his responses. When the conversation has been completed, the User Prompter returns to the calling program a code that indicates the response to the original message.

ATTENTION HANDLING

The Attention Handler accepts interruptions from the task monitor, and interprets any input which may occur. The possible responses by the system are as follows:

1. Intervention is prevented if the Attention Interruption Prevention Switch (AIPS) is set and control is returned normally.
2. A user routine (specified in an AETD macro instruction) is given control to handle the attention interruption.
3. The current process is interrupted, and commands are accepted from the console at a new level.
4. The attention interruption is ignored (an immediate carriage return at the terminal).

SOURCE LIST HANDLING

The Source List Handler processes all entries and deletions to the source list. The source list contains all of the commands that have been entered into the system by the user. Whenever there are no commands in the source list, the Source List Handler will call GATE to issue an underscore and backspace to the user.

The following routines allow manipulation of the source list:

1. PUSH/POP routine.
2. Buffer fetch routine.
3. Marker processors.
4. Synonym expander.
5. UPDATE routine.
6. SYSIN routine.

CONTROL DICTIONARY HANDLING

The Control Dictionary Handler processes all requests regarding the combined dictionary. The combined dictionary, which is the source of the names of all procedures, synonyms, and defaults, is created by the LOGON2 routine (CZBTB).

This routine utilizes a set of internal subroutines and macro instructions, which operate on any dictionary of the form of the control dictionaries. The following routines and handlers are available:

1. STARTFIX routine.
2. STARTVAR routine.
3. HASH macro instruction.
4. RFR routine.
5. NEXTRFR routine.
6. ENTR routine.
7. DELENT routine.
8. EXTDIC routine.
9. PACKVAR routine.

THE COMBINED DICTIONARY

The Dictionary Handler Module (CZASD) is capable of handling any hashed entry. In the command system this module processes entries for the combined dictionary.

The system references: a synonym dictionary, default dictionary, BUILTIN procedure library, textual procedure library, an internal symbol dictionary, and a task dictionary which lists external symbols. These dictionaries cover the range of the language that is used by the system (the vocabulary of the system).

The formation of a combined dictionary permits a reduction of search time. However, it is not possible to include all of the individual dictionaries in the combined dictionary. The internal symbol dictionary and task dictionary are not included in the combined dictionary.

Each entry in the combined dictionary is known by an eight-character name; some information about the kind of entry (source, purpose) is included. Entries containing less than 200 bytes are inserted directly into the combined dictionary. Larger entries contain a pointer in the dictionary entry. Pointers are used for system modules and procedures; values are used for command variables, synonyms, and defaults; in general, the combined dictionary contains the value.

The LOGON2 routine (CZBTB) contains the scheme for creating the combined dictionary; it allows the user dictionaries to be constructed from the user and from system information. In this manner a centralized repository of vocabulary is established.

The system library provides a source of information for the combined dictionary. BUILTIN and textual procedure names are obtained from a data set called SYSPRD, which is a dictionary of system procedures and system BUILTIN procedures. SYSPRD is established at the same time that SYSLIB is established, and it becomes part of the combined dictionary

each time the LOGON procedure occurs. If the user adds any dictionary entries in the form of procedures or builtins to his USERLIB, then these entries will become part of the combined dictionary the next time this user logs on. The dictionary search logic allows a search of the SYSLIB version or the USERLIB version; the user may designate the SYSLIB in preference to USERLIB.

SYSPRX (system prototype file) stores system defaults and synonyms in SYSLIB. If the user has created his own profile, then there is a SYSPRX member of USERLIB. LOGON2 selects the library to be used.

If there is a SYSPRX member of USERLIB (which indicates that the user has created his own profile environment), then that data set is picked up, and the synonyms, defaults, and command variables are added to the combined dictionary. If the user has not created his own profile, there will be no SYSPRX member of USERLIB, and the SYSPRX member of SYSLIB will be used.

The references to dictionary handlers are made by a number of entry codes and macro instructions. GDV (Get Default Value) is the most commonly used macro instruction; this is a call to find a default value for a given name.

Direct calls require no further explanation. However, it should be noted that some of the entry codes are placed in the entry table so they may be called from nonprivileged code.

The SYN/DEF (CZATR), PROCDEF (CZATP), and BUILTIN routines create dictionary entries. The user may create dictionary entries by employing the SET command if he is referencing a command variable or command symbol. PCS (Program Control System) will recognize that a name is a command symbol. If it is, a dictionary entry will be created and the command symbol will be established.

After LOGON2 has established it, the combined dictionary is available to the users. The combined dictionary resides in one or more pages of virtual storage. The space is obtained by GETMAIN.

The PROFILE command is designed to take those entries which represent synonyms and defaults (and optionally, those dictionary entries that are command symbols), and to write a new version of SYSPRX in the USERLIB. This is the way in which the user's profile is built. Profile also gets the syntactical characters (set by MCAST and the translate table) from GATE's PSECT and New Task Common, and makes the characters part of SYSPRX.

At LOGON time GATE and the User Prompter cannot use the combined dictionary because it has not been built. System hard-coded defaults are used until the combined dictionary exists, but this imposes a strict limitation on the system in the initial phases.

The logic of GETMAIN handling for dictionaries starts the dictionary handler with a single page of space obtained through GETMAIN. Additional space is obtained as required. As entries are deleted there is a conservative effort to do some page cleanup.

THE SOURCE LIST

The source list contains a sequenced list of events which are going to happen in the future. There is practically no record of past events in the system. The user is able to change the sequence of events in the source list by invoking an OBEY macro instruction or an expanded procedure.

The source list operates through a pushdown, popup structure (a list structured event). As commands are entered into the system, they are recorded in the source list which also provides working space for various parts of the system. For example, the source list provides a register save area when it is pushed.

When four or five commands on a single line, separated by semicolons, are entered into the source list, the Command Analyzer and the Verb Scanner will isolate a verb. A procedural verb requires that a number of commands be inserted in its place. The new commands are placed in a sublist, and a pointer to the sublist is made available to the system. These sublists can be generated almost indefinitely.

The Command Analyzer always checks the source list for the next command. When the source list is empty, the Command Analyzer links to GATE to prompt the user for additional input.

There are two special occurrences in the source list:

1. Whenever a sublist is created, registers are saved to permit a return to original conditions.

2. E markers indicate the ends of lines or ends of logical segments and sublists.

Items are not erased from the source list until the space is needed. The source list is handled in virtual storage in the same manner as the GETMAIN logic of the dictionaries; only required pages are used. An indefinitely recursive routine can run the source list out of virtual storage, and thus cause a system blowup.

Certain PCS commands (such as AT, IF, DISPLAY, SET) access the source list directly. The syntax of these commands is such that the Command Analyzer would not be able to analyze them.

COMMAND CONTROLLER ROUTINES

The various routines and subroutines that make up the command controller are described in detail in this section.

Command Analyzer and Executor (CZASA)

This routine serves as a driver for command interpretation. It comprises three major modules: the Command Analyzer, the Verb Scanner, and the BUILTIN Call Processor. The Command Analyzer serves as the mainline processor. After it initializes the source list, the Command Analyzer calls the Verb Scanner, which identifies and isolates a verb in the source list. When it returns, the Command Analyzer takes whatever action is required by that verb -- deletes nulls or comments, expands procedures and synonyms, controls language processor, and links to the BUILTIN Call Processor to load and execute BUILTINs. (See Chart AA.)

ENTRIES: The three major modules, as well as certain other functions are defined by the entry points in the Command Analyzer and Executor:

CZASA1 - normal entry to the Command Analyzer
CZASA2 - normal entry to the Verb Scanner
CZASA3 - normal entry to the BUILTIN Call Processor
CZASA4 - entry to the Command Analyzer for nonprivileged users (OBEY entry)
CZASA5 - entry to the Command Analyzer when a command that has been executed (via an attention interruption) returns to it. This entry point is also used on completion of LOGON procedure

CZASA6 - entry point to assume control of attention interruptions (USATT macro instruction)
CZASA7 - entry point to release control of attention interruptions (CLATT macro instruction)
CZASAA - entry to the Command Analyzer after a CC=1 ABEND

MODULES CALLED: The modules called by the Command Analyzer appear below. Later in the list the modules called by the Verb Scanner and the BUILTIN Call Processor are listed.

Source List Handlers:	
POPSL (CZASC1)	Deletes sublists.
PUSHSL (CZASC1)	Constructs new sublists.
SL Update (SCZASC6)	Updates the source list to end of a sentence or a phase.
Verb Scanner (CZASA2)	Isolates verbs within the command string.
Control Dictionary Handlers:	
RFR Routine (CZASD3)	Searches combined dictionary for procedures.
User Control (CZAMZ)	
AT (CZASW3)	Processes transient commands.
AT (CZBTG1)	Handles direct calls without parameters.
PROCEDURE Expander (CZATE1)	Expands identified procedures into the source list.
BUILTIN Call Processor (CZASA3)	Loads and passes control to BUILTIN procedures.
PCS Phase I (CZANA1)	Completes phase processing.
PCS Program Call (CZAMG2)	Handles direct calls with parameters.
User Prompter (CZATJ1)	Prints error messages at the user's terminal.

During its operation the Verb Scanner calls the following routines:

Control Dictionary Handlers:	
RFR Routine (CZASD3)	Determines if the isolated verb has a synonym value.
Source List Handlers:	
SLMAIN (CZASC4)	Processes source list markers.
SYNSL (CZASC5)	Expands synonym values into the source list.

During its operation the BUILTIN Call Processor calls the following routines:

Procedure Expander:	
LISTEQ (CZATE3)	Analyzes parameters and constructs a table of pointers to them.
DEFSEARCH (CZATE4)	Searches for default values.

Source List Handlers:

SL Update (CZASC6)	Updates the source list to point to the next command.

EXITS: When its processing is ended (on error or on exhaustion of the source list), the Command Analyzer returns to its calling routine via the RETURN macro instruction. The Verb Scanner returns to the Command Analyzer when it completes processing. The BUILTIN Call Processor exits by calling a privileged command directly, or be calling the User Controller to call a nonprivileged command processor. When this portion of command processing is complete, the command routines exit via the RETURN macro instruction, which returns to the Command Analyzer at CZASA5.

OPERATION: When entered at CZASA1, this routine calls PUSHSL (CZASC1) to create a new sublist within the source list (SL). After setting up the source list, this routine links to the Verb Scanner (CZASA2) to isolate and analyze a verb in the source list.

Verb Scanner: On entry the Verb Scanner initializes three internal values to reflect a zero return code (RC), character count (CC - length of verb), and depth count (D - used for expressions in parentheses). An additional flag is set to designate an initial outside of comment condition. The GNC macro instruction is now expanded to locate the SL character at the current source address (SA). The following actions provide a description of the verb scan. Leading blanks and underscores are ignored. A left parenthesis causes all characters to be included as part of the verb, until a corresponding right parenthesis is found. This includes all inner sets of parentheses, but should an end of line occur before all of the sets are resolved (one to one correspondence on left and right), an error exit is made. If the first character is the transient command prefix character tne Verb Scanner exits with an RC indicating that a transient statement will follow.

If the first character is a quotation mark, all subsequent characters are ignored until either the quotation is paired, or an end of line occurs. Detection of a comma or equal sign at the beginning of the verb, or outside of quotation marks or parentheses, results in an error exit after scanning (without storing), until a semicolon or end of line occurs.

When the verb's delimiter has been found, the verb is tested for a synonym value. If there is a synonym value, SYNSL moves the verb into the SL, updates the SL and returns control to the verb scanner. The synonym value is examined. If the synonym value changes 100 times during examination, a loop condition is assumed and an error return code is set. If there is no synonym value, the routine exits to the mainline Command Analyzer.

The following checks are made for possible verb errors, and a special code is returned to the mainline Command Analyzer for each type: bad clause followed by semicolon or end of line; bad clause due to unpaired parentheses.

The following return codes will be issued:

RC=0 = Verb, null, or comment followed by blank.
RC=4 = Verb, null, or comment followed by semicolon.
RC=8 = Verb, null, or comment followed by end of line.
RC=12 = Transient statement indicator detected as first character.
RC=16 = Bad clause followed by semicolon.
RC=20 = Bad clause followed by end of line.
RC=24 = Bad clause due to unpaired left parenthesis.
RC=28 = Possible synonym loop.

Return to Mainline: If an error is detected in the verb format (for example: unpaired parentheses), the User Prompter (CZATJ1) is called to send an appropriate message to the user's terminal. Then, the Source List Update routine (CZASC6) updates the source list past the verb and its subfields. Control passes to check for an end of sentence. If a verb is a null clause or a comment, and is not a transient command, control also passes to check for an end of sentence. If a transient command has occurred in a dynamic sentence, an error is recognized, CZATJ1 and CZASC6 are called to account for it. If the transient command did not occur in a dynamic sentence, the Command Analyzer calls User Control (CZAMZ) at CZASW3 to process the transient command. On return, the Command Analyzer loops back in its processing line to again call the Verb Scanner to isolate and analyze the next verb in the source list.

If a verb has a correct format and is neither a null nor a comment, CZASA1 hashes the verb via the HASH macro instruction and calls the RFR routine (CZASD3) to determine if the verb is a procedure name (PROCDEF or BUILTIN). If the verb is not a procedure name, control passes to the PCS Program Call routine to process the direct call if parameters have been supplied. Control otherwise passes to the User Controller at CZBTG1 for a parameterless direct call. If it is a procedure name and the dynamic sentence flag is on (that is, an AT verb exists in the line) the verb is checked to ensure that it is a legal PCS verb; if not, an error condition is diagnosed. If the verb is a procedure, it is expanded into the SL via the Procedure Expander routine (CZATE1); then control is returned to the Verb Scanner to begin processing the PROCDEF. If the verb had been a BUILTIN procedure, the BUILTIN routine (CZASA3) would be called to load and to analyze the calling parameters of the verb, and pass control to the program.

BUILTIN Call Processor: The BUILTIN's dictionary entry is examined to determine if the BPK external name field has been resolved to a VM address. If the BPK external name field has not been overlaid by a VM address, and if the entry indicates a system BUILTIN is being processed, a search of COMTAB is made for the address of the external name. (COMTAB is a table in CZASA's PSECT that contains the external names and BPKD address of each of the system BUILTINs.) If the external name is found in COMTAB, the BUILTIN Call Processor need not call the Loader. Otherwise, this processor retrieves the external name that identifies the BPK from the BUILTIN's dictionary entry, and uses it, in conjunction with the LOAD macro instruction, to insure that the BPK does reside in the task's VM before the verb is called.

The third word of the BPK is tested for an identifier to determine if the BUILTIN command has subparameters. If no subparameter processing is required, a test is made to determine if the BPK defines calling parameters. If no parameters are defined, a further test is made to determine if parameters are supplied. These parameters, which are supplied (but not required) are stripped, and the user is notified that they are being ignored. Calling parameters require additional duties to be performed.

The SL will have to assume a temporary expansion to store parameters. Since the expansion will be deleted upon termination of the BUILTIN, the current entry point is saved for future restoration. The TYPE table which is described in the procedure expander (identifies each BUILTIN dummy parameter as normal, or a special quoted string) is set so that all dummy parameters are normal. This assures that no calling parameter will be stripped of its enclosing apostrophes.

At this point, the LISTEQ routine (CZATE3) is called to establish a table of pointers to the calling parameters. Upon return from LISTEQ, the calling parameters will exist in the SL as a continuous string, where each parameter is preceded by a one-byte length characteristic. The newly constructed PLIST (parameter list constructed by CZATE) will

contain pointers to the calling parameter strings, and each BUILTIN dummy parameter that did not have a corresponding calling value will cause a zero entry within the PLIST at the same relative position.

The DEFSEARCH routine (CZATE4) is now entered to determine if the dummy parameters, corresponding to zero entries in the PLIST, have default values. If they have, the PLIST table is updated with default pointers, thus completing the task of updating the BPK with pointers to the BUILTIN's calling parameters.

If subparameter processing has been identified, processing similar to that described above is performed; however the BUILTIN command processor will call only the LISTEQ routine (CZATE3). LISTEQ will directly call DEFSEARCH (CZATE4) for subparameters. The end result is the same; the PLIST table is established with pointers to the calling parameters.

Processing now is similar for BUILTINs with or without parameters or subparameters. There are two kinds of modules in the system: privileged and nonprivileged, and it is necessary to determine the attributes of this particular BUILTIN. If the BUILTIN is privileged, the module is called directly from CZASA3. Nonprivileged BUILTINs are called via User Control (CZAMZ1) which will perform the privileged/nonprivileged interface and will dispatch them as nonprivileged.

The user is prompted (via PRMPT macro) and an immediate exit is taken under the following conditions:

1. Use of the PIREC macro to check the validity of the BPKD's VCON and RCON causes a program interruption.

2. Module to be loaded cannot be found.

When control is returned to the Command Analyzer, if the BUILTIN is a legal PCS procedure (for example, DISPLAY, IF, RUN), and the dynamic sentence flag is off, control is passed to PCS Phase II to complete phase processing, and invoke the command. At this point, if the invoked verb had been a false IF condition, the SL is updated to the end of that line. If the verb had not been a legal PCS PROCDEF, PCS Phase II would be skipped.

After processing each verb, the end of level flag is checked (indicating that a RUN or BRANCH verb has just been executed, or the end of an OBEY command line has been reached), and if it is on, POPSL is called to delete the current sublist. If the sublist deletion was due to the termination of an OBEY string, control is returned to the calling routine; otherwise, control passes back to initiate a scan of the previous sublist. If the end of level flag is not on, the end of sentence flag is being checked. If the flag is on in conjunction with the dynamic sentence flag, PCS Phase II is entered to complete the processing of all the verbs that follow the AT in the line being processed. In either case, any pending interruption is now allowed to occur prior to processing the next verb.

The Command Analyzer is called at CZASA6 and CZASA7 to assume and release user control of attention interruptions. For these entry points the Command Analyzer issues a direct macro instruction and sets the TCMATT flag in task common. Then control is returned to the calling routine.

ERROR CONDITIONS:

1. If an isolated verb has a format error (ascertained from the verb scan return code), a message to that effect is sent to the terminal, and the verb is effectively ignored.

2. If a non-PCS command is encountered in a dynamic statement, a message to that effect is sent to the terminal, and the verb is effectively ignored.

3. If a transient command is encountered in a dynamic statement, a message is sent to the terminal, and the verb is ignored.

SYSTEM CONTROL BLOCK USAGE: These control blocks are used by the Command Analyzer, the Verb Scanner and the BUILTIN call Processor:

Builtin Procedure Key (CHABPK)
Control Dictionary Entry (CHADEN)
Interrupt Storage Area (CHAISA)
New Task Common (CHANTC)
Profile Character and Switch Table (CHAPCT)
Sublist Header (CHASLH)
Source List Marker (CHASLM)
Source List Page Header (CHASLP)

GATE Routine (CZATC)

Gate is a closed, device independent input/output routine that transmits information from SYSIN devices or data sets to the calling routine, and from the calling routine to a SYSOUT device or data set. (See Chart AB.)

ENTRIES:

CZATC1 - privileged routines to process GATE call
CZATC2 - nonprivileged routines to process GATE call
CZATC3 - GATE's keyboard card reader switch
CZATC4 - maximum line length/device type for conversational I/O
CZATC5 - standard character translate and function table
CZATC6 - TCT slot address
CZATC7 - output character translation table address
CZATC8 - nonconversational SYSIN DCB
CZATC9 - nonconversational SYSOUT DCB
CZATC0 - flag byte
CZATCB - access Control Dictionary for SYSIN=K,ALPHA=B change

MODULES CALLED:

ABEND (CZACP1)	To abort or arrest processing and write an error message.
Control Dictionary Handlers:	
ENTR (CZASD5)	Changes SYSIN and ALPHABET defaults.
GDV (CZASDX)	Gets user defaults for SYSIN, ALPHABET, RSVP, and LINES.
RTAM	To perform actual I/O on conversational SYSIN/SYSOUT device.
CKCLS macro	To verify user parameter pointers.
GET and PUT macros	To access SYSIN and SYSOUT data sets non-conversationally.

EXITS:

Normally - via the RETURN macro with return code in register 15
Abnormally - via the ABEND macro (cc=1 through cc=3)

OPERATION: Five macro instructions, GATWR, GATRD, GTWAR, GTWSR and GTWRC call GATE to write on SYSOUT, to read from SYSIN, or both. GATE first processes any required writing by dividing the message into device-sized lines, or smaller; then the appropriate access method is determined, the output character translation table is applied to the message, and the access method is used to transmit the message to SYSOUT. When reading is required, GATE determines the appropriate access method, uses it to obtain the input message (GATE input buffer is for VAM only; GATE uses RTAM input buffer for conversational reads), and applies the character translation table to the message as it transmits the message to the user's buffer.

GATE Supervisor - GASP: Gate receives control at its proper entry point in GASP. Internal control flags and switches are initialized and build areas are cleared. If user has entered defaults since last entry, GASP retrieves, from the combined dictionary via GDV macro, those defaults applicable to the particular SYSIN/SYSOUT device being used and the mode of operation of the task. System defaults are supplied when there is no entry in the dictionary. Defaults used are SYSIN, ALPHABET, RSVP, and LINES. User parameter list addresses for output are verified via CKCLS macro if user has requested a write. GASP sets up text pointer and length for FAVOR or WORM if address verification is positive.

For conversational tasks, if the device is not assigned to RTAM and the GATE request is from outside the task, GATE will ignore the request. If user is processing in a write/compute environment, GASP will wait for previous write to complete, if necessary, and ascertain normal completion before processing new request. For all write operations GASP acquires the address of the Output Translation Table and checks it. Then, for conversational writes, GASP enters FAVOR to perform output formatting and control character recognition. Nonconversational writes are directed to WORM for similar handling. If a read is required, either separately or in conjunction with a write, GASP will set up user parameters for COWARD conversationally or BARD nonconversationally. User input parameter verification is handled in TRAM routine. GATE's return code to the user is built internally in the PSECT by the GATE routines entered to process the GATE request. The return code is then set at the single GATE exit for return to the user.

Format Conversational Output - FAVOR: User output length is verified. If equal to or less than 0, the write request is ignored. If length exceeds 512, 512 is set for the user length. If user wants carriage control (GTWRC), control characters are prefixed to user output line for double and triple spacing, or skip-to-channel. Control requests for single spacing or no spacing result in no prefixing of characters. If a GTWRC request exceeds one physical line, subsequent lines are single spaced. FAVOR is sensitive to certain function characters, which it processes according to the SYSOUT device being accessed. If the SIC value for a write request has not been specified, FAVOR will translate the output (excepting valid function bytes). SIC causes output translation to be bypassed.

The function characters to which FAVOR is sensitive are restore, bypass, prefix, punch off, punch on, reader stop, tab, new line, backspace, idle, end-of-block, preferred break point, and linefeed. If user line exceeds maximum physical line of SYSOUT device, GATE will break the line at the last preferred break character, or, if none were found, GATE will use the total line before starting the next line. As each physical line is completed, FAVOR adds appropriate carriage control and idles, if necessary, to allow device to position to left margin for next line. When the user's text is exhausted, COWARD is entered to put out user request as a single write. Trailing blanks are stripped off the user's message before any editing occurs.

Write Only (Nonconversationally) - WORM: If SYSOUT DCB is closed, or if user write length is equal to or less than 0, WORM exits to effectively ignore write request. If user wants printer line control (GTWRC), WORM passes user-supplied carriage control character to SYSOUT line and continues processing of text. If user line on GTWRC exceeds one physical line, subsequent lines are given single spacing with no page control. Otherwise WORM controls SYSOUT lines with single spaces and new pages according to user-supplied value for lines-per-page. If SIC is not specified, WORM will translate the output. (SIC as a parameter for a write operation negates translation.)

WORM is sensitive to preferred break-point, new line function characters and the delete function. WORM will always start a new physical line when the new line function is encountered. Preferred breakpoint characters will control line segmenting only when user line exceeds a physical line on SYSOUT device. All preferred break characters are replaced by blanks. When a character is assigned a delete function code, that character will be deleted. Maximum write length nonconversationally is 512 bytes per write request.

Conversational Writes and Reads - COWARD: Upon entrance, COWARD checks the ISA attention bit; if an interruption is pending, COWARD branches to PATTER. Otherwise COWARD determines the appropriate I/O operation, sets up the Terminal Control Table (TCT) slot accordingly, and sets Register 0 to point to the TCT slot.

COWARD determines, by checking the ALPHABET value (originally obtained by GASP) and the device type, whether RTAM or GATE will translate. If RTAM is to translate, this is indicated in the TCT slot. If the terminal is not presently assigned to RTAM, GATE will purge all I/O to that device and then assign the terminal to RTAM.

Next, COWARD issues the ATCS macro (SVC 219), transferring control to RTAM TCS for actual execution of the I/O operation. Figure 7 shows the interaction of COWARD and RTAM.

Upon return, the I/O flag bytes set by RTAM in TCTWFD are reset and tested. If the necessary flags are zero and the Write/Compute flag (in GATE's PSECT) is off, a WAIT T (SVC 204) is issued.

If the Write/Compute flag is on, COWARD exits through GATEEXIT to the calling routine. If the flag byte does not have the necessary flags zeroed, the flag byte is tested further to determine what conditions exist. For I/O errors, COWARD exits to TERROR. For attentions, COWARD exits to PATTER. Otherwise the WAIT T is issued.

When the wait is complete, COWARD rechecks the I/O flags in TCTWFD.

1. For I/O errors, COWARD exits to TERROR.

2. For attentions, COWARD will exit to PATTER.

3. a. For GATWR and GTWRC complete, COWARD exits to the common exit routine unless GASP called COWARD to complete the WRITE, in which case COWARD returns to GASP.

   b. For GATRD, GTWSR, and GTWAR complete, COWARD calls TRAM. Upon return from TRAM, an ATCS is issued for a CLEAR operation, after which COWARD exits to LATE's exit routine. At the exit routine, in the case where the device was reassigned to RTAM, the device is reassigned to the user (unless the GATE request was from LOGOFF or ABEND).

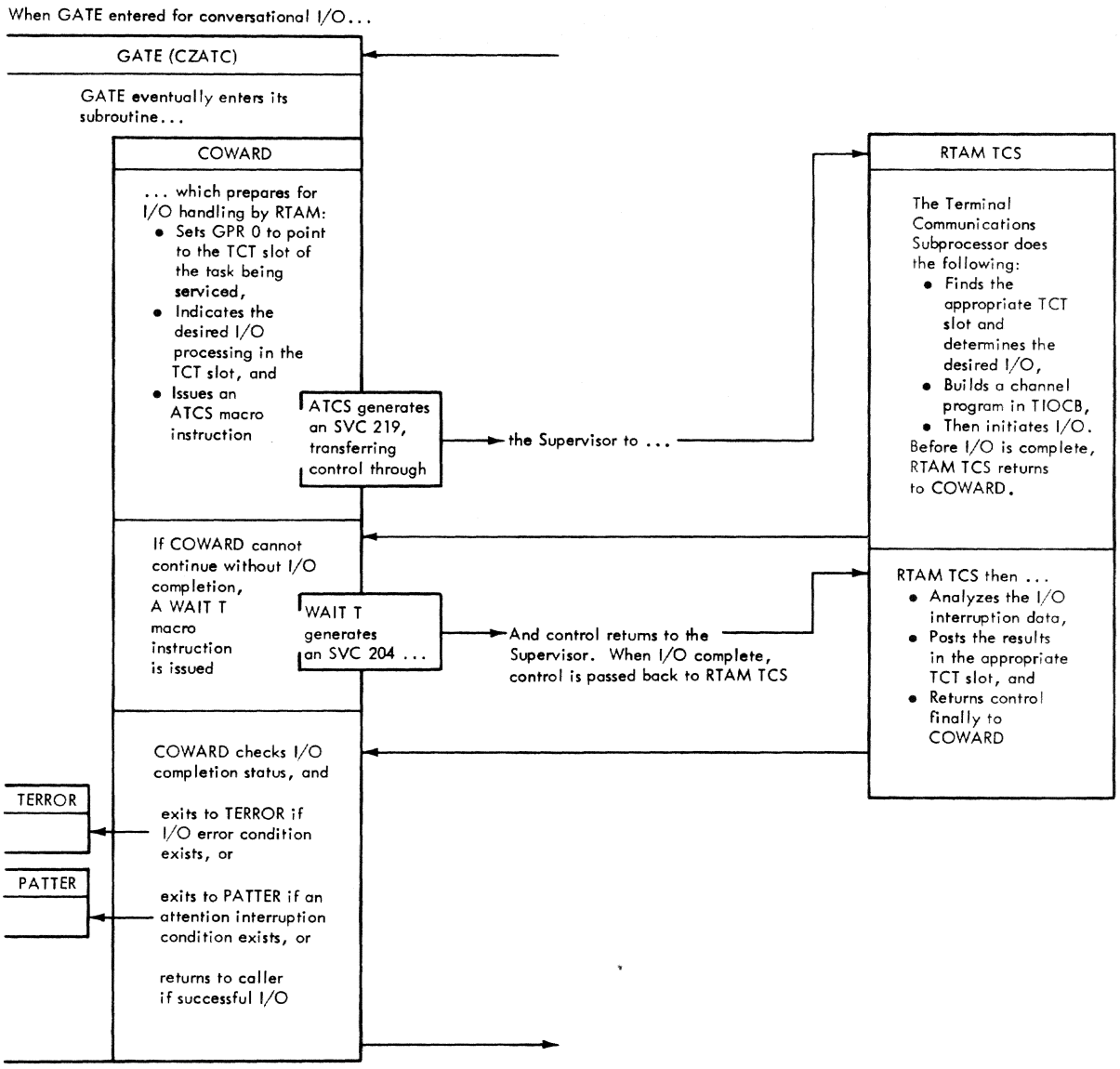


Figure 7. Interaction of COWARD and RTAM TCS

Terminal Error Handler - TERROR: TERROR processes all abnormal conversational I/O termination other than attention interrupts. Error conditions processed by TERROR are unrecoverable error, negative polling response, and input buffer overflow. In unrecoverable error, Gate will retry the operation if it had not retried previously. Otherwise TERROR calls ABEND (cc=3) after setting CZACPSW3 flag indicating terminal cannot be accessed.

For negative polling response (Read operation), GATE retries the read if previous read was from terminal card reader. Retry goes to the keyboard for input. Otherwise ABEND (cc=3) is called.

For buffer overflow (Read), user is informed of input overflow and prompted to re-enter his input at the keyboard.

TERROR also handles operation retry with no error as posted by VSS processor in TCT when user interrupts I/O with VSS active and then requests resumption of task.

Process Attention from Terminal - PATTER: PATTER first determines if user is set for conversational SYSIN. If not, PATTER calls KAYBEE, which in turn calls the Control Dictionary Handler (CZASD5) to orient SYSIN to keyboard with folded translate table. GATE's internal control flags are synchronized with the dictionary. When the attention interruption occurred on any read request, input received before the interruption will be processed and passed back to the caller. After PATTER sets the attention return code in build area, control is passed to the exit routine.

VAM Error - VERSE: VERSE is GATE's VISAM SYNAD routine, and VSAM/VISAM EODAD routine. It exits by calling ABEND.

Translate and Move Input - TRAM: Before this routine is called, GASP has checked certain user parameter pointers for validity with CKCLS macro. TRAM checks further and calls ABEND if user location is inaccessible. If RTAM did not translate input from line code to EBCDIC, TRAM will provide this translation in the RTAM buffer.

TRAM sets its own exit for normal conditions; then it determines whether the SIC mode is specified. If it is, TRAM moves as much of the input record as possible to the calling program's buffer. TRAM performs no character translation, or any other character-indicated function. It sets the truncation return code, if necessary, and furnishes the true character count to the calling routine; then exits. If the SIC mode is not specified, TRAM processes each character, performs the function indicated for the character in the character translation table, and moves the character to the calling program's buffer as required. Scanning is done from left to right. The following paragraphs discuss each of the standard mode functions in detail.

- The Translation Function

When the translation function is specified for a given character, GATE replaces the character with the post-translation value found in the corresponding entry of the character translation table.

- The Null Function

TRAM deletes each encountered character that has a specified null function. If the deletion is imbedded in the line, it is accomplished by joining the characters on the right of the deletion with those on the left, thus condensing the line. If the deletion is at either end of the line, TRAM deletes by appropriately adjusting the starting or ending point of the line before transmitting the line to the user's input area.

- The Character-kill Function

When it encounters a character with an assigned backspace function, TRAM deletes not only that character, but also the preceding character, if any exists. TRAM performs the deletion here just as it does for the null function.

- The Cancel Function

When the cancel function is specified for a given character, and the character is encountered as the last character of the current line, TRAM discards this line and obtains a new line from SYSIN. GATE treats the new line as a replacement for the discarded one.

- The End of Block Function

The end of block (EOB) function truncates the input string by moving only those characters, that are to the left of the EOB, to the user's input buffer.

- Escape Function

Any character to which this code is assigned becomes a one-character escape. The character immediately following is always treated as data.

- The Terminal Null Function

When TRAM encounters a character to which the terminal null function is assigned, it honors that character as a null character only if the EOB function is assigned to the next character, or if the terminal null function is assigned to the last character translated. Otherwise, it treats the character as one to which the translation function is assigned.

Before it returns, TRAM sets the truncation return code, if necessary; it executes the CONT sequence to analyze the continuation status, if truncation is not required, furnishes the true character count to the caller, and exits. If TRAM encounters a cancel function during the processing, it sets its exit for retry, and exits. If no cancel function is found, TRAM terminates when the input runs out, when the user's buffer is full, or when an EOB function is found.

Analyze Continuation Status - CONT: CONT first determines whether the input is in card or keyboard format. CONT assumes a card format if SYSIN is a VSAM data set containing fixed length records; it assumes that the record contains a keyboard/card reader code if SYSIN is a VISAM data set, or VSAM data set containing variable length records, and CONT determines the format from that code. If SYSIN is conversational, CONT determines the format from the source of input. If a card format is determined, the continuation convention requires a hyphen as the last nonblank character in the line. The keyboard convention requires a hyphen as the next-to-last character of the line (before a carriage return). If a continuation indicator is present, CONT sets the continuation return code, and exits. If no continuation indicator is present, CONT exits without setting the continuation return code.

Process Nonconversational (Batch) Read - BARD: BARD performs the input operations required for nonconversational tasks, and controls the translation of input; BARD also controls the transmission to the user's area. BARD calls VAM to obtain the requested input. If no errors are present, BARD executes the TRAM sequence to translate and move the record to the calling program's buffer. If TRAM requests a retry, BARD returns to its initial sequence; otherwise, BARD exits, after setting byte three of the return code to indicate the type of SYSIN data set (VSAM or VISAM). If a transmission error does occur, the VERSE (VAM Error) routine is entered to process the error.

INPUT TRANSLATION (LINE CODE TO EBCDIC) PROVIDED BY GATE:

1. 1052-7 Terminal - Operator's Console.
2. 1050, 2741 Terminals - Support KA option.
3. 1056 card code - Support CA option.
4. 029 card code - Support CB option.
5. 2741 (Correspondence) Terminal - Support ATS Terminal

ERROR CONDITIONS: Conditions which cause GATE to call ABEND:

1. Invalid GATE operation code for nonprivileged user.

2. CKCLS on user parameter pointers indicates location does not have access necessary for GATE to process user request or is not assigned to user's Virtual Memory.

(Conversational Only)

3. Unrecoverable error when accessing terminal.

4. Unable to read from terminal input device(s).

(Nonconversational Only)

5. GTWSR macro used nonconversationally. RSVP not equal to Y.

6. SYSIN DCB closed on Read request.

7. EODAD exit taken on GET from SYSIN.

8. SYNAD exit on GET or PUT request.

9. BARD finds RECFM neither fixed nor variable.

10. BARD finds other than a line data set when RECFM is VI.

Condition which causes GATE to call SYSER:

Invalid GATE operation code for privileged user.

SYSTEM CONTROL BLOCK USAGE:

Character Translation Table (CHACTT)
Data Control Block (CHADCB)
Profile Character and Switch Table (CHAPCT)
New Task Common (CHANTC)
Task Common (CHATCM)
Terminal Control Table (CHATCT)
Interrupt Storage Area (CHAISA)

SCAN Routines (CZAAC)

The SCAN package edits and validates input parameters for the command routines. The SCAN package consists of six routines and five internal subroutines. Nonprivileged entries are provided to five of the routines. At these entries, ten levels of interrupt handling are provided. The routines operate independently, and issue codes to indicate the types of parameters, delimiters, or errors found in the input strings. (See Chart AC.)

The routines are:

NEXTPAR	scanning routine
CHEKDS ALFNUM NUMSTG CHKNUM ALFBET	validating routines

The internal subroutines are:

CKQUAL	validating subroutine
SCINIT	initialization subroutine

FNDBLK
BACKUP scanning subroutines
VALCHK

SCAN CONTROL BLOCK: The SCAN Control Block serves as the input and output control block for the SCAN routines. Each routine expects register 1 to contain a pointer to the control block. The format and contents of the control block are as follows for each SCAN routine:

Byte	0-3	4-7	8-11	12-15	16-19	20-23
NEXTPAR	A	C	D	E	F	G
CHEKDS	A	B	NA	NA	NA	NA
ALFNUM	A	B	NA	NA	NA	NA
NUMSTG	A	B	NA	NA	NA	NA
CHKNUM	A	B	H	I	J	NA
ALFBET	A	B	NA	NA	NA	NA

A = Address of pointer to starting location for scan.

B = Address of 2-byte field containing length of string to be scanned.

C = Address of a 4-byte area in which a pointer to the first non-blank character is to be stored.

D = Address of a 4-byte area into which a pointer to the first delimiter is stored.

E = Address of a 4-byte area in which:

Byte 1 = Initial delimiter code:
1 if left parenthesis
0 if other

Byte 2 = Final delimiter code:
1 if equal sign (=)
2 if comma but no right parenthesis
6 if comma preceded by a right parenthesis
12 if EOM, or EOM preceded by a right parenthesis

Bytes 3 & 4 = Actual character count excluding delimiters and leading leading and trailing blanks.

F = Address of field containing maximum number of characters to be moved or zero, if no characters are to be moved. (NEXTPAR only)

G = Address of first byte in the area to which field is to be moved, if a move is requested. (NEXTPAR only)

H = Address of 4-byte field containing maximum allowable value for CHKNUM.

I = Address of 4-byte field containing minimum allowable value for CHKNUM.

J = Address of area to which converted value is to be moved (CHKNUM only).

NEXTPAR ROUTINE (CZAAC1): NEXTPAR scans an input string for delimiters, and invalid characters, and moves the field to a specified area. It also provides a character count of the actual field, and a code indicating the type of delimiters contained in the string. (See Chart AC.)

Entry: CZAAC1 - normal entry

Modules Called: None.

Internal Subroutines:

SCINIT Performs initialization and housekeeping functions.

FNDBLK Finds the first nonblank character, or the first nonblank character after a left parenthesis.

BACKUP Finds the last nonblank character before a delimiter.

Exits: The routine returns to the calling routine. If an error occurs, the routine sets a return code to indicate the type of error and returns to the calling routine.

Operation: NEXTPAR locates the first nonblank character in an input string, and stores the address in the parameter list. The input string is scanned by NEXTPAR which compares each character in the string to a string of valid character and delimiter codes. If the first nonblank character found is a left parenthesis, this fact is indicated in the output control block, and NEXTPAR scans for the first nonblank character after the left parenthesis. Each character is scanned, and the proper codes are set in the output control block to indicate delimiters or invalid characters. If no invalid characters have been found, the character count is computed. The field is then moved, if the count (specified by calling routine) is correct, and the move option is specified. A return code of 0 is set to indicate a valid parameter.

If an invalid character is found, NEXTPAR will set a return code to indicate this fact, and proceed to the next delimiter in order to position the pointers correctly; however, no characters will be moved. All valid EBCDIC characters are treated as valid by NEXTPAR.

Error Conditions: The routine returns a hexadecimal code in register 15:

Code	Significance
04	Field too long to be moved.
08	Field contained 256 or more valid characters without a delimiter.
0C	Field contained invalid characters.

System Control Block Usage: Task Common (CHATCM)

CHEKDS ROUTINE (CZAAC2/CZAAC2P): CHEKDS checks data set names for valid characters and type of dsname. (See Chart AC.)

Entries:

CZAAC2 - normal entry
CZAAC2P - BPKD entry
SYSAAC2 - nonprivileged BPKD entry

Modules Called: None.

Internal Subroutines:

SCINIT Performs initialization and housekeeping.

CKQUAL Validates qualifiers and module names.

BPKDCNVT Converts BPKD entry to expected form.

Exits: The routine returns to the calling routine. If an error occurs, the routine sets a return code to indicate the type of error, and returns to the calling routine.

Operation: If the BPKD entry is called, register 1 contains a pointer to the entry in the BPKD expansion which points to the dsname. The dsname length is contained in the byte which precedes it. The information is made compatible with the normal entry control block format and processing continues as for a normal entry.

The first qualifier of a dsname is checked for an asterisk to determine if it is an IBM System/360 Operating System name. (Reference: DDEF Command Routine, Operand Handling section.) Each qualifier is then checked for a valid alphameric format. The last qualifier is checked for parenthesis (indicating a relative generation number or member name). Then the validity of the relative generation number or member name is checked. Length checks are also performed. The following codes are returned in register 15:

Code	Significance
00	Valid dsname.
04	Valid member of a generation data group.
08	Valid member of a partitioned data set.
0C	Valid member of a generation data group.
0C	Valid member of a generation data group in a partitioned data set.

Error Conditions: The routine returns a hexadecimal code in register 15:

Code	Significance
10	TSS/360 data set (not a member of PDS or GDG) name exceeds 35 characters.
14	Member of generation data group (GDG) name (exclusive of relative generation number) exceeds 26 characters.
18	Member of partitioned data set (PDS) name (exclusive of module name and relative generation number) exceeds 26 characters.
1C	Member of PDS in GDG name (exclusive of module and relative generation number) exceeds 26 characters.
20	GDG relative generation number is invalid.
24	Incorrect number of parentheses.
28	Data set qualifier blank (2 adjacent periods).
2C	Data set qualifier contains invalid character, or field is empty.

30	Data set qualifier has more than 8 characters.
34	Data set qualifier first character not alphabetic.
38,3C, 40,44	Same as 28,2C,30, and 34 but for module name.
48	IBM System/360 Operating System name contains more than 44 characters with no relative generation number, or more than 35 characters with a relative generation number.

System Control Block Usage: None.

ALFNUM ROUTINE (CZAAC3/CZAAC3P): ALFNUM scans an alphameric string of one to eight characters; it checks that the first character is alphabetic. (See Chart AC.)

Entries:

CZAAC3 - normal entry
CZAAC3P - BPKD entry
SYSAAC3 - nonprivileged BPKD entry

Modules Called: None.

Internal Subroutines:

SCINIT	Performs initialization and houskeeping.
VALCHK	Tests for an empty field or invalid characters.
BPKDCNVT	Convert BPKD entry to expected form.

Exits: The routine returns to the calling routine. If an error occurs, the routine sets a return code to indicate the type of error, and returns to the calling routine.

Operation: If the BPKD entry is used, register 1 contains a pointer to the entry in the BPKD expansion which points to the input string. The length of the input string is contained in the byte which precedes it. The information is made compatible with the normal entry control block format and processing continues as for a normal entry.

ALFNUM tests the input string to insure that it is not greater than eight characters. Next ALFNUM checks the string, character by character, for valid characters, and checks that the input string starts with an alphabetic character. If no error occurs, a return code of 0 is set in register 15.

Error Conditions: The routine returns a hexadecimal code in register 15:

Code	Significance
04	Empty field (character count zero).
08	Invalid character.
0C	Field greater than eight characters.
10	First character not alphabetic.

System Control Block Usage: None.

NUMSTG ROUTINE (CZAAC4/CZAAC4P): This routine scans a field of 256 characters or less for all numeric characters. (See Chart AC.)

Entries:

CZAAC4 - normal entry
CZAAC4P - BPKD entry
SYSAAC4 - nonprivileged BPKD entry

Modules Called: None.

Internal Subroutines:

SCINIT Performs initialization and housekeeping.

VALCHK Tests for an empty field or nonnumeric characters.

BPKDCNVT Convert BPKD entry expected form.

Exits: The routine returns to the calling routine. If an error occurs, the routine sets a return code to indicate the type of error, and returns to the calling routine.

Operation: If the BPKD entry is used, register 1 contains a pointer to the entry in the BPKD expansion which points to the character field to be scanned. The length of the character field is contained in the byte which precedes it. The information is made compatible with the normal entry control block format and processing continues as for a normal entry.

Error Conditions: The routine returns a hexadecimal code in register 15:

Code	Significance
04	Empty field (character count zero)
08	Invalid characters (characters other than numeric).

System Control Block Usage: None.

CHKNUM ROUTINE (CZAAC5/CZAAC5P): This routine converts a numeric field of not more than 15 characters, to a binary number. (See Chart AC.)

Entries:

CZAAC5 - normal entry
CZAAC5P - BPKD entry
SYSAAC5 - non-privileged BPKD entry

Modules Called: None.

Internal Subroutines:

SCINIT Performs initialization and housekeeping.

VALCHK Tests for an empty field or non-numeric characters.

BPKDCNVT Convert BPKD entry to expected form.

Exits: The routine returns to the calling routine. If an error occurs, th routine sets a return code to indicate the type of error, and returns to the calling routine.

Operation: If the BPKD entries are used, register 1 points to the scan control block in which the first word is a pointer to the string with the length in the preceding byte. The second word is a pointer to maximum value, etc. After CHKNUM converts a numeric field into a binary number, it tests the number to ensure that it lies within specified minimum and maximum limits. If no error occurs, a return coe of 0 is set in register 15.

Error Condition: The routine returns a hexadecimal code in register 15:

Code	Significance
04	Empty field (character count zero).
08	Invalid character (characters other than numeric).
0C	Field greater than 15 characters.
10	Binary numer exceeds specified upper limit.
14	Binary number is less than specified lower limit.

System Control Block Usage: None.

ALFBET ROUTINE (CZAAC6/CZAAC6P): This routine scans a field of 256 characters or less for all alphabetic characters. (See Chart AC.)

Entries:

CZAAC6 - normal entry
CZAAC6P - BPKD entry
SYSAAC6 - nonprivileged BPKD entry

Modules Called: None.

Internal Subroutines:

SCINIT Performs initialization and housekeeping.

VALCHK Tests for an empty field or nonalphabetic characters.

BPKDCNVT Convert BPKD entry to expected form.

Exits: The routine returns to the calling routine. If an error occurs, the routine sets a return code to indicate the type of error, and returns to the calling routine. If no error occurs, a return code of 0 set in register 15.

Operation: If the BPKD entry is used, register 1 contains a pointer to the entry in the BPKD expansion which points to the character field to be scanned. The length of the character field is contained in the byte which precedes it. The information is made compatible with the normal entry control block format and processing continues as for a normal entry.

Error Conditions: The routine returns a hexadecimal code in register 15:

Code	Significance
04	Empty field (character count zero).
08	Invalid characters (characters other than alphabetic).

System Control Block Usage: None.

BACKUP SUBROUTINE (AMG2): This subroutine fields the last nonblank character in a string. (See Chart AC.)

Entry: None. Called at BACKUP.

Modules Called: None.

Internal Subroutines:

FNDBLK Finds the first nonblank character in a string.

Exits: The subroutine returns to NEXTPAR.

Operation: BACKUP is called by NEXTPAR, after NEXTPAR has found a delimiter, to find the last nonblank character.

Error Conditions: None.

System Control Block Usage: None.

CKQUAL SUBROUTINE: This subroutine is used to validate qualifiers and module names processed by CHEKDS. (See Chart AC.)

Entry: None. Called at CKQUAL.

Modules Called: None.

Exits: The subroutine returns to CHEKDS. If an error occurs, the subroutine sets a return code to indicate the type of error, and returns to CHEKDS.

Operation: CKQUAL is called by CHEKDS to ascertain that the length of the qualifier or module name is not zero, or greater than eight characters. This subroutine subtracts the length of the qualifier from the total length of the input string, and tests that there are no invalid characters, and that the first character is alphabetic. It then sets a pointer to the character after the delimiter, and returns to the calling point in CHEKDS.

Error Conditions: The subroutine returns a hexadecimal code to CHEKDS.

Code	Significance
28 if qualifier 38 if module name	Data set qualifier or module name blank.
2C if qualifier 3C if module name	Data set qualifier or module name contains invalid characters.
30 if qualifier 40 if module name	Data set qualifier or module name has more than eight characters.
34 if qualifier 44 if module name	Data set qualifier or module name has a nonalphabetic first character.

System Control Block Usage: None.

VALCHK SUBROUTINE (AMA2): This subroutine scans for an empty field or invalid characters. (See Chart AC.)

Entry: None. Called at VALCHK.

Modules Called: None.

Exits: The subroutine returns to the calling routine. If an error occurs, the subroutine sets a return code to indicate the type of error, and returns to the calling routine.

Operation: VALCHK is called by CHKNUM, NUMSTG, ALFNUM and ALFBET to validate a string of characters. If the length of the input string is not zero, VALCHK stores the length in a work area, and tests the string for invalid characters.

Error Conditions: The subroutine returns a hexadecimal code to the calling routine.

Code	Significance
04	Empty field (character count is zero).
08	Invalid characters.

System Control Block Usage: None.

FNDBLK SUBROUTINE: This subroutine scans a string of characters from the start of the string, to find the first nonblank character. (See Chart AC.)

Entry: None. Called at FNDBLK.

Modules Called: None.

Exits: The subroutine exits to the calling routine.

Operation: FNDBLK is called by NEXTPAR and BACKUP to find either the first character in a string (which may be a delimiter), or the first nonblank character after a delimiter. It will check the characters in the string until it finds something other than a blank or a tab, or until it has scanned 256 characters.

Error Conditions: None.

System Control Block Usage: None.

SCINIT SUBROUTINE: This subroutine performs the initialization and housekeeping for the SCAN routines NEXTPAR, CHECKDS, ALFNUM, NUMSTG, CHKNUM and ALFBET. (See Chart AC.)

Entry: None. Called at SCINIT.

Modules Called: None.

Exits: The subroutine returns to the calling routine.

Error Conditions: None.

System Control Block Usage: None.

User Prompter Routines (CZATJ)

The User Prompter is a centralized message locator, display, explanation and response handling facility. It uses the user-defined message file, which is built using the text editor, the system message file, and a table of the most recently used standard messages of the system message file, to perform its functions. (See Chart AD.)

ENTRIES: The User Prompter module is invoked by the PRMPT macro instruction. Individual functions within the User Prompter are invoked by the CALL macro instruction.

CZATJA - entry point for main user prompter (nonprivileged entry)
CZATJ1 - entry point for main user prompter (privileged entry)
CZATJ2 - entry point for MSGSYNTH
CZATJ3 - entry point for MSGEXPL
CZATJ4 - entry point for MSGRESP
CZATJ7 - entry point for EXPLAIN command
CZATJB - entry point for ABEND to unlock message table
CZATJE - entry point for NEWMSG

MODULES CALLED:

GATE (CZATC1)	Prints messages and obtains responses.
ABEND (CZACP1)	When SYSLIB(SYSMLF) is closed.
Dictionary Handlers:	
GDV (CZASDX)	Gets default values for message severity and length.
Source List Handler (CZASC1)	Adds VCON/RCON pair to ABEND table.
SCAN Package:	
CHKNUM (CZAAC5)	Validates and converts a response code in the response line of SYSMLF.

The User Prompter also uses VISAM SETLs and GETs to manipulate the message file.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION (MAINLINE): When a PRMPT macro instruction call is received, the prompter control routine (CZATJA/CZATJ1) calls upon MSGSYNTH (CZATJ2) to locate the message in the message table or the message file. After checking with the filter in the user profile, MSGSYNTH inserts parameters as required and returns control to the User Prompter control routine. If no response is required, control uses GATWR to display the message. If a response is expected, the control routine displays the message, and reads the response using GTWSR. This initial response is checked to see if further explanation is required. If an explanation is requested, the control routine uses MSGEXPL to determine the type of explanation, retrieve the explanatory message required, and use GTWSR until the user is satisfied. If no explanation is required, and the option is unpredictable, the response is stored in a User Prompter work area (CHAAAA). If the option is predictable, MSGRESP accomplishes the task of comparing the response against valid responses. A response code is returned for the valid response. Otherwise, the response is stored in CHAAAA.

MSGSYNTH Routine: This routine retrieves a message, inserts any parameters required, and prepares the message for display. To do this, it first determines the brevity and severity of the message by calling GDV to examine the user profile. This routine then calls its own internal subroutine, GETMSG, to search USERLIB and SYSLIB for the message. The message to be displayed is assembled by inserting parameters, provided as required, into the skeletal message. The ID will be prefixed to the message for identification if the brevity is S or E. It will not be prefixed if the brevity is T or X. Only the ID is given if brevity is M.

MSGEXPL Routine: This routine determines the type of explanation message requested and provides it. After scanning the EXPLAIN response, MSGEXPL acts as follows:

1. If the response is EXPLAIN, the preceeding message is checked to determine whether it is explainable. The message explanation is then provided by calling MSGSYNTH.

2. If the response requests a word explanation, it is retrieved. The word cannot be ORIGIN, RESPONSE, TEXT, MSGS, MSGE, or MSGID. Two one-byte masks, one for the system message file, and one for the user message file, are provided to control searches for word explanations. Each bit in the mask corresponds to a byte in the ID of the message that contains the explainable word. Bits in the mask indicate how many bytes of the current message ID are to be compared in the search for a word explanation record. The mask is scanned from the right. Each bit encountered causes an access to the message file, using the current message ID bytes as a search argument.

   The message file is accessed by using a SETL to the region specified by the mask-designated message ID. If an ID match is found, a serial search matches the explainable word with the word filed in the explanation record. If a word match is found, the search ends. If the message ID or the word do not match, the mask is scanned to the left until the next bit is encountered. This determines the next region where the search will continue. If no bits are encountered in the mask, a blank argument is used to look for a universal explanation before the word explanation is assumed not to exist in the file. If an explanation is not available in the user message file, the system message file is similarly searched.

3. If the response is EXPLAIN ORIGIN, the location of the PRMPT call from a system program and the original message ID will be returned.

4. If the response is EXPLAIN TEXT, the message identified by the message ID name will be located and, if possible, an explanation will be provided according to paragraph one.

5. If the response is EXPLAIN RESPONSE, the possible responses will be retrieved using MSGSYNTH's GETMSG routine.

6. If the response is EXPLAIN MSGS or EXPLAIN MSGE, the standard or extended message line is retrieved using MSGSYNTH's GETMSG routine.

7. If the response is EXPLAIN MSGID, the message ID name is returned.

MSGRESP Routine: This routine determines the validity of the user response, and informs the calling routine about the nature of the response. If a predictable response can be expected, MSGRESP compares the response with the defined responses, and returns the appropriate code to the calling routine if the comparison is satisfied. If no match is found, an error message and error code 28 are returned. If the appropriate code is not valid, error code 48 is returned.

EXPTEXT Routine: This routine permits the direct use of the User Prompter to handle the EXPLAIN command. This routine constructs the explain string, calls MSGEXPL to resolve the explanation, and then issues a GATWR to display that explanation.

NEWMSG Command Routine: This routine allows a privileged system programmer (O authority) to initialize the message table (CHBMSG) when he has changed the SYSLIB copy of the system message file, SYSMLF, during a session. The routine uses TSEND to test the table's lock until it is found unlocked. It then locks the table, initializes pointers and counters to clear the table, and unlocks the table before exiting.

ERROR CONDITIONS: An overall set of error codes is assigned, as shown in the table. The appropriate code is right-justified in register 15 for all routines.

Error Code	Error Description
0	No error. Normal return.
4	I/O error (SYNAD or EODAD, when resp-opt=p).
*12	Message not found in MSGSYNTH routine.
16	Message filtered by user.
20	Insufficient output buffer space provided. Message truncated..
24	Explanation not found in MSGEXPL routine. NO EXPLANATION AVAILABLE displayed to user.
28	Matching response not found in MSGRESP routine.
32	Resp-opt parameter not N, P, or U.
*36	Message continued.
40	Attention interrupt during I/O operation.
44	Too many reference messages chained.
48	Invalid response code in response line of SYSMLF.
52	Response message not in SYSMLF (resp-opt=p).

Those codes shown preceded by an * are internal error code. They are never returned to the module that issues the PRMPT macro.

For returns from a user prompter macro instruction call, the user response (user-resp) return parameter has different values that depend on the error code in register 15.

User Response Contains	When Register 15=
Unchanged (for resp-opt=n)	0, 16, 20, 32, 44
Response Code (for resp-opt=p)	0
Pointer to string (for resp-opt=u) (Note: for zero length response, string is unpredictable)	0
Zero	4, 20, 40, 44
Pointer to EXPLAIN request line	24
Pointer to user response line	28, 48, 52

For returns from the MSGRESP routine, the RSPCD return parameter has the same function as user-resp in a PRMPT macro instruction call. If the error code is register 15 is 0, RSPCD contains the response code. For any error conditions, RSPCD contains 0.

A system error with the code 050330301 is issued when SYSLIB (SYSMLF) is closed. The ABEND message is: 'SYSTEM MESSAGE FILE CLOSED. UNABLE TO WRITE MESSAGE.'

SYSTEM CONTROL BLOCK USAGE:

Editable Data Set (CHACVF)
Data Control Block (CHADCB)
Interrupt Storage Area (CHAISA)
New Task Common (CHANTC)
Profile Character and Switch Table (CHAPCT)
TABLEA - a work area for CZATJ (CHAAAA)

Attention Handler Routine (CZASB)

This routine performs one of the following functions:

- Invokes user-written routines specified by an AETD macro instruction, or
- Obtains a command from the terminal and prepares it for execution for the Command Analyzer by creating a new sublist (thus affecting the source list itself), or
- Obtains and acts upon a terminal request to ignore the attention interruption, repeat the last command, or invoke abnormal termination, or
- Honors another attention interruption.

(See Chart AE.)

ENTRIES:

CZASB1 - actual attention entry from task monitor
CZASB2 - simulated attention entry
CZASB3 - entry from CZAMZ created stack
CZASB5 - entry from AETD macro instruction
CZASB6 - entry for ABEND command
CZASB7 - entry for STRING function if entered other than after ATTN

MODULES CALLED:

GATE Routines (CZATC1):

GATRD	Reads from SYSIN.
GATWR	Acknowledges the attention, and writes diagnostic and prompting messages on task SYSOUT.
GATWSR	Acknowledges the attention, and obtains input from the terminal.

User Control (CZAMZ):

PCSEXEC (CZAMZ1)	Executes the user's attention routine.
INTERVENE (CZAMZ3)	Issues a call to the Command Analyzer to process inputs from the terminal if intervention is not prevented.

User Prompter (CZATJ1)	Reports errors.

EXITS: The Attention Handler exits to the Task Monitor. It does not provide a return code, although it is responsible for setting certain switches.

OPERATION: When entered at CZASB1, its entry point for servicing actual attention interrupts, the Attention Handler decrements CZCJT1, a field

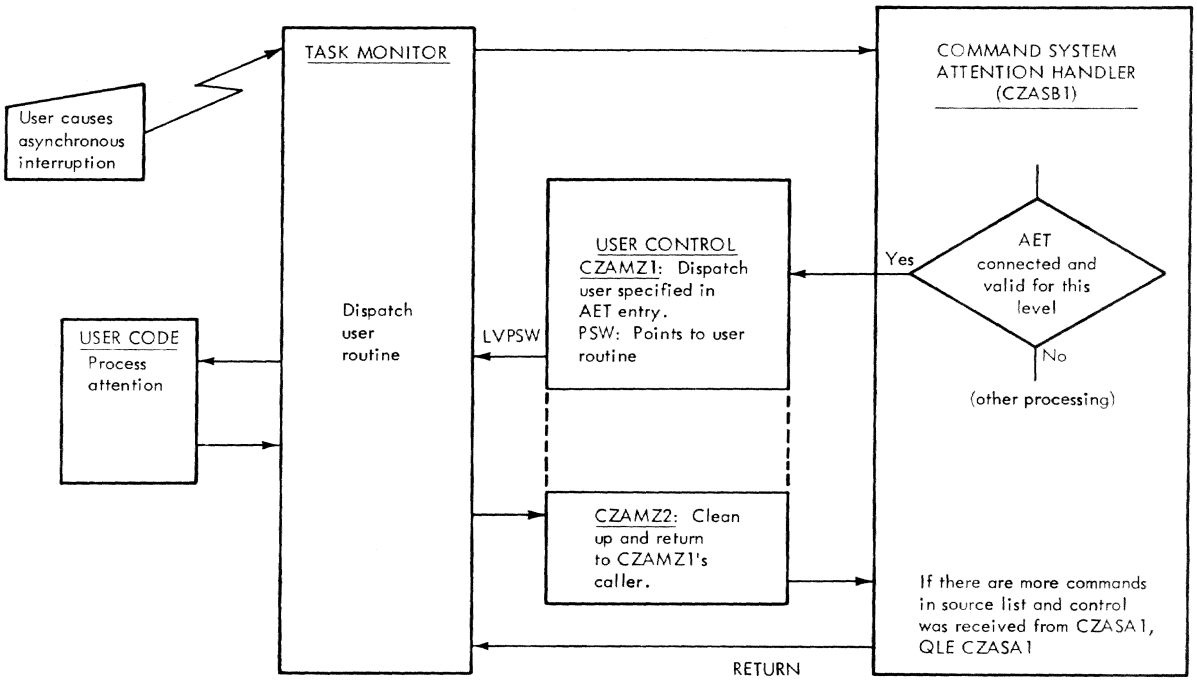


Figure 8. Attention processing with AET connected

in the Task Monitor that contains a count of pending QLEs. (Entry was gained at CZASB1 through a QLE -- one less QLE exists to be processed.)

The Attention Handler then tests the AIPS switch. This switch can be set by a user program to prevent an attention interrupt. If it is on, the Attention Handler arms the Attention SVC, by placing an SVC in an area that is usually a NOP. If the user executes the SVC, an attention is simulated. Consequently the Attention Handler turns on an "internal simulated attention pending" flag. The user is prompted that the AIPS switch is on, and the Attention Handler branches to its exit logic.

If AIPS is not found on, the Attention Handler takes the extra precaution of turning it off and clearing the area that would be occupied by the Attention SVC.

The Attention Handler then tests for an Attention Entry Table (AET). If an AET is connected, the Attention Handler increments the attention count in CZAMZE and uses that count to determine if the user has specified a program to deal with this level of attention. (The AET contains entries which correspond to the values of the attention interrupt count.) If the user has specified a valid program, the Attention Handler links to User Control (CZAMZ1) to dispatch the user routine. On return, the Attention Handler decrements the attention count in CZAMZE, and branches to its exit logic. Attention processing with an AET connected is illustrated in Figure 8.

If no AET is connected, or if there is not a valid user program for the updated count in CZAMZE, the Attention Handler tests the QLE pending count in CZCTJ1. If CZCTJ1 equals other than zero, this routine returns to its calling routine -- to allow other QLEs to be dispatched.

If the QLE count is zero, the Attention Handler will service the attention interrupt. The appropriate attention prompt character is transmitted to the terminal via GTWSR. The character transmitted is determined as follows:

Character	Situation Governing It
-	Attention occurred during execution of a privileged command which was the last command to be processed within the current input string.
*	Attention occurred during the execution of a privileged command that is embedded in the command string. Only occurs if additional commands remain to be processed within string.
!	Attention occurred during execution of a command or routine running in the nonprivileged state.

(The user may have defined his own string in place of the underscore.) A number of responses to the prompt character are possible:

1. Another attention interrupt -- The Attention Handler immediately exits, returning to its calling routine.

2. A continuation is indicated -- The Attention Handler uses GATRD to continue reading the response, and, on return, loops back through its response-checking logic.

3. Input is null -- The Attention Handler returns to its calling routine, after first putting a QLE to CZASA if control was passed from it.

4. STRING function is requested -- The Attention Handler displays all items in the source list which have not yet been processed. It then loops back to test for pending QLEs.

If none of the five responses shown are encountered, the Attention Handler calls the INTERVENE routine (CZAMZ3) in User Control to analyze the response input. On return, if the AET is not active, the Attention Handler branches to its exit logic. If the AET is active, it first decrements the count in CZAMZE.

Exit Logic: If the task is the operator's and no more commands exist in the source list, the Attention Handler exits directly to its calling routine. Otherwise, if control was passed from the Command Analyzer (CZASA1), the Attention Handler puts a QLE to the Command Analyzer and then exits.

Simulated Attentions: At CZASB2, its entry point for simulated attentions, the Attention Handler tests its "internal simulated attention pending" flag. If it is not on, this routine links to CZATJ1 to inform the user that a simulated attention has occurred. Then, or if the flag is on (as the result of an earlier attention with AIPS on), the Attention Handler turns off the flag, disarms the attention SVC, and enters the main routine logic at the point where the main routine tests to determine if an AET is connected.

AETD Macro Servicing: At CZASB5, the Attention Handler first clears the attention count in CZAMZE. It then tests to determine whether (1) the address of the new AET has been supplied, and (2) there are any entries in the AET table. If the answer to either query is no, this routine clears CZAMZD, which is the AET pointer in CZAMZ's PSECT, then turns off the active AFT flag, and returns to its calling routine. But, if both conditions are satisfied, the Attention Handler begins testing the entries in the AET to find one that is used. When one is found, this routine sets a new pointer in CZAMZD, sets the active AET flag on, and exits. If the list is exhausted before a used entry is found, this routine clears CZAMZD, turns off the AET flag, and exits.

Other Functions: There are three other entry points in this module: CZASB3, CZASB6, and CZASB7. CZASB3 is an entry point put into CZAMZ's stack and used, when encountered, to return to the Command Analyzer via QLE. CZASB6 is the entry point for the ABEND command. CZASB7 is used if the user requests the STRING function at a time other than after an attention interrupt. When entered, this routine calls CZATJ1 to inform the user that STRING is only valid after attention, and, on return, exits.

ERROR CONDITIONS: None, other than misuse of the STRING function, as described above.

SYSTEM CONTROL BLOCK USAGE:

Attention Entry Table (CHAAET)
Interrupt Storage Area (CHAISA)
New Task Common (CHANTC)
TABLEA (CHAAAA)
Profile Character and Switch Table (CHAPCT)

Source List Handlers (CZASC)

The Source List Handlers package consists of seven separate functions (each defined as a separate entry point) that provide the command system with a means of updating and interpreting the contents of the source list. (See Chart AF.)

ENTRIES: Each of the following entry points defines one of the source list handlers:

CZASC1 - sublist addition (push)/ deletion (pop) routine
CZASC2 - source list page acquisition routine
CZASC3 - source list processor for all markers except E
CZASC4 - source list processor for E markers
CZASC5 - source list synonym expander
CZASC6 - source list update routine
CZASC7 - privileged entry for read from/into source list via SYSIN macro
CZASC8 - nonprivileged entry for read from/into source list via SYSIN macro

SOURCE LIST HANDLER PUSH/POP (CZASC1): This routine allows for the addition (PUSH) and deletion (POP) of sublists within the source list (SL). (See Chart AF.)

Entry: CZASC1 - normal entry

Modules Called:

Virtual Memory Allocation (CZCGA)	GETMAIN (CZCGA2) gets storage for the first page of the source list.

Exits: The routine normally returns to the calling routine, via the RETURN macro instruction.

Operation: The routine tests an address field for the source list (CZASC9) to determine if this is the task's first entry for construction of the SL. If CZASC9 is zero, the entry flag (INITE) is zeroed and set to indicate initial entry (INITEM=ON). GETMAIN is then called to reserve storage so that the task's SL can be constructed from scratch. If CZASC9 is not zero the entry flag (INITE) is tested for initial entry (INITEM=ON) and for a previous POP action which reverted to the SL's first sublist (INITEM1=ON). The SL is then reinitialized to its first possible sublist state. For both cases, values are set in the SL Header, a new sublist is constructed and INITEM is set ON for successive entries.

If INITEM=ON, the current sublist base is set from SLPCSL, and a test is made to determine what type of entry was made. For a POP entry, the sublist's generation marker's pointer field (SLHPTR) is tested. If the field is zero, INITEM=OFF, INITEM1=ON, and EXIT. If it is nonzero, SLPCSL is reset from SLHPTR (the sublist pointer now points to the previous sublist), and SLPAVL is enlarged by the number of bytes existing in the POPed sublist. If the new sublist is a G type (non-OBEY), SLPGIP is reset to point to its SLHPTR field. In either case, an immediate EXIT is made.

If the entry is a PUSH type, a check for sufficient sublist space is made (390_1 bytes minimum). Insufficient space results in a call to CZASC2 for an additional page. The new sublist base is returned from CZASC2 which also updates SLPAVL accordingly. If there was enough space initially, the new sublist base is computed from the previous sublist's end point (SLHEND), which is rounded to the next fullword boundary.

In either case, the previous sublist pointer is moved into the SLHPTR field of the new sublist. If the new sublist is an OBEY type (user call to command analyzer), its generation marker is set to U; otherwise, is set to G. Also for the G type, the previous sublist's SLHPTR field is set to zero, and a null command line is input, the start address (SLHCSA) is set to the beginning of the generation marker; SLHEND is set to its end; SLPAVL is diminished by the number of bytes in the new sublist; SLPCSL is updated; and, an EXIT is made.

If the input command line isn't null, it is moved into the SL after the new generation marker. An E marker is placed at the line's end, which points back to the generation marker's beginning. SLHCSA is set to the line's first byte address; SLHEND is set to the address of the first available byte after the E marker; SLPAVL is diminished by the number of bytes in the sublist plus the command line (including 6-byte E marker); and, an EXIT is made.

Error Conditions: There are no error conditions for this routine.

System Control Block Usage:

Source List Marker (CHASLM)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)
Task Common (CHATCM)

SOURCE LIST HANDLER BUFFER FETCH ROUTINE (CZASC2): This routine is called to obtain a page of virtual storage for source list (SL) expansion and to provide page-to-page linkage. (See Chart AF.)

Entry: CZASC2 - normal entry

Modules Called:

Virtual Memory Allocation (CZCGA)	GETMAIN (CZCGA2) obtains additional source list pages.

Exits: The routine normally returns to the calling routine, via the RETURN macro instruction.

Operation: The routine tests the first four bytes of the current SL page. If they are zero, GETMAIN is executed to obtain an additional SL page after which the address of the new page is placed in the first four bytes of the previous page. If the four bytes were nonzero, the page to which they point would be used.

If the entry is for a line expansion (input=pointer to the fullword containing 1), the current SL end pointer (SLHEND) is set to the address

of the new page's first available byte (fifth byte position); the SL available byte count (SLPAVL) is updated to 4092_{10}; and, an EXIT is made. If the entry is for a new sublist (input pointer to full-word containing zeros), the new page's address is returned to the calling routine; SLPAVL is set to 4092_{10}; and, an EXIT is made.

An entry for textual procedure expansion is assumed if the two previous types are not specified. A T marker is placed in the old page at the current end point, which points to the first available byte in the new page. SLHEND is updated to the address of the first available byte in the new page; SLPAVL is set to 4092_{10}; and, an EXIT is made.

Error Conditions: There are no error conditions for this routine.

System Control Block Usage:

Source List Marker (CHASLM)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)

SOURCE LIST HANDLER MARKER PROCESSORS (CZASC3/CZASC4): The routine which is entered to process source list (SL) markers, recognizes when new lines must be input via GATE and, in addition to expanding the SL, it deletes information when the markers indicate that this action is necessary. (See Chart AF.)

Entries:

CZASC3 - entry point to process all but E-type (connection/termination) markers
CZASC4 - entry point to process E markers

Modules Called:

GATE (CZATC) To write a prompt string and read into the source list.

Exits: The routine normally returns to the calling routine, via the RETURN macro instruction.

Operation: When the module is entered at CZASC3, the current start address (SLHCSA) is tested for a pointer to a T marker. If there is a pointer, SLHCSA is reset from the T marker's pointer field, and an EXIT is made. If there is no T, then a G marker is assumed. The address of the standard prompt string is obtained from PCTPS new task common. Control now passes to CZASC4 to prompt for a new command line.

At entry point CZASC4, SLHCSA is immediately reset from the E marker's pointer field. If it doesn't point to another marker, SLHCSA is updated, and an EXIT is taken. Otherwise, a check is made for a U marker. If there is one, register 15 is set to hexadecimal FF to signal the calling routine to initiate a sublist deletion, and an EXIT is taken. Also, if the U marker doesn't point to a G or P marker, an EXIT is taken. If the U marker points to a P marker, the current end point (SLHEND) is set to the beginning of the P marker; SLHCSA is set from the P marker's pointer field; the available byte count (SLPAVL) is increased by the number of bytes from the beginning of the P marker to the end of the E marker, and an exit to the calling routine is made.

The last possibility is the G, or generation marker. In this case, the G marker address is saved. If its pointer field is nonzero, SLPAVL is reset to reflect the G marker's position relative to the SL page beginning, and control passes to prompt for a new line to be read into the SL, after the G marker. If the G marker's pointer field is zero, the SL must be reset to its first sublist position. To do this, the sublist pointer (SLPCSL) is reset to 16_{10}; the G marker address is reset

to the first SL generation marker position and saved; SLPAVL is reset to 4058_{10}; and, the G type sublist pointer (SLPGIP) is reset to point to the new G marker.

At this point, regardless of the above path, GATWR is used to prompt for the next command line with the standard prompt message (underscore-return) in new task common. This is followed by a call to GATRD for the command line to be read into the SL after the G marker. An E marker is now placed after the command line, and a test is made for a continuation line (if the last character in the line is a hyphen, a continuation line is expected). If no continuation is expected, the E marker is set to point back to the saved G marker address; SLHEND is reset to the end of E marker, SLHCSA is reset to the first byte of the command line; SLPAVL is diminshed by the line length (plus the 6-byte E marker); and, an EXIT is taken.

If there is a continuation, a test is made to determine if sufficient SL space exists for another line. If not, CZASC2 is called to expand the SL by another page. In any event, the E marker is set to point to the first available SL byte past itself (this may be on another page), and control passes back to prompt to read another line into the SL at the position to which the E marker points. This logic is repeated until all continuation lines are read, after which the above procedure for no continuation line is executed (last E marker points back to the G marker).

Exits: This routine normally returns to the calling routine, via the RETURN macro instruction.

System Control Block Usage:

New Task Common (CHANTC)
Task Common (CHATCM)
Source List Marker (CHASLM)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)

SOURCE LIST HANDLER SYNONYM EXPANDER -- SYNSL (CZASC5): This routine is called when a synonym value exists for an isolated command. The synonym value is placed into the SL, and pointers are reset for its eventual execution. (See Chart AF.)

Entry: CZASC5 - normal entry

Modules Called: None.

Exits: The routine normally returns to the calling routine, via the RETURN macro instruction.

Operation: The routine establishes the base of the combined dictionary entry. A test is made to determine if enough space exists in the SL for the synonym value, plus 12 bytes for the prefix P and terminating markers E. If space is not available, CZASC2 is called to expand the SL by one page. In either case, a P marker is now placed in the SL at the current end pointer (SLHEND), which points back to the current start address (SLHCSA). The P marker address is saved, and SLHCSA is set to the first byte after the P marker. The synonym value is now moved from the combined dictionary entry (its length precedes it in the dictionary entry) into the SL at the new SLHCSA. An E marker, which points back to the P marker, is placed at the end of the synonym value in the SL. SLHEND is set to the end of the E marker; the available byte count (SLPAVL) is diminished by the number of bytes in the synonym value plus 12 bytes for the markers, and an EXIT is taken.

Error Conditions: There are no error conditions for this routine.

System Control Block Usage:

Source List Marker (CHASLM)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)

SOURCE LIST HANDLER UPDATE ROUTINE (CZASC6): This routine is called to update the source list (SL) to the next command, or to the end of a command line. (See Chart AF.)

Entry: CZASC6 - normal entry

Modules Called: None.

Exits: The routine normally returns to the calling routine via the RETURN macro instruction.

Operation: The routine establishes the current sublist base from SLPCSL, and executes the GNC macro instruction to isolate the SL character at the current start address (SLHCSA). The isolated character is tested for an EOB. If it is an EOB, an immediate EXIT is taken. If it is not, a test is made to determine if this entry is to update the SL to the next command. If it is not to update the SL, the routine isolates the next character. If it is, the current character is tested for a semicolon. The absence of a semicolon causes another character to be isolated; the presence of a semicolon causes an EXIT. In any event, an EXIT updates SLHCSA to the next command (the previous command is delimited by a semicolon) or to the end of the line (delimited by an EOB).

Error Conditions: There are no error conditions for this routine.

System Control Block Usage:

New Task Common (CHANTC)
Source List Marker (CHASLM)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)

SOURCE LIST HANDLER SYSIN ROUTINE (CZASC7/CZASC8): This routine is called via the SYSIN macro when a user desires to read data lines into or from the source list (SL) and to pass this information to a user-specified area. (See Chart AF.)

Entries:

CZASC7 - privileged entry
CZASC8 - nonprivileged entry

Modules Called:

GATE (CZATC)	To write from or read into the source list.
Control Dictionary Handler (CZASD)	To test hashed string for a default value.

Exits: The routine normally returns to the calling routine via the RETURN macro instruction.

Operation: The user may indicate the data source by specifying one of the following EBCDIC source codes:

- G indicates that a line is to be obtained from SYSIN via GATE.
- L indicates that a line is to be obtained from the source list.

- E indicates one of two alternatives: (1) if the default value for SYSIN is G, a line is to be obtained from SYSIN via GATE. (2) If the default value for SYSIN is not equal to G, a line is to be obtained from the SL.

The user may also indicate that commands are to be treated as data or message input by appending location code, S, to one of the above source codes.

In processing G or E (type 1) requests, the user's SYSIN device is prompted for an input line. If there is not enough space in the SL for a new line (268_{10} bytes), the SL is expanded via a call to CZASC2. The current starting address of the SL (SLHCSA) is saved. If the task is conversational, the routine tests for a prompt message. If there is a message and the default value for LINENO is Y, a GTWAR is issued, otherwise, a GATRD is issued. The input line is read into the SL at its current end point (SLHEND). SLHCSA is updated to the line's first character and an E marker that contains the previously saved SLHCSA is placed at the end of the line. If the task is nonconversational, the prompt string (if one exists and LINENO is Y) is moved into the SL; SLHEND is updated past the prompt string and a GATRD is issued. The SYSIN line is put into the SL at its updated SLHEND and the actual record length is passed to the user. Then the entire string (prompt and input data) is written via GATWR to SYSOUT. An E marker that contains the previously saved SLHCSA is placed at the end of the line. SLHEND is updated to point past the E marker and the available byte count (SLPAVL) is diminished by the number of bytes in the line plus 6 bytes for the E marker. Then for both conversational and nonconversational tasks, the input line is transmitted to the user.

In processing L or E (type 2) requests, the character at SLHCSA is tested. If SLHCSA points to an E marker and the SYSIN code is LS, an exit is taken with a length of zero passed to the caller. If SLHCSA points to an E marker and the E marker points to a U marker, the user is notified via User Prompter that the request is non-serviceable and an exit is taken. Otherwise, the E marker is processed via CZASC4 and a line is obtained from the SL.

For all requests, the SL pointer is updated, beginning at SLHCSA, to the first nonblank character, and the user input value for the maximum number of transmission characters is set. If the first nonblank character is not an underscore, a data line is transmitted. If the first nonblank character is an underscore, the next character is tested for an underscore. If there are two underscores at the beginning of the SL, the line is treated as data. Otherwise, the line is treated as an immediate command. Before termination, a test is performed to determine if the command is to be treated as data. If not, a return code of C is set in register 15 and an exit is taken. The data is now moved, one character at a time, until the user maximum is reached or an EOB is encountered. If the maximum is exceeded, a return code of 4 is set in register 15 and an exit is taken.

Note: All nonconversational input data obtained via GATRD will be adjusted in the SL to eliminate the key, etc., depending on whether the input was VISAM, VSAM, etc. The line length passed to the user will reflect the actual record. A truncation return code will not be set for a nonconversational task if only blanks exceed the user's buffer.

Error Conditions: The routine returns a hexadecimal code in register 15:

Code Hexadecimal	Significance
00	No errors detected.
04	Number of characters to move exceeds maximum.
08	Attention interrupt detected by GATE.
0C	An immediate command was detected and executed.
10	The input line was in keyboard format and a normal return was made.
14	The input line was in keyboard format and the line was truncated.
20	The input line was in card reader format and a normal return was made.
24	The input line was in card reader format and the line was truncated.
40	SYSIN request no processed because: 1) CKCLS detected error. 2) E marker points to U marker.
100	Continuation detected by GATE.

Note: If continuation is detected, a return code of 10 is returned in byte 2 of register 15.

System Control Block Usage:

New Task Common (CHANTC)
Task Common (CHATCM)
Source List Marker (CHASLM)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)

GNC MACRO: The code generated by GNC initially picks up the character pointed to by the current SA and places it in output register 1. If the character is an EOB (first character of an SL marker), a test is made to determine if it begins an E marker. If an E marker exists at the current SA, control passes to the first instruction past the macro expansion (hereafter called an EXIT); otherwise CZASC3 is called to process the marker. Since CZASC3 determines a new SA, control passes back to examine the new character.

If the character at the current SA wasn't an EOB, the current SA is incremented by one and the same character is tested for a continuation mark. If it's not a continuation character, an EXIT is made; otherwise a test is made to determine if the continuation character is followed by an E marker. If not, an EXIT is made with the continuation character in the output register. If it was, CZASC4 is called to update the SA to the first character in the continuation line and control passes back to process the new character.

An operand may be specified with the GNC macro instruction. This operand specifies the location to get control if a POPIT return code is received after calling CZASC3 of CZASC4. In effect, the operand specifies a special exit.

Programming Notes: NEW TASK COMMON, CHANTC, and the SUBLIST HEADER, CHASLH, must be covered to assemble and use this macro. Use of this macro is currently restricted to the Command Analyzer and Executor, the Source List Handler, PCS, and the PROCDEF Expander.

Control Dictionary Handler (CZASD)

The control dictionary handlers are a set of internal subroutines that operate on any dictionary of the form of the command system control dictionaries. They provide the means to initialize, maintain, and use any dictionary of this form. (See Chart AG.)

ENTRIES:

CZASD1 - STARTFIX
CZASD2 - STARTVAR
CZASD3 - RFR
CZASD4 - NEXTRFR
CZASD5 - ENTR
CZASD6 - DELENT
CZASD7 - EXTDIC
CZASD8 - PACKVAR
CZASDX - GDV

MODULES CALLED: See individual routines.

EXITS: The routines normally return to the calling routine, via the RETURN macro instruction.

OPERATION (GENERAL): Dictionary initialization is performed by either STARTFIX or STARTVAR, depending on the need for fixed-length or variable-length entries. These routines establish the dictionary and hash table in space provided by the user. A HASH macro instruction is provided for determinig the referece value of an eight character entry name. The RFR and NEXTRFR routines provide the capability to locate the appropriate position within the dictionary in order to use, add, replace, or delete an entry. NEXTRFR eliminates the necessity for beginning the search each time. Either one or the other of these muse be used prior to the use of ENTR or DELENT.

ENTR performs the addition or replacement of an entry, and in the process calls EXTDIC as required. EXTDIC provides additional, initialized space. DELENT deletes an entry and makes necessary changes in the chain pointers. PACVAR is available to regain lost space in variable length dictionaries by reforming the dictionary. This function should be necessary only after excessive deletions have occurred.

The GDV routine is called by the GDV macro instruction to retrieve a default value for a parameter name from the combined dictionary. Upon entry, R1 contains the VMA of the parameter name. The name length is preceded by a byte containing the string length, which cannot exceed eight bytes. On exit, the default value is moved to table A. Then R1 is set to point to table A which will contain the default value, if one is found in the combined dictionary. If a default value is not found, R1 is set to 0. The default value is preceded by a byte that contains the string length.

ERROR CONDITIONS: These are given in the individual routines.

SYSTEM CONTROL BLOCK USAGE: These are given in the individual routines.

STARTFIX Routine (CZASD1): This subroutine performs the functions necessary to initialize a dictionary composed of fixed-length entries. It establishes the hash chain and other necessary initial entries. (See Chart AG.)

Entry: CZASD1 - normal entry

Modules Called: None.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: This routine uses the page of virtual storage indicated and sets the hash table entries to point to the available space-chain. Then the entry pointers are initialized for all entry spaces, as required by the length of the entries; finally, entry codes are initialized.

Error Conditions: There are no error conditions for this routine.

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

STARTVAR ROUTINE (CZASD2): This subroutine performs the functions necessary to initialize a dictionary that will contain variable length entries; it also establishes the necessary initial entries. (See Chart AG.)

Entry: CZASD2 - normal entry

Modules Called: None.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: STARTVAR uses the space and length to point the hash table entries to an available space.

Error Conditions: There are no error conditions for this subroutine.

System Control Block Usage: Control Dictionary Heading (CHADCT).

RFR ROUTINE (CZASD3): This subroutine is used to locate a dictionary entry which may be added, replaced, or deleted. (See Chart AG.)

Entry: CZASD3 - normal entry

Modules Called: None.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: After verifying the hash value, the next entry is examined for name and entry code. When the correct entry is located, the location of the entry is returned. Successive entries in the hash chain are cnecked until either a correct one is found or the end of the particular hash chain is reached. The absolute location of the preceding pointer is always retained.

Error Conditions: Register 15 will contain a 0 if the entry was successfully located, and a 4 if no entry was found. It will contain 8 if the hash value was not valid, or the page containing the dictionary is not initialized.

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

NEXTRFR ROUTINE (CZASD4): This subroutine is used to locate an entry in a dictionary without going through the entire hash chain. (See Chart AG.)

Entry: CZASD4 - normal entry

Modules Called: None.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: This routine begins at the point where RFR stopped. Successive entries in the hash chain are checked until either a correct one is found or the end of the particular hash chain is reached. The absolute location of the preceding pointer is always retained, and the location of the entry found is returned.

Error Conditions: Register 15 will contain:

0 - successful location
4 - unsuccessful location
8 - error

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

ENTR ROUTINE (CZASD5): This routine, which is used to add or to replace an entry, is always used following an RFR or a NEXTRFR. If the preceding locating routine was successful, ENTR will replace the old entry. If the preceding locating routine was unsuccessful, ENTR will add the new entry. (See Chart AG.)

Entry: CZASD5 - normal entry

Modules Called: EXTDIC (CZASD7) - To extend the dictionary.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: ENTR checks the preceding pointer, which was retained by RFR or NEXTRFR, and determines whether to replace or to add the new entry. It extends the dictionary, if necessary, by calling EXTDIC and proceeds to perform the addition or replacement. The preceding pointer is changed, and retained, while the location of the entry is returned.

Error Conditions: Register 15 will contain:

0 - successful entry
4 - unsuccessful entry
8 - error

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

DELENT ROUTINE (CZASD6): This routine is used to delete an existing dictionary entry. (See Chart AG.)

Entry: CZASD6 - normal entry

Modules Called: None.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: This routine is used only following a successful RFR or NEXTRFR. It deletes the located entry by setting the entry code equal to 0 and by modifying the next entry pointers of the hash chain. It also changes the retained preceding pointer. The new space is made available in fixed-length dictionaries.

Error Conditions: The retained pointer is checked for zero to prevent the destruction of the sequence. Register 15 contains:

0 - successful deletion
8 - invalid request

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

EXTDIC ROUTINE (CZASD7): This routine is used when it is necessary to provide additional space for the dictionary. It is called by ENTR when there is insufficient space to accommodate a new entry. (See Chart AG.)

Entry: CZASD7 - normal entry

Modules Called:

EXPAND To get additional contiguous space.

GETMAIN To get storage.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: If it is necessary, GETMAIN is used to get additional storage. Then the dictionary is moved to ensure page alignment. EXTDIC calls EXPAND to get additional space to insure contiguous space for the continuation of the dictionary. The additional space is initialized, and the dictionary length and dictionary origin are changed appropriately.

Error Conditions: There are no error conditions for this routine.

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

PACKVAR ROUTINE (CZASD8): This routine is used when it becomes feasible to reclaim lost space which is due to numerous deletions. This is necessary only in the case of variable-length entry dictionaries where deleted entry space is not reused. (See Chart AG.)

Entry: CZASD8 - normal entry

Modules Called: STARTVAR (CZASD2). To start a new dictionary.

Exits: The routine returns to the calling routine, via the RETURN macro instruction.

Operation: PACKVAR uses the dictionary handlers to form a new dictionary and to establish a new hash table which works through each hash chain in turn, until all entries from the existing dictionary are put into the newly formed dictionary. The routine returns the new dictionary origin.

Error Conditions: Register 15 will contain:

0 - successfully packed dictionary
4 - error

System Control Block Usage:

Control Dictionary Heading (CHADCT)
Control Dictionary Entry (CHADEN)

GDV ROUTINE (CZASDX): This routine is called by the GDV macro to determine whether a default value exists in the dictionary for a specified entry. (See Chart AG.)

Entry: CZASDX - normal entry

Modules Called:

RFR (CZASD3) To search the dictionary.

Exits: The routine returns to the calling module via the RETURN macro instruction.

Operation: GDV uses a pointer in register 1 to find a pointer to an eight-byte field containing the name for which the search is to be made. The name is hashed, using the HASH macro. RFR is called, using these as parameters: name, hash value, default code, dictionary location, and the location in which the pointer to the value is to be placed. If a value is found, a pointer to the value is returned.

Error Conditions: Register 1 will contain the VMA of the default value, if one was found. If one was not found, register 1 will contain all zeros. The default value will be preceded by one byte containing the string length.

System Control Block Usage: None.

SECTION 3: TEXT EDITOR

The Text Editor is a collection of routines that provide a facility for creating and manipulating lines of data in a VISAM data set. Communications are provided to allow editing to be performed at the same time that a language processor is compiling or assembling from a source data set.

The principal interface between the Text Editor routines and the rest of the system is provided by the User Controller routine (CZAMZ). (The User Controller serves as an interface to all non-privileged code. All Text Editor routines are nonprivileged except the Edit Initialization and DATALINE routines.)

Most of the Text Editor routines are command processor routines; they are invoked when the specific commands they process are issued by the user. Two of these, LIST (CZASP) and EXCISE (CZASL), are also called by many of the other Editor command processors.

The Translation Table Initialization (TRIN) routine (CZBTA/CZBSY), Translation Table Update (TRUP) routine (CZASS/CZBSX), MATCH (CZAST), and DATALINE (CZASG) are all service routines; they do not process specific commands, but instead serve as common code for the command processors. Figure 9 shows the logical connections between command

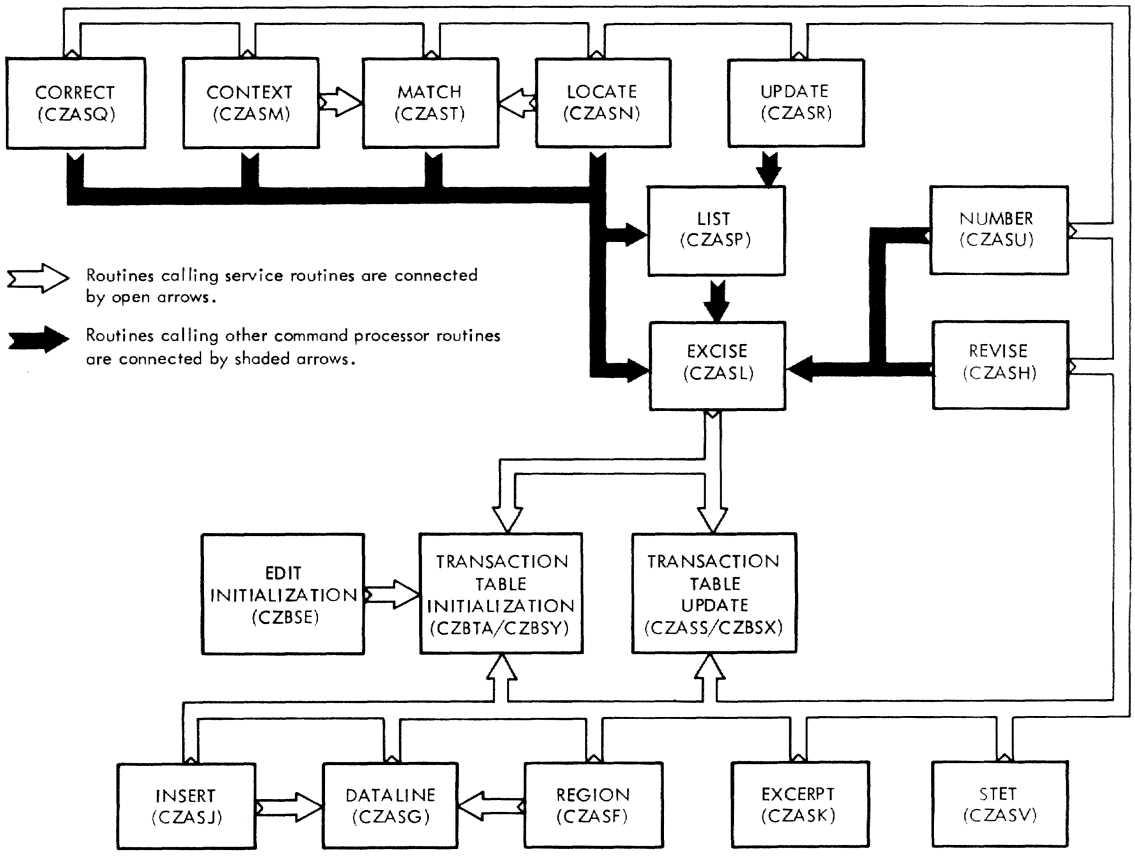


Figure 9. Interrelation of Text Editor Command Processor and Service routines

processor and service routines, and among the command processors themselves.

The Edit Controller (CZATS) is essentially a command processor routine for the EDIT and END commands. It initializes the Text Editor when an EDIT command is issued, and terminates Text Editor processing when an END command is encountered (or another LPC is initiated). The Edit Controller performs a function that is similar to the LPC function for assembler, FORTRAN, PL/I, and linkage editor.

The principal function of the Edit Initialization routine (CZBSE) is to establish a transaction table by means of TRIN and GETMAIN when a LPCINIT macro instruction has been issued. After the Transaction Table has been established, control is returned to the User Controller to process the next command.

The ENABLE and DISABLE commands are not handled by command processor routines in the Text Editor; they are instead entry points in the User Controller routine.

The virtual access method (VAM) is used for the data sets being edited. The DCB for the data set is in the PSECT that is common to all the modules in the Text Editor except the Edit Controller. All Text Editor modules issue a COPY macro instruction; each DSECT (CZBSEDM) generated by this macro maps the Text Editor's common PSECT (CZBSEW).

All of the commands of the Text Editor are entered in the BPKD form. Each command points to a common BPKD that is located in the portion of the User Controller routine that is part of its LPCEDIT routine. The BPKD is located by the User Controller, and executed. The procedure permits a controlled dispatch of many nonprivileged routines under the precise control of the Language Processor Controller which is located in a section of the User Controller routine.
cou

Text Editor Controller -- EDIT Command Routine (CZATS1/CZATS2)

This routine is the language processor controller for the Text Editor. When entered as the result of the EDIT command, this routine initiates the Text Editor and monitors any succeeding Editor commands. When entered as the result of the END command, it terminates Text Editor processing. (See Chart AH.)

ENTRIES:

CZATS1 - entry exclusively from the Command Analyzer through the Edit Controller for the EDIT command
CZATS2 - entry is from the User Controller (CZAMZ) upon recognition of the END command, or if the user initiates another LPC

MODULES CALLED:

Control Dictionary Handler GDV (CZASDX)	Gets default value for REGSIZE.
DDEF (CZAEA1)	Creates a JFCB for the data set, if none already exists.
RELEASE (CZAFJ3)	Releases the data definition, during END processing, if CZATS had to call DDEF during EDIT processing.
FINDDS (CZAEC1)	Determines whether the requested data set has been defined.

FINDJFCB (CZAEB1)	Locates the object data set, if FINDDS cannot find the data set.
REGION (CZASF1)	Validates RNAME, if it has been entered as one of EDIT's parameters.
SCAN Routine CHECKDS (CZAAC2)	Validates the data set name.
User Control (CZAMZ): LPCINIT (CZASW1)	Initiates the Editor as an LPC.
LPCEDIT (CZASW4)	Begins execution of Editor commands.
User Prompter (CZATJ1)	Issues diagnostic messages to the user.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: An overview of the Editor Controller's logic is displayed in Figure 10. The routine must first validate the data set name. When it is assured of a valid dsname, it can determine whether that dsname represents an already defined data set by calling FINDDS. If FINDDS returns without a JFCB, CZATS performs the data definition itself: it creates a ddname, fills in the DCB with default values, calls DDEF to create a JFCB, and then notes that it supplied the DCB parameters. These DCB parameters depend in part on whether REGSIZE has been specified. If REGSIZE is zero, CZATS sets KEYLEN to seven. If there is a value for REGSIZE, CZATS adds seven to it to determine KEYLEN. Values of RECFM=V and RKP=4 are assumed. The value for LRECL is determined by REGSIZE: When REGSIZE=0, LRECL=132; when REGSIZE≠0, LRECL=256.

If FINDDS locates a JFCB, CZATS determines the disposition of the data set. If it is other than "old," CZATS branches to another section of logic to examine the Task Data Definition Table (TDT) and set the attributes of the data set. If KEYLEN in the TDT=0, and REGSIZE is valid, CZATS sets the DCBKEY=REGSIZE+7. If LRECL in the TDT is 0, it sets RECL=132. It tests for record format (RECFM) in the TDT. If there is no setting, or if the setting indicates variable format records, CZATS sets RECFM=V and RKP=4. If RECFM=F, there must not be a setting for RKP in the TDT.

CZATS next tests for and validates a member name, and, after testing the user's qualifications, opens the requested data set either for input or update. A value for REGSIZE may be entered as one of the EDIT command's operands -- CZATS tests for and validates such an occurrence, and sets LRECL in the DCB accordingly. It then tests for a partitioned data set. If this data set is not partitioned, CZATS can now begin the final stage of its processing. However, if the data set is partitioned, CZATS must first deal with it.

If the data set is partitioned, CZATS tries to locate the member and open it via the FIND macro instruction. If the member is found, or has already been opened, and it is a VISAM member, CZATS branches to its final stage. (If in this processing CZATS finds that REGSIZE+7≠DCBKEY, it informs the user that REGSIZE has been set to 8.) If the member is new, and not in SYSULIB, CZATS creates and closes it via the STOW macro instruction. The following DDEF options are present, if the member is in SysuliB, and if the member being processed is either SYSPRO or SYSMLF: RKP=4, KEYLEN=15, RECFM=V, LRECL=256. In any event, the new member is now reopened via the FIND macro instruction.

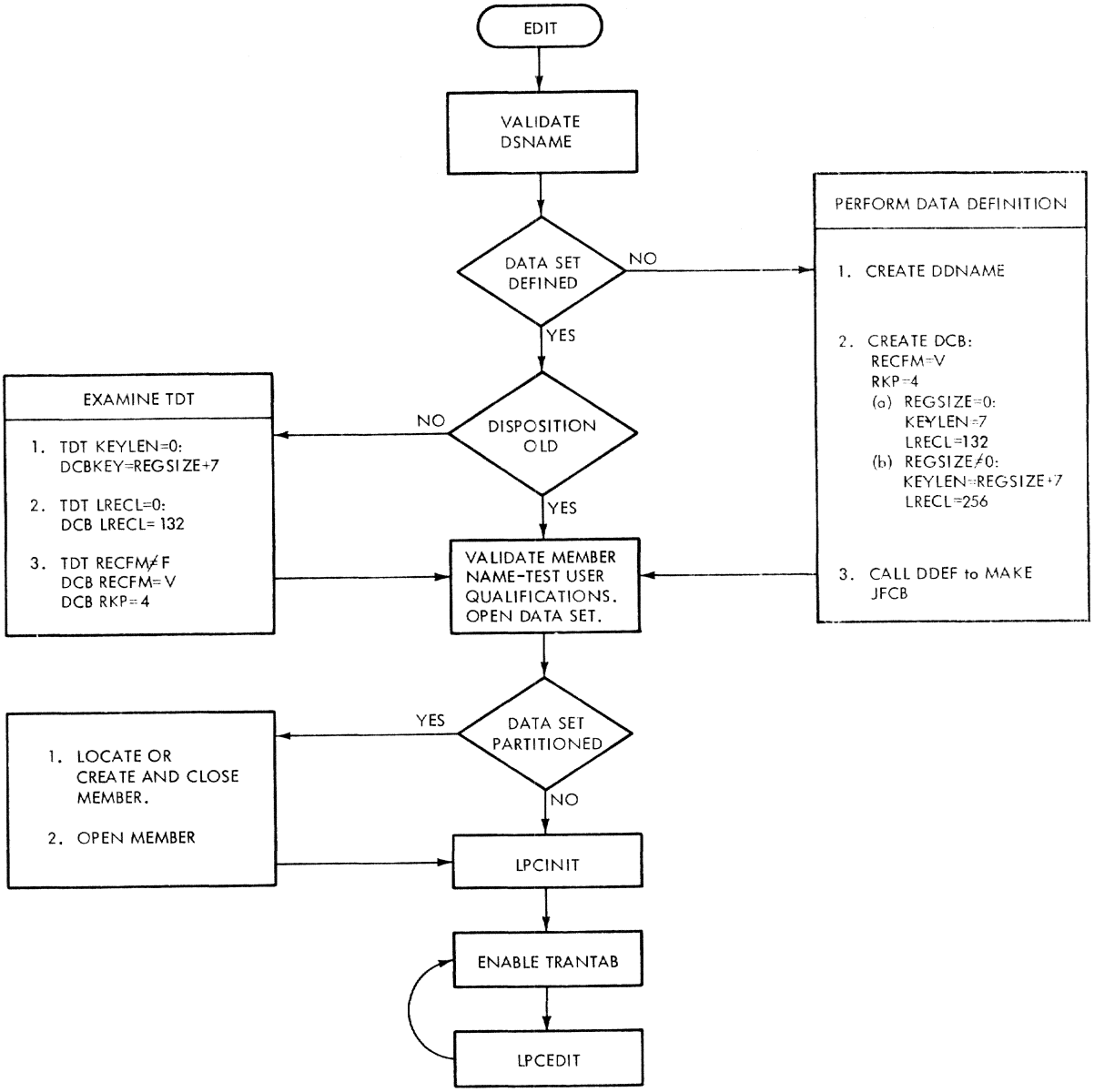


Figure 10. Overview of the Text Editor Controller logic

In its final stage of processing, CZATS determines if it provided the DCB parameters. If not, it ensures that the RECFM and RKP parameters are valid. CZATS then executes the LPCINIT macro to initialize itself as a language processor controller. On return, it determines whether a region name (RNAME) was entered as one of the EDIT command's operands. If so, it calls the REGION processor (CZASF). CZATS then sets the Transaction Table ENABLE/DISABLE switch to enabled. If this is an old region or old line data set, the CLP is set to the first line of the region or data set, and an exit is issued to return the user to command mode. If this is a new region or new line data set, the editing procedure begins with the execution of the LPCEDIT macro instruction, which returns to the command system to await new Text Editor commands. CZATS loops on LPCEDIT until the user terminates the Text Editor with an END command or another LPC. At this point, CZATS2 (END) is called. END closes the object data set (or members if partitioned), releases EDIT's DDEF, if necessary, and exits.

ERROR CONDITIONS: The following conditions will cause a diagnostic message to be issued and an error exit taken:

1. Member name or dsname not supplied.
2. Invalid dsname.
3. Member name longer than eight bytes.
4. Invalid REGSIZE specified.
5. Object data set not VISAM or VPAM.
6. Object data set new and read-only.
7. Object data set VPAM and no member name supplied.
8. Unsuccessful FIND on the supplied member name.
9. Member not VISAM.
10. DDEF unsuccessful.
11. Invalid DCB parameters.
12. Invalid BASE default attribute.
13. Invalid region name (Name).

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Task Definition Table (CHATDT)
Transaction Table (CHATRN)

CONTEXT Command Routine (CZASM)

The routine is entered from the User Controller when a CONTEXT command is encountered. It searches within a range of lines (n1 to n2) and character positions in the object data set for a given string (string 1), and upon finding it, replaces that string with a second string (string 2). (See Chart AI.)

ENTRY: CZASM1 - normal entry

MODULES CALLED:

Control Dictionary Handler - GDV (CZASDX)	Gets default value.
TRIN (CZBTA)	Initializes the Transaction Table.
EXCISE (CZASL2,3)	CZASL2 resolves the beginning line number (n1); CZASL3 resolves the last line number (n2).
TRUP (CZASS1,2)	CZASS1 adds a line to the Transaction Table addition list. CZASS2 adds a line to the Transaction Table deletion list.
MATCH (CZAST)	Searches for the string to be replaced (string 1), and converts hexadecimal input when specified.
User Prompter (CZATJ1)	Issues diagnostic messages to the user.

LIST (CZASP) Converts the substitution string (string 2) for hexadecimal input.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. An error exit to the calling routine is made when the required parameters are not specified.

OPERATION: CONTEXT establishes the range of lines to be searched. It sets up the MATCH routine parameters to start at the given character position for the first line, obtains the first line (via GET), and calls MATCH to search the line for string 1 and convert hexadecimal input when specified. If string 1 is found, it is replaced with string 2. LIST (CZASP3) resolves hexadecimal input for string 2. The match character position is then reset beyond the position where the last string was found and MATCH is reentered until the line is exhausted. The new record is written into the data set. CONTEXT then checks the transaction processed switch (TRNPRO). If the the previous transactions have been processed, TRIN is called to reset the transaction table. Successive lines are located and searched until the entire range of lines has been processed.

If the default value for TRANTAB is Y, in every line where string 1 has been replaced with string 2 at least one time, the original line is entered in the transaction table as a deletion, and the edited line is entered as an addition. When the entire range has been processed, the current line pointer is set to the line following the last line searched (n2), or, if n2 is the last line of the data set or region, to n2 + INCR; CONTEXT then returns to the calling program.

ERROR CONDITIONS: A diagnostic message is issued and an exit taken under the following conditions:

1. String 1 not given.
2. Data set null.
3. n1 or n2 not resolved (message issued by CZASL).
4. n1 > n2 (invalid range).
5. Starting character position greater than ending position.
6. Starting or ending character positions invalid.
7. Error on WRITE operation.
8. Match not found.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)
New Task Common (CHANTC)

CORRECT Command Routine (CZASQ)

The routine is entered by the User Controller when a CORRECT command is encountered. It initializes the Text Editor to accept correction lines from SYSIN, and make corrections within a range of lines and character positions in the data set. (See Chart AJ.)

ENTRY: CZASQ1 - normal entry

MODULES CALLED:

SYSIN (CZASC7,8)	Reads the expected data.
TRUP (CZASS1&2)	Updates the Transaction Table (CHATRN).
TRIN (CZBTA1)	Initializes CHATRN.
User Prompter (CZATJ1)	Issues messages to the user.
EXCISE (CZASL2,3)	Calculates line numbers n1 and n2 as indicated by input values.
LIST (CZASP2)	Converts hexadecimal input data and prints a line on user's SYSOUT.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: CZASQ1 establishes the parameter defaults and replaces the correction markers if the CORMRK parameter has been specified. The first line (n1) is then obtained via a READ-by-key operation, and, if hexadecimal output is specified, CZASP2 is called to convert the data to the EBCDIC representation of hexadecimal. When only one line is specified to be corrected, the user is prompted with the line. SYSIN is called to read the correction line (that is, the string of correction markers).

A scan of both the object line and the correction line is initiated at the column of data specified by the SCOL parameter in the CORRECT command. If none was present, the scan starts at the first character of data.

The scan proceeds character by character until the end of the object line is reached. The first correction line character (CLC) is examined to determine if it is a correction marker. If it is not a correction marker, the CLC is placed in the output line. If it is a duplication marker, the object line character (OLC) is placed in the output line. The input and output byte counts are advanced and the next CLC and OLC are examined in the same fashion.

The relationship of OLC to CLC is summarized as follows:

1. CLC=* All OLCs are duplicated until the next correction marker is encountered.

2. CLC=$ This column is duplicated and all following non-correction-marker characters replace their corresponding OLCs until another correction marker is found.

3. CLC=% This column is removed from the object line, and following object columns are duplicated until another correction marker is reached.

4. CLC=@ This OLC is duplicated, and all succeeding output characters are obtained from a replacement line (read at this time by SYSIN) until an @ (or end of line) is encountered in the replacement line, after which the scanning resumes from the correction line.

5. CLC=# This functions exactly as the @ character except that the characters entered for the replacement line are treated as hexadecimal characters. Only valid hexadecimal characters are accepted.

When the end of the object line is reached, the line is written to the data set; if the default value for TRANTAB is Y, the transaction table will be updated. The transaction processed switch (TRNPRO) is tested. If previous transactions have been processed, TRIN is called to reinitialize the transaction table. CZASQ then calls TRUP to manipulate the addition and deletion lists in the transaction table. On return, the next object line is read (if there is one), all byte counts are reinitialized, and the same correction/replacement lines are applied to the new object line. The CLP is set to the line number after n2, or to n2 + INCR if n2 is the last line of the data set.

ERROR CONDITIONS: The following error conditions will cause a diagnostic message to be issued and an exit taken:

1. Output line exceeds a page.
2. Correction line end found prior to completion. The rest of the output line is left untouched.
3. n1 or n2 cannot be resolved.
4. Hexadecimal input contains invalid character.
5. SCOL value is non-decimal.
6. Invalid range specified.
7. CHAR parameter invalid.
8. SYNAD on WRITE operation.
9. SCOL value greater than record length.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)

DATALINE Entry Service Routine (CZASG)

This privileged routine is entered when data is expected by EDIT, INSERT, REGION, or REVISE. It reads the line using SYSIN, assigns a line number, and places the new line in the data set. (See Chart AO.)

ENTRIES:

CZASG1 - normal entry
CZASG2 - hexadecimal input conversion routine

MODULES CALLED:

TRIN (CZBTA1)	Initializes the Transaction Table.
TRUP (CZASS1)	Places an addition in the Transaction Table.
SYSIN (CZASC8)	Reads a line.
User Controller (CZAMZ1,14)	Dispatches nonprivileged scan routine; cancels scan routine previously terminated by an attention interruption.
User Prompter (CZATJ1)	Prints diagnostic messages.

EXITS: The routine processes lines reiteratively until an unusual condition is detected (such as the CLP exceeding N2MAX, or the recognition of a command) and control is returned to the calling program. See "Error Conditions" for a list of the conditions under which exits are taken.

OPERATION: If a previous scan routine has been terminated by an attention interruption, User Controller (CZAMZ14) is called to cancel it. DATALINE next compares the CLP to N2MAX. If the comparison is high or equal, the User Prompter is called to inform the user that the line already exists; register 1 is set to zero, and control is returned to the calling routine. If the CLP is less than N2MAX, and DATALINE was entered as a result of a REVISE or INSERT command, the increment value is resolved.

The Source List Handler SYSIN routine (CZASC8) is then called via the SYSIN macro instruction to read the line from the source list. If a command or an attention interruption is detected, register 1 is set to zero and the routine exits. Otherwise, the conversion routine (CZASG2) within DATALINE is entered; this routine provides for conversion of hexadecimal data to EBCDIC representation.

On return, DATALINE checks to see if concatenation is required. If CONREC=Y and a concatenation character is detected at the end of the line, CZASC8 is reentered until concatenation is complete.

If there is a value in TRNSCAN, DATALINE calls the routine found there via CZAMZ1 with a two-word parameter list. The first word points to the input string, preceded by a byte indicating the string length; the second word points to a byte in CHBAAA where the scan routine may pass a return code.

If the scan routine indicates that the line is acceptable, DATALINE enters the line into the data set with a WRITE-by-key operation and causes the line to be added to the Transaction Table addition list. The CLP is then updated, and DATLINE loops back to read another line.

ERROR CONDITIONS: Under the following conditions the user is prompted with a diagnostic message, the current record is ignored, and DATALINE reads another line of data:

1. The specified scan routine indicates that the data line was invalid, and the next data line should be read (return code 4 or 8).

2. An error occurs during the WRITE operation, keys equal.

The user is prompted with a diagnostic message and DATALINE exits under the following conditions:

1. CLP equal to or greater than N2MAX.

2. Invalid INCR.

3. An underscore detected at the beginning of the data line (no diagnostic issued).

4. An attention interruption condition detected (no diagnostic issued).

5. Specified scan routine indicates that the data line is invalid and further input should not be accepted (return code null or C).

6. An error occurs during the WRITE operation, keys not equal.

7. The line number plus INCR exceeds 9999999.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
New Task Common (CHANTC)
Profile Character and Switch Table (CHAPCT)
Transaction Table (CHATRN)

Edit Initialization Routine (CZBSE)

This routine initializes the Transaction Table for a new data set. (See Chart AK.)

ENTRY: CZBSE1 - normal entry

MODULES CALLED:

User Prompter (CZATJ1)	Prompts user with current key.
TRIN (CZBTA)	Initializes addition and deletion lists in the Transaction Table.
GETMAIN (CZCGA2)	Gets a page to be shared by all Text Editor modules.
User Controller (CZASW9)	Processes an implicit end.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: The Edit Initialization routine is entered from the User Controller as a result of the execution of the LPCINIT macro instruction. This may be a result of the FTN, ASM, or other language processor invocation. After a SETL is performed to the beginning of the data set, the first record is obtained. If the data set is empty, the CLP will be set to the value specified by BASE (if valid) or to 100; otherwise the CLP is set to the first line of the region or data set.

TRIN is then called to initialize the addition and deletion lines in the Transaction Table. Register 1 is set to zero for a region data set, or to the address of the CZASG1 VCON/RCON pair for a non-region data set. Control is then returned to the User Controller routine.

ERROR CONDITIONS: The routine prompts the user, invokes an implicit end, and returns if the line number or BASE is invalid.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)
New Task Common (CHANTC)

EXCERPT Command Routine (CZASK)

This routine is called by the User Controller when an EXCERPT command is encountered. EXCERPT incorporates a range of lines from another data set into the data set currently being edited. Entry parameters are the name of the data set to be sampled, the region name, and the numbers of the first and last lines to be included. (See Chart AL.)

ENTRY: CZASK1 - normal entry

MODULES CALLED:

TRUP (CZASS1)	Updates the transaction table.

TRIN (CZBTA1)	Reinitializes the transaction table.
User Prompter (CZATJ1)	Prints error messages.
FINDDS (CZAEC1)	Finds JFCB for DSNAME in input parameter.
SCAN Package (CZAAC)	Analyzes input parameters.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: On entry, if no data set name is input, the user is prompted and an error exit is taken. CZASK then validates the region name, if supplied, and issues a CHECKDS macro to validate the data set name. If the data set from which the data is to be taken is the same as the target data set, the target data set is first closed and then opened for update. CZATS next calls CZAEC1 to locate the JFCB, and on return, opens the included data set.

The line numbers parameters are examined to determine what part of the data set is to be included:

1. If both line numbers are defaulted, it is assumed that the entire data set is to be included. A SETL to the beginning is performed; n2 is set to LAST.

2. If only a region name is present, the entire region which has been named is included. A series of SETLs is used to find the first and last lines of the region; n1 and n2 are initialized accordingly.

3. If only n1 is present, the single line referred to is included; both n1 and n2 are set to the same key.

4. If both n1 and n2 are explicitly given, the range of lines from n1 to n2 is included.

After establishing the range of inclusion, the included lines are placed in order. Each line is renumbered using BASE and INCR and a check is performed to ensure that N2MAX has not been exceeded. The included lines are then written into the data set using WRITE-new-key. The transaction processed switch (TRNPRO) is tested to see if previous transactions have been processed. If so, TRIN is called to initialize the Transaction Table prior to entering the first line. As each line is written, it is entered in the Transaction table as an addition if the default value for TRANTAB=Y. When the task has been completed, EXCERPT calls CLOSE to close the sampled data set.

ERROR CONDITIONS: Should the included renumbered lines overflow the maximum (N2MAX), the overflow lines will be ignored and the user will be prompted with a diagnostic message. The records already processed are deleted, and control is returned to the calling routine.

If the range of lines or region indicated is not found in the input data set, or if the data set or region does not exist, the entire command will be ignored and the user will be prompted with a diagnostic message.

If the data set from which lines are being included ends before the line number specified by n2 is encountered, the user is informed by a diagnostic message. The routine exits to the calling program.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Task Data Definition Table (CHATDT)
Transaction Table (CHATRN)

EXCISE Command Routine (CZASL)

This routine is called by the User Controller when an EXCISE command is encountered; it is also called by other Text Editor routines to resolve the user's references to line numbers in the data set being edited. EXCISE deletes a range of lines from the data set. Entry parameters are the first and last lines to be deleted. (See Chart AM.)

ENTRIES:

CZASL1 - entry point to delete lines from the data set
CZALS2 - entry point to resolve n1 of a range of lines
CZASL3 - entry point to resolve n2 of a range of lines
CZASL4 - entry point to resolve one number not part of a range

MODULES CALLED:

EXCISE (CZASL1,2,&3)	Calculates line numbers n1 and n2 as indicated by input values.
TRUP (CZASS2)	Defines linkage to this routine.
TRIN (CZBTA1)	Initializes addition and deletion lists.
User Prompter (CZATJ1)	Issues diagnostic messages to the user.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: When entered at CZASL1, EXCISE resolves any defaults in the line number parameters. Each line in the indicated range is then deleted by calling DELREC. The transaction processed switch (TRNPRO) is tested. If previous transactions have been processed, TRIN is called to reinitialize the Transaction Table. A deletion entry is made for each line in the Transaction Table. The CLP is set to the first line deleted and N2MAX is set to the first line beyond the deletion range if there is a line, or 9999999 if there is no line.

When EXCISE is entered at CZASL2, CZASL3, or CZASL4, it is determined whether the line number the user specified was relative or absolute. If absolute, the line number is resolved by using a SETL by key. If the line number is relative, SETL NEXT or SETL PREVIOUS is used to arrive at the requested line.

If CZASL is able to successfully resolve the line number, it stores the address of the key for the record in the location that was specified in register 4 by the calling program. Otherwise, register 2, which upon entry contained the address of the input line, is cleared to zeros.

ERROR CONDITIONS: The user is issued a diagnostic message and an exit is taken under the following conditions:

1. Only one line is specified and it does not exist in the data set.
2. No lines exist in the current region or line data set.
3. n1 or n2 specified as a plus-or-minus integer, and there are not as many lines from the CLP in the region or data set.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)

INSERT Command Routine (CZASJ)

INSERT, which is entered from the User Controller when an INSERT command is encountered, prepares the Text Editor to accept data lines for insertion following a given line in the source data set. (See Chart AN.)

ENTRY: CZASJ1 - normal entry

MODULES CALLED:

DATALINE (CZASG) Reads the expected data.

EXCISE (CZASL4) Resolves the line number.

User Prompter (CZATJ) Prints messages to the user.

TRIN (CZBTA1) Initializes the Transaction Table.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction. Register 1 contains the address of the CZASG1 VCON/RCON pair to indicate that data is expected.

OPERATION: The data set is initialized so that subsequent new data lines are inserted following line n1. N2MAX is set to the next existing line after n1. There is no deletion of existing lines.

ERROR CONDITIONS: If a line number is entered which EXCISE cannot resolve, or if an invalid increment is specified, the user is prompted and the command is canceled.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)

LIST Command Routine (CZASP)

The routine is entered by the User Controller whenever a LIST command is encountered. It lists a range of lines from a data set and places them on SYSOUT. Entry parameters are either the current line pointer, or the first (n1) and last (n2) lines to be listed (along with optional starting and ending character positions) plus the mode (CHAR) in which the lines are to be listed. (See Chart AP.)

ENTRIES:

CZASP1 - normal entry
CZASP2 - hexadecimal input conversion entry
CZASP3 - hexadecimal output conversion entry

MODULES CALLED:

Control Dictionary
Handler - GDV (CZASDX) Gets default value for LINENO.

EXCISE (CZASL2,3) Calculates the limits of the area to be listed.

GATWR,GTWRC (CZATC) Writes records to user's SYSOUT.

User Prompter (CZATJ1) Issues diagnostic messages to the user.

EXITS: CZASP1 and CZASP2 exit normally when their function is completed; register 1 is set to zero, notifying the Text Editor that additional data is not expected. Control is passed to the calling routine via the RETURN macro instruction.

OPERATION: If the first parameter (n1) is specified as CLP, the current line pointer (in CHATRN) will be listed and CZASP1 will exit. If n1 (not specified as CLP) and n2 are specified, CZASP will resolve any character positions, then call CZASL to resolve the n1 and n2 values. If n2 is defaulted, only n1 is listed. If n1 is defaulted and n2 is specified, n1 will be set to the CLP. If both n1 and n2 are defaulted, the entire data set will be listed. The CHAR parameter is checked for validity (valid values are H, C, and M). If it is defaulted, and there is no value in the combined dictionary, C is assumed.

GET is used to retrieve the lines, and GATE is used to list them on SYSOUT. A maximum of 256 characters are listed from each line. Data will be listed as specified by the CHAR parameter in character, hexadecimal or mixed (character and underscored hexadecimal) format. The CLP is set to the line number after n2, if one exists; if n2 is the last line, the CLP is set to n2 + INCR.

Entry point CZASP2 lists lines on SYSOUT, or converts data for LOCATE and CORRECT.

Entry point CZASP3 converts hexadecimal input data from the EBCDIC representation.

ERROR CONDITIONS: The user is prompted and an error return taken when n1 is greater than n2, n1 or n2 cannot be resolved, an invalid character position is specified, the CHAR parameter is invalid, or the data set is empty.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
New Task Common (CHANTC)
Transaction Table (CHATRN)
Task Common (CHATCM)
RESTBL Header (CHARHD)

LOCATE Command Routine (CZASN)

The routine is entered from the User Controller when a LOCATE command is encountered. LOCATE searches a range of lines in a data set for a given character string. Parameters to LOCATE are the first (n1) and last (n2) line numbers to be searched (along with optional starting and ending character positions), and the string to be located. (See Chart AQ.)

ENTRY: CZASN1 - normal entry

MODULES CALLED:

Control Dictionary Handler - GDV (CZASDX)	Gets default value.
EXCISE (CZASL2,CZASL3)	Calculates the limits of the area to be searched.
LIST (CZASP2)	Prints the line containing the string.

MATCH (CZAST1) Searches a line for the desired string.

TRIN (CZBTA1) Initializes CHATRN.

User Prompter (CZATJ1) Issues diagnostic messages to the user.

EXITS: After printing the current key value the routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: LOCATE resolves the entry parameters. If all of the parameters are defaulted, or if the string is not found, the CLP is set to the line following the last line searched (n2), or, if n2 is the last line of the data set, to n2 + INCR.

If n1 and n2 are defaulted, the entire data set is searched. If n2 is defaulted, only the line specified by n1 is searched. Both n1 and n2 may have character positions specified. These positions will apply to the entire range to be searched. LOCATE retrieves the record via VISAM GET (locate), then calls MATCH to search for the string. If the string is found, it is printed using LIST, and the CLP is set to the line containing it. (The CHAR value in the combined dictionary will determine the mode of the output. If there is no value, C is assumed.)

ERROR CONDITIONS: Under the following conditions the user is issued a diagnostic message and control is passed to the calling routine via the RETURN macro instruction:

1. No string specified.
2. Specified string not found.
3. Data set empty.
4. n1 greater than n2.
5. n1 defaulted, but CLP set to non-existent line.
6. n1 or n2 not resolved (no message issued by LOCATE).
7. Invalid starting or ending character positions.
8. Specified starting character position greater than ending position.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)
New Task Common (CHANTC)

MATCH Service Routine (CZAST)

This routine is used to search a line, or a portion of a line, for a given string. (See Chart AR.)

ENTRY: CZAST1 - normal entry

MODULES CALLED:

LIST (CZASP3) Converts to hexadecimal output.

Control Dictionary
Handler - GDV (CZASDX) Gets default value for HEXSW.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: If a conversion from EBCDIC to hexadecimal is required, the LIST routine (entry point CZASP3) is called. The given line is searched character by character until a match with the given string is found, or a match has not been found when the limit of the given line is reached. The result of the search is placed in register 2 and passed to the calling routine. If a match was not found, this result is zero; if a match was found, register 2 will contain the character position within the line where the matching string begins.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE: New Task Common (CHANTC)

NUMBER Command Routine (CZASU)

NUMBER is called from the User Controller when NUMBER command is encountered. NUMBER is used to renumber a range of lines in the data set. (See Chart AS.)

ENTRY: CZASU1 - normal entry

MODULES CALLED:

EXCISE (CZASL2,3)	Resolves line number n1 and maximum number n2.
TRUP (CZASS1,3)	Updates addition and deletion lists.
TRIN (CZBTA1)	Initializes the Transaction Table.
User Prompter (CZATJ1)	Issues diagnostic messages to the user.
GETMAIN	Obtains pages to store data set while numbering.
FREEMAIN	Releases pages acquired with GETMAIN.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: NUMBER determines whether both line number parameters, n1 and n2, have been defaulted. If so, n1 is set to the first record of the data set or region, and n2 is set to the last record. If n2 has been specified but n1 has been defaulted, n1 is set to the value of the current line pointer; if n1 has been specified but n2 defaulted, n2 is set to n1. EXCISE (CZASL2) is called to resolve line number parameters that have been specified. Unspecified BASE and INCR parameters are set to n1 and 100, respectively.

The records are read via GET in the locate mode. The transaction processed switch (TRNPRO) is tested to determine whether previous transactions have been processed. If they have, TRIN is called to initialize the transaction table. The record is then put in the deletion list to be deleted with DELREC-by-key. As the records are deleted, the number-of-lines counter is incremented. This is to insure all records will fit between BASE and limiting line. If they cannot, recovery is initiated by WRITE with the old key, and the command is cancelled. The lines are added to the addition list and placed with WRITE-new-key in the object data set until all of the lines are renumbered.

If INCR is specified in such a way that it causes the limiting line number to be exceeded, NUMBER calculates a new increment by dividing the difference between BASE and the limiting line by the count of lines to be renumbered:

Resulting quotient:	Increment furnished:
100	100
50 - 99	50
20 - 49	20
10 - 19	10
5 - 9	5
2 - 4	2
1	1

The user is then informed that the Text Editor has furnished the increment, and execution continues.

ERROR CONDITIONS: If the BASE or INCR are incorrectly specified or invalid, the user is prompted and NUMBER exits to the calling routine.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)

Profile Handler Command Routine (CZASZ)

This privileged, reenterable, and nonresident routine saves the virtual storage user profile in USERLIB when a PROFILE command is executed. (See Chart AT.)

ENTRY: CZASZ1 - normal entry via BUILTIN call mechanism from the Command Analyzer and Executor.

MODULES CALLED:

DICTEXTR (CZBTBX) Adds entries to the primary dictionary.

STARTVAR (CZASD2) Initializes New Primary Dictionary in buffer.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: The Profile Handler routine saves a session profile by copying it into USERLIB from virtual storage when a PROFILE command is issued. Entry is via the BUILTIN procedure call mechanism. The profile to be copied is constructed in a contiguous virtual storage area. The Character Translation Table, pointed to by PCTCTT, is moved into the PFLCTT area. The output character translation character, pointed to by PCTOCT, is moved into the PFLOCT area. The Profile Character and Switch table is moved from New Task Common into the PFLPCT area.

The Primary Dictionary, starting at PFLPDC, is reinitialized using the STARTVAR routine of the Control Dictionary Handler. Then the Combined Dictionary is searched, extracting all Primary Dictionary entries, which include synonym, default and command symbol definition entries. The extraction of command symbols is optional, based on the setting of the Command Symbol Switch. This switch setting originates as a PROFILE command parameter, which is passed as a calling parameter to the Profile Handler. The extracted entries are added to the empty Primary Dictionary in the profile through a call to the DICTEXTR routine (CZBTBX).

The USERLIB DCB is located, and the DCBLRE field set with the overall length of the profile. A FIND is issued to locate the SYSPRX member in the USERLIB. If one exists, it is deleted and replaced by the new profile, using a VSAM PUT. A STOW is issued to update the POD, and control is returned to the calling routine.

ERROR CONDITIONS: A code 1 ABEND is taken if an error return code is received following the FIND, STOW (Type D) or STOW (Type N) macro instructions.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Control Dictionary Header (CHADCT)
User Profile (CHAPFL)
New Task Common (CHANTC)
Profile Character and Switch Table (CHAPCT)

REGION Command Processor (CZASF)

This routine is entered from the User Controller whenever a REGION command is encountered; it is also entered when the RNAME parameter has been specified in an EDIT command. It locates or initializes a region of lines within the data set having a common region name (string prefix to the line number). The entry parameter is a 0 - 240 character string which is saved in the Transaction Table as a region name, and is prefixed to all subsequent line numbers until another REGION is encountered. If the entry parameter is defaulted a null string prefixes the subsequent line numbers, provided that the data set has regions. (See Chart AU.)

ENTRY: CZASF1 - normal entry

MODULES CALLED:

DATALINE (CZASG)	Reads the expected data.
User Prompter (CZATJ1)	Prints error messages and gives the current key to the user.
TRIN (CZBTA1)	Initializes the Transaction Table.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction, with register 1 containing the address of the CZASG1 VCON/RCON pair to indicate that data is expected. The error exit is to the calling routine when the data set has no regions although a region has been specified. Normally, the Text Editor informs the user of the current key by means of the User Prompter.

OPERATION: The DCB opened for the data set is retrieved from the transaction table and SETL is called to find the location of the key having the maximum number in the given region. The key is tested to determine if this is in the given region. If it is not, no region of that name exists, and the CLP is determined as follows: If a BASE exists, the CLP is set to the value of BASE; otherwise, a GDV will be done for BASE and the default value used to set the CLP. If no default exists, 100 is assumed. If the region does not exist, the CLP is set to the first line of the region.

In the case where the region name is defaulted a name of all blanks is assumed if the data set has regions. A data set might not have regions, and in this case the key is a seven-digit number which can be initialized or added to as such. If this were the case, it would be unnecessary to use a REGION command. After a region has been located or initialized, REGION exits with the address of the CZASG1 VCON/RCON pair in register 1 to indicate data is expected.

ERROR CONDITIONS: If the user has specified a region name, the length of the key (DCBKEY) of the data set is tested. If it is equal to 7, the data set key has no region name, and the user is informed with a message issued by the User Prompter.

If the length of the specified region name plus 7 is greater than the length already specified for the data set key (DCBKEY), the specified region name is truncated to conform and the user is informed with a message issued by the User Prompter.

If a record is found with a key corresponding to the maximum key of the region (Region 9999999), the region is assumed to be full and the user is prompted with two messages issued by the User Prompter.

Note: No records can be added to the data set until the user is more specific as to where records can be inserted.

If the data set key is less than 7, the user receives a diagnostic error message.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)
Task Data Definition Table (CHATDT)

REVISE Command Routine (CZASH)

This routine is entered from the User Controller whenever a REVISE command is encountered. (See Chart AV.)

ENTRY: CZASH1 - normal entry

MODULES CALLED:

EXCISE (CZASL) Deletes specified lines.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: REVISE prepares the Text Editor to accept data for inclusion in the data set at a given point. If an increment is provided, it is stored in INCR for later reference by CZASG. If it is not provided, an increment of 100 is the default. CZASL1 is entered to resolve the line numbers, and delete the lines. REVISE exits with the address of the CZASG1 VCON/RCON pair in register 1 to indicate that data is expected.

ERROR CONDITIONS: The user is prompted if the specified increment is invalid.

SYSTEM CONTROL BLOCK USAGE:

Transaction Table (CHATRN)

STET Command Routine (CZASV)

The routine is entered by the User Controller when a STET command is encountered. If TRANTAB=Y, it restores the data set to its condition prior to the most recent set of unprocessed transactions. If TRANTAB=N, STET will return to its caller. If the Text Editor is disabled, all of the editing commands entered since it was last enabled will be reversed. If the Text Editor is enabled, only the last transaction is reversible. (See Chart AW.)

ENTRY: CZASV1 - normal entry

MODULES CALLED:

TRIN (CZBTA1) Initializes the Transaction Table.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: The STET routine restores the data set to its previous condition by using DELREC and WRITE. The key of the line in the addition list (KEYA) is compared to that of the line in the deletion list (KEYD). If KEYA < KEYD, DELREC is called to delete the line of KEYA from the data set. If KEYA > KEYD, WRITE is called to write the line of KEYD into the data set. If the record already exists in the data set, the user's issued a message by the User Prompter. If KEYA = KEYD, WRITE is called to replace line KEYA with line KEYD. When the transactions have been processed, the pointer to the deletion list is exchanged with the pointer to the addition list. If there are no transactions, or if there is no Transaction Table (TRANTAB=N), this command has no effect on the data set.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)

Transaction Table Initialization Routine (CZBTA/CZBSY) - TRIN

These routines are called by other Text Editor routines to reinitialize the addition and deletion lists in the Transaction Table. This is normally done either when a new data set is to be processed or when a set of transactions have been processed by a language processor. CZBTA is nonprivileged; CZBSY is privileged - which eliminates the need for the Text Editor to use SVC-type linkage. (See Chart AX.)

ENTRIES:

CZBTA1 - normal entry for CZBTA
CZBSY1 - normal entry for CZBSY

MODULES CALLED:

FREEMAIN Releases pages used for additions and deletions.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: FREEMAIN is called iteratively to release all but the first page of the addition and deletion lists. The Transaction Table containing the pointers to the addition and deletion lists is never on the same page as the additions and deletions; therefore, its page is never released. Upon completion of the routine the list pointers in CHATRN are zeroed before control returns to the calling routine.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE: Transaction Table (CHATRN)

Transaction Table Updater Service Routine (CZASS/CZBSX) - TRUP

The Transaction Table Updater is entered to add a record to the addition list or deletion list. CZASS is nonprivileged; CZBSX is privileged - which eliminates the need for the Text Editor to use SVC-type linkage. (See Chart AY.)

ENTRIES:

CZASS1 - entry point in CZASS for an addition to the addition list
CZASS2 - entry point in CZASS for an addition to the deletion list
CZBSX1 - entry point in CZBSX for an addition to the addition list
CZBSX2 - entry point in CZBSX for an addition to the deletion list

MODULES CALLED:

GETMAIN Obtains a page for the record.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: The Transaction Table Updater gets the address of the next available space (TRNNLK) from the Transaction Table, and determines whether the record will fit in the available page. If not, another page is acquired with GETMAIN and the table pages are chained. The record is moved into the available space and TRNNLK is updated.

If the record represents an addition or deletion entry, the corresponding list is updated so that the records in the list are pointed to in sequence. Also, a deletion entry causes the addition list to be examined and, if a record which is identical to the deletion entry is found, it is deleted from the addition list.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE: Transaction Table (CHATRN)

UPDATE Command Routine (CZASR)

This routine is entered from the User Controller whenever an UPDATE command is encountered. The data set must have been opened prior to entry and the DCB address stored in the Transaction Table. The current region is updated with input lines read by SYSIN. (See Chart AZ.)

ENTRY: CZASR1 - normal entry

MODULES CALLED:

Control Dictionary Handler - GDV (CZASDX)	Gets default value.
SYSIN (CZASC8)	Reads a line.
LIST (CZASP3)	Converts hexadecimal data.
TRUP (CZASS1)	Updates the Transaction Table (CHATRN).
TRIN (CZBTA1)	Initializes the Transaction Table (CHATRN).
User Controller (CZAMZ1,14)	Dispatches nonprivileged scan routine; cancel scan routine previously terminated by an attention interruption.
User Prompter (CZATJ1)	Issues a diagnostic message to the user.

EXITS: UPDATE sets register 1 to zero and exits to the calling routine when a command or attention interruption is recognized. See "Error Conditions" for a list of other situations in which exits are taken.

OPERATION: Upon entry, UPDATE checks to ensure that a language processor mode exists, and determines if a previous scan routine has been terminated by an attention interruption. If such a scan routine exists,

the User Controller (CZAMZ14) is called to cancel it. UPDATE next issues an initial prompt to the user, requesting line numbers and data.

The routine then checks for variable-length records. If the records are not variable length, they are padded with blanks. UPDATE then enters the Source List Handler SYSIN routine (CZASC8) via the SYSIN macro instruction to read the line. If no line, line number, or blank stop character was supplied, the user is prompted with a diagnostic, and UPDATE reads another line from the SYSIN device. If hexadecimal input is recognized, UPDATE calls CZASP3 to convert it to EBCDIC representation.

The line is next checked for concatenation characters. If CONREC=Y and a concatenation character is detected at the end of the line, CZASL8 is again entered until concatenation is complete.

UPDATE checks TRNSCAN to see if a scan routine has been specified. If there is a value in TRNSCAN, the routine found there is called via CZAMZ1 with a two-word parameter list. The first word points to the input string, preceded by a byte indicating the string length; the second word points to a byte in CHBAAA where the scan routine may pass a return code.

If the scan routine indicates that the line is acceptable, UPDATE writes the record into the data set with a WRITE-new-key. If a line already exists with the key, the line is processed by a WRITE-replace-by-key. If there are no write errors, UPDATE checks the transaction processed switch (TRNPRO). If previous transactions have been processed, TRIN is called to reinitialize the Transaction Table and TRUP to update it.

UPDATE then loops back to read another line. No prompt for this line is issued. UPDATE exits only when it encounters the break character preceding a command as input, or when it recognizes an attention interrupt.

ERROR CONDITIONS: Under the following conditions the user is issued a diagnostic message, the current record is ignored, and UPDATE reads the next data line:

1. No line or line number entered.
2. No blank stop character separates the line number and the data.
3. The indicated line number is greater than seven digits.
4. The specified scan routine indicates that the data line was invalid and the next data line should be read (return code 4 or 8).

Under the following conditions the user is issued a diagnostic message and UPDATE exits:

1. The routine is entered with no language processor active.
2. An error occurs on a READ or WRITE operation while updating the record.
3. The specified scan routine indicates that the data line was invalid and further input should not be accepted (return code null or C).

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Transaction Table (CHATRN)

SECTION 4: COMMAND ROUTINES

This section describes command routines supported by the command system. Each of these routines carries out the actions requested by a command, and usually does this by calling upon system service routines in a prescribed sequence. Figure 11 lists the names and functions of the command routines. This section also describes the four support routines that are sometimes employed during command processing. These are:

- ABEND routine (to handle abnormal task termination)
- FINDDS routine (to find or create the job file control block for a specified data set)
- FINDJFCB routine (to locate the job file control block associated with a specified data definition name)
- PAIR routine (to assist in abnormal task termination)

General operation of the command routines is given in the following sequence. Typically, the Command Analyzer links to the command routine when its corresponding command name appears on SYSIN. The command routine then employs the SCAN routines to isolate and validate each operand of the command. If diagnostic or prompting messages are needed, the routine issues them via the User Prompter. When the operands prove acceptable, the command routine carries out the requested actions, often by a series of calls to system service modules, such as OPEN or CLOSE, issues (via User Prompter) any response or confirmation messages needed, and finally returns control to the Command Analyzer.

Command routines can be called directly by other command routines or, in some cases, via macro instruction usage. The called routine operates in the same general way, but returns to its calling routine upon completion.

The PUNCH, RT, WT, and PRINT commands are an important exception to the usual pattern. When the Command Analyzer encounters one of these commands, it calls the BULKIO preprocessor. The preprocessor then validates the command operands and, if they are acceptable, sends a request for task initiation to the batch monitor. Acting upon this request,

Routine Name	Symbolic Name	Purpose
ABEND	CZACP	To terminate an abnormal task.
BACK	CZABC	To switch a conversational task to nonconversational mode.
CDD	CZAFS	To execute one or more prestored DDEF commands.
CANCEL	CZABJ	To terminate a specified nonconversational task.
CATALOG	CZAEI	To create or to update the catalog entry for a data set.

Figure 11. Command routines (part 1 of 3)

Routine Name	Symbolic Name	Purpose
CDS	CZAFV	To make a copy of a specified data set.
CHGPASS	CZATI	To change password.
CONVERT	CZCQV	To convert VAM 1 formatted volumes to VAM 2 format.
DATA	CZADF	To build a VISAM or VSAM data set.
DSS?/PC?	CZAEL	To present status information about one or more data sets.
DDEF	CZAEA	To define a data set and describe its characteristics to the system.
ERASE/DELETE	CZAEJ	To erase a data set on public storage and to delete the catalog entry for a data set.
EVV	CZCFB	To catalog VAM data sets which reside on private volumes and were produced on one system to allow their use on another system.
EXECUTE	CZABB	To initiate a nonconversational task.
FINDDS	CZAEC	To search the task data definition table for a job file control block.
FINDJFCB	CZAEB	To locate the job file control block for a given data definition name.
FLOW	CZAGD	To regulate the number of different task types.
IF String Comparison	CZBLT	To provide fast substitute for PCS processing of the IF verb when EBCDIC strings are being compared.
JOIN	CZAFK	To grant a user access to the time sharing system.
KEYWORD	CZATH	To display keywords or command parameters in user's SYSLIB and USERLIB.
LINE?	CZAEM	To present one or more lines from a specified data set.
LOGOFF	CZAFN	To log a task off the system.
LOGON	CZAFM	To begin a task logon.
LOGON2	CZBTB	To complete a task logon.
MCAST/MCASTAB	CZATU	To make changes in the profile character, switch table, and translation table.
MODIFY	CZAEG	To alter, delete, and insert lines in a VISAM data set.

Figure 11. Command routines (part 2 of 3)

Routine Name	Symbolic Name	Purpose
MSGWR	CZAAD	To issue system messages and fetch system responses.
PAIR	CZACS	To add and to remove addresses of interlock release routines in the ABEND Interlock Release (AIR) table.
PERMIT	CZAFH	To indicate that sharing of a data set or catalog level is permitted to other users.
POD?	CZCOX	To print the member names of individual members of a cataloged VPAM data set.
PROCDEF	CZATP	To maintain the procedure library.
Procedure Expander	CZATE	To expand a procedure name in the source list.
QUIT	CZAFL	To deny a user access to the time sharing system.
RET	CZAEN	To modify the mode field in the DSD and the equivalent JFCB field.
RPS/CVV	CZAXX	To recreate public storage.
RELEASE	CZAFJ	To destroy the job file control block(s) for one or more data sets defined during the current task.
SECURE	CZAFU	To secure all devices needed for the execution of a nonconversational task.
SHARE	CZAFI	To allow a user to share a data set or catalog level.
SYN/DEF	CZATR	To enter or to delete synonym or default values in the combined dictionary.
SYSXPAT	CZATF	To pass parameters in their original textual form.
TIME	CZAVB	To specify a time interval for task completion.
UPDTUSER	CZAGC	To initialize the counts of data sets for the system and for each user.
USAGE	CZAGB	To display user accounting information on SYSOUT.
User Control	CZAMZ	To preserve command system interfaces with nonprivileged programs.
VAM TAPE	CZAET	To process VAM data sets.
VSS	CZAVR	To determine if user may use the Virtual Support System (VSS).

Figure 11. Command routines (part 3 of 3)

the batch monitor will initiate the PUNCH, PRINT, WT, or RT command routine as a new nonconversational task, independent from the task that originally issued the command. In this way PUNCH, RT, WT, and PRINT functions -- which depend upon the availability of input/output devices -- can be performed at the system's convenience, and the user's task does not have to wait. A full description of nonconversational operations and the PUNCH, PRINT, WT, RT, ASNBD, FORCE, HOLD/DROP, MSG/BCST, REPLY, SHUTDOWN, and XWTO routines is contained in the Operator Task and Bulk I/O PLM, GY28-2047.

ABEND (CZACP, CZACQ, and CZACR)

The abnormal task termination (ABEND) procedure terminates a logical task sequence at the point the error was detected. (See Chart BA.)

ENTRIES:

CZACP1
(for ABEND routine I-CZACP) - normal entry point, generated by ABEND macro instruction

CZACP2 - entered for ABENDREG to print diagnostic message

CZACP3 - entered on VDMEP call

CZACQ1
(for ABEND routine II-CZACQ) - main entry point. This module is only entered from CZACP

CZACR1
(for ABEND routine III-CZACR) - entry point used only by a QLE which was queued by the external interrupt processor as a result of the VSEND executed by CZACQ of the original task

CZACR2
(for ABEND routine III-CZACR) - parameter list containing address of MCB, used by CZACP

MODULES CALLED: The major ABEND routine, CZACP, is heavily dependent on supervisor and access methods routines to fulfill its function. These are the supervisor routines it calls:

PURGE (CEAALP)	Removes any I/O requests made by the task.
XTRACT (CEAH03)	Extracts the taskid, conversational flag, and I/O pending flag from the TSI.
LVPSW (CEAHQP2)	Loads the VPSW and unmasks any interrupts.
TSEND (CEAH19)	Ends the task's time slice until the task's pending I/O requests are removed.
RESET (CEAAHR)	Resets devices in the task's symbolic device list which were disabled during PURGE.

These are the access methods routines called by CZACP:

VAM ABIR (CZCQQ1)	Releases interlocks set by VAM routines. ABIR= ABEND Interlock Release.
STOW (CZCOK1)	Modifies SYSCAT member in the POD to reset the lock.
INTLK (CZCOH1)	Locks RESTBL while ABEND is using it.

RLINTLK (CZCOI1) — Unlocks RESTBL. Note: INTLK and RLINTLK set and release, respectively, a read or write interlock on a shared data set entry.

FREEQ (CZCTC6) — Logically disconnects any multiple terminals for MTT tasks.

DELVAM (CZCFT1) — Deletes a VAM data set (marks all its DSCB and data page entries "available" in the page assignment table.)

CLOSE COMMON (CZCLB) — Logically disconnects the data set from the problem program, closes the DCB and relinquishes main storage.

DUPCLOSE (CZCEZ1) — Closes a duplexed data set.

Other modules outside the command system also called by ABEND (CZACP) are:

DISCONNECT (CZHNEB) — Disconnects VSS from the task.

RELEAS (CZCAD1) — Releases any private devices.

Loader (CZCDL15) — Determines the program that called ABEND.

Loader Logoff (CZCCD) — Unlinks the task from the shared data set table.

Scratch (CZCES1) — Cleans up the DSCB.

ITI — Inhibits task interrupts.

PTI — Allows task interrupts.

SRCHAUL — Searches the Active User List.

(ABEND also uses READ, WRITE, WTL, WTO, and VSEND.) This ABEND module (CZACP) also calls modules within the command system:

Control Dictionary Handlers
GDV (CZASDX) — Gets the default value for DIAGREG.

GATE (CZATC1) — Clears buffer.
(CZATCB) — Writes the standard ABEND message.

User Control (CZAMZ7) — Processes a call to clean up nonprivileged QLEs.

User Prompter (CZATJ2) — Find ABEND message in SYSMLF.

The second ABEND routine, CZACQ, also calls routines both within and without the command system. The routines within the command system are:

ERASE/DELETE (CZAEJ6) — Erases the SYSOUT data set.

EXECUTE (CZABB3) — Prints SYSOUT with an erase option for a nonconversational task.

GATE (CZATC) — Uses GATWR to send ABEND message.

CZACQ calls some modules outside the command system:

CAT FLUSH (CZCFX1) — Flushes the system catalog (SYSCAT).

CLOSE COMMON (CZCLB) — Closes SYSCAT, SYSLIB, Active User List.

TSEND (CEAH19)	Ends the task's time slice until the task's pending I/O requests are removed.
VAM ABIR (CZCQQ1)	Releases interlocks set by VAM routines.

(CZACQ also uses WTL, ATCS, EBCDTIME, and DLTSI macros.) The third ABEND module, CZAAQR only makes calls within the command system -- to log on a new task:

Initial Attention Interrupt Processor (CZAHB1)	To begin logging on a new task.
LOGON (CZATD2)	To complete logging on a new task.

EXITS: ABEND routine I (CZACP) normally returns to the task monitor after processing a completion code 1 condition for conversational tasks or nonconversational tasks for which a TSKABEND data set has been supplied. The routine exits to LOGOFF after processing a completion code 1 condition for a nonconversational task for which no TSKABEND data set has been supplied or if the task being ended abnormally is an Express Batch task. If a minor system error occurs during CZACP processing, ABEND is reinvoked.

ABEND routine II (CZACQ) normally exits to DLTSI to delete the task in which a completion code 2, 3, or 4 error condition occurred. If a major system error occurs during CZACQ processing, CZACP exits to the system error recovery routine (SERR).

ABEND routine III (CZACR) exits to the task monitor, via the RETURN macro instruction, after creating a new task for a conversational task in which a completion code 2 condition occurred. If a VSEND cannot be sent to the old task or if the VSEND reply from the old task is not received, ABEND is reinvoked.

OPERATION: The ABEND procedure processes four types of error conditions, completion code 1 (referred to as the ABEND condition), completion code 2 (referred to as the ABORT condition), completion code 3, and completion code 4 (terminal held condition). Conditions specified by codes 1 and 2 will terminate the logical task sequence at the point the error was detected and return control to the user. In a completion code 1 (ABEND) condition, control is returned to the user with the virtual storage as it was when the error occurred. The user can then determine the cause of the error and direct the task termination. In a completion code 2 (ABORT) condition, control is returned to the user with a completely new virtual storage. In a completion code 3 or 4 condition, the user does not receive control again. His virtual storage is destroyed and he is disconnected from the system.

The ABEND procedure consists of three routines. Routine I performs the initial processing of completion code 2, 3, and 4 conditions and the complete processing of completion code 1 conditions. Completion code 2 conditions require further processing by routine II for nonconversational tasks, and by routine II and routine III for conversational tasks.

Routine I determines whether the task being abnormally terminated has a special purpose ABEND routine; if so, ABEND links to it via a call to the User Control routine; if not, ABEND processing continues. A CHANGE macro is issued to move the task to Schedule Table Level 21.

For completion code 1 conditions, routine I will set off any mask bits set on in the VPSW, terminate the task's I/O, release the system interlocks set by the task, issue an error message and, in the case of a conversational task, return control to the user. For nonconversational tasks, but not for an Express Batch task, routine I will look for a new

SYSIN data set with a ddname TSKABEND. If the user has indicated a data set TSKABEND, this data set replaces the SYSIN data set and becomes the new SYSIN data set.

The task is then reinitialized. A WTO is issued to document the ABEND. If the user has not specified a TSKABEND data set, LOGOFF is called to terminate the task. If the task is an Express Batch task, a GET is issued to flush the SYSIN data set to the next LOGON card. If an error occurs during routine I processing of a completion code 1 condition, the ABEND procedure will be reinvoked and will process the error as a completion code 2 condition.

For completion code 2 conditions, routine I will unmask any masks set in the VPSW, terminate the task's I/O, and release the system interlocks set by the task. When the completion code 2 occurs in a nonconversational task or in a task that is not logged on, ABEND writes the error messages on SYSOUT.

The data sets are closed, data sets to be deleted at LOGOFF are deleted, private devices are released, and the user table is updated. Routine I then calls routine II to continue processing the completion code 2 condition. The task's shared modules are unlinked from the shared data set table. In the case of a nonconversational task, a message is sent to inform the batch monitor of the task's impending deletion. Routine II terminates the nonconversational task, or any task that is not logged on, by removing any I/O requests made by the task and deleting the task's symbolic device list. A WTL is issued to document the ABEND. Routine II then deletes the task from the system.

For conversational tasks, routine II creates a new virtual storage and collects information needed to activate a new task and sends this information to the new task.

Routine III in the new task sets the field used by the ABEND procedure to indicate the task is being created by ABEND and logs on the new task, using the information sent from routine II. The new task then issues the error messages, and routine III returns to the task monitor, which returns control to the user.

For completion code 3 situations, the task is deleted in the same way as for completion code 2 situations, except that routine III is never invoked to initiate a new task.

CZACP2 issues a GATWR for the module locations and registers associated with the last occurrence of ABEND, if any.

ERROR CONDITIONS: The routine may issue these system errors:

SYSER Code	Significance
050506701	MO/BM ABEND in pre-logon state.
050506702	MO/BM ABEND in post-logoff state.
050506703	ABEND unable to CHANGE task to Level 21.
050506704	STOW of SYSCAT member failed. Member interlock might be left on.
050506705	QLE failed before comp 1 exits.
050506706	TSICIO count not zero after 96 TSENDs.
050506707	ABEND in new task before SETAE completed.
050506708	DCB header not released after CLOSE.
050506709	ABEND called recursively in LOADER LOGOFF.
050506710	Used for system debugging stops.
050506711	VSS disconnect failed.
050506731	VSEND to BM for nonconversational tasks failed.
050506732	Close of SYSCAT data set failed.
050506733	Close of SYSLIB (SYSMLF) data set failed.

050506734 VAM interlocks release (CZCQQ2) failed.
050506736 PURGE I/O failed.
050506737 VSEND of 1 of 1 MCB to new task failed.
050506738 VSEND of 2 of 2 MCB to new task failed.
050506739 Reply not received from the new task in 255 TSEND.
050506740 ABEND during scanning of SDAT.
050506741 SDAT entry lock not reset after 255 TSENDs.
050506742 SCRTSI failed.
050506743 VSEND of 1 of 2 MCB to new task failed.
050506744 SETAE to shift SYSIN device for new task failed.
050506745 Other error conditions during creation of new task.
050506746 VSEND of reply to the new task failed.
050506747 Close of SYSSVCT (CZCFX5) failed.
050506748 ABEND unable to RELEAS a private data set.
050506749 CLOSE of USERLIB (SYSMLF) failed.

Note: All SYSERs have minor severity.

SYSTEM CONTROL BLOCK USAGE:

ABEND Interrupt Release Table (CHAAIR)
Active User Limits Table (CHAAUL)
Data Control Block (CHADCB)
Data Control Block Header (CHADHD)
Data Event Block (CHADEB)
Data Event Control Block (CHADEC)
Interrupt Storage Area (CHAISA)
Interrupt Table (CHAITB)
Message Control Block (CHAMCB)
Message Event Control Block (CHAMEB)
New Task Common (CHANTC)
Relative External Storage Correspondence Table (CHARHD)
Shared Data Set Member Entry (CHASDM)
Shared Data Set Table (CHASDS)
Symbolic Device Allocation Table (CHASDA)
System Activity and Resources Table (CHASAR)
System Common (CHASCM)
Task Accounting and Statistical Data (CHAACT)
Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Task Dictionary Table (CHATDY)
Task Status Index (CHATSI)
Terminal Control Table (CHATCT)
User Table (CHAUSE)

BACK Command Routine (CZABC)

This routine switches a conversational task to the nonconversational mode. BACK operates in conversational mode only. (See Chart BB.)

ENTRY: CZABC1 - normal entry

MODULES CALLED:

User Prompter (CZATJ1)	Issues error messages to the user.
FINDDS (CZAEC1)	Determines that a SYSIN data set exists and that JFCB for it has been built.
GATEOPEN (CZBTBA)	Opens nonconversational SYSOUT.

BACK also executes the following macro instructions:

RCR	1. Updates the User Table entry. 2. Gets ration. 3. Vacates task if no task space exists.
VSENDR	Sends requests for nonconversational tasks to the batch monitor (CZABA).
USELOCK	Unlocks User Table entry.
ESETL	Unlocks SYSUSE data set.
XTRACT	Accesses the TSI field containing the conversational/nonconversational indicator.
SETUP	1. Sets nonconversational indicator in TSI. 2. Puts the symbolic device address of nonconversational SYSIN in TSI. 3. Puts the batch sequence number in the TSI.
OPEN	Opens nonconversational SYSIN.
SCHED	Updates the schedule table index in the TSI to third level task.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: This routine first validates the environment in which the BACK command was issued -- VSS cannot be connected and the task must be conversational. The user must also have entered a valid dsname with the BACK command. If all is in order, the BACK processor calls FINDDS to determine whether a data set for the given dsname exists. Also, if the data set has not been defined during the current task, FINDDS creates a job file control block (JFCB) for it, provided the SYSIN data set name has been cataloged. A number of invalid conditions can result from the FINDDS operation -- see Error Conditions for elaboration.

On return, with RC=0, the BACK processor sends the request (via VSENDR) to the batch monitor to determine whether the system can accept a new background task at this time. The batch monitor will return a batch sequence number if a new task is acceptable, and BACK will then issue that number to the user, confirming the acceptance.

The remaining operations switch the task's mode. The conversational terminal is disconnected via the ATCS macro (SVC 219). The user's entry in the User Table is then updated to show that he no longer has an active conversational task, and the mode indicator in the Task Status Index (TSI) is reset accordingly. The new SYSIN and SYSOUT are set up for use, and opened. Finally, the routine issues a SCHED SVC, changing the task's schedule table index to reflect its new mode. Execution of the nonconversational task will then occur at the system's convenience.

ERROR CONDITIONS: The BACK Processor issues a diagnostic message to the user and returns to its calling routine if it determines that:

1. VSS is connected to the requesting task.
2. The requesting task is not conversational.
3. The user defaulted the data set name in the BACK command.
4. The data set name supplied has invalid length.

5. FINDDS indicates (RC=8) that the dsname is for some reason invalid -- for instance, it could not find the dsname in the catalog.

6. FINDDS indicates (RC=16) that the data set name cannot be retrieved.

7. The batch monitor indicates that no task space exists, or that shutdown is in progress.

Certain conditions cause the BACK processor to issue the following system error:

SYSER Code	Severity	ABEND Message	Significance
050202700	Minor	INVALID RETURN CODE FROM A CALLED ROUTINE	BACK received an invalid return code from a called module.

Conditions which can cause this error are:

1. Invalid return code from either FINDDS or the batch monitor.

2. FINDDS indicates (RC=4) that no JFCB was found in TDT and that there was no request to create one; or indicates (RC=10) that no JFCB in TDT and unable to build one -- for instance, because space unavailable.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Data Control Block (CHADCB)
User Table (CHAUSE)
New Task Common (CHANTC)
Task Status Index (CHATSI)
Task Data Definition Table (CHATDT)

Batch Work Queue (BWQ) EXHIBIT Processor (CZAYF)

The EXHIBIT Director (CZAYD) calls CZAYF to process requests for an informational display of the Batch Work Queue, when the BWQ option is specified in the EXHIBIT command. Various types of displays may be requested, which may result in a display of all entries or of a smaller subset of entries. (For example: only print requests may be called for.) (See Chart BC.)

ENTRIES:

CZAYF1 - normal entry
CZAYF2 - entry from ABEND processing

MODULES CALLED: None.

EXITS: CZAYF returns to its calling routine.

OPERATION: All information required for this display operation is available in the individual BWQ entries. If the task requesting a display of the batch work queue is not the operator task, only the requesting task's DCB for the Batch Work Queue is opened. The operator tasks is presented with the status of all jobs on the Batch Work Queue. CZAYF initializes the Batch Work Queue and determines whether the type specified is BSN.

If not, CZAYF obtains each BWQ entry in turn, releases the lock bytes, determines whether it is the type requested, and, if so, moves the entry into the work area provided by CZAYD. On end of data, if no entries have been found, a message to that effect is moved into the work area.

If the type specified is BSN, CZAYF reads on the BSN key, moving either the synad message or a 'no entry found' message into the work area.

In either case, after the entries have been examined, CZAYF TCLOSEs the BWQ for the operator task, and begins to write headers on the SYSOUT device. All write operations are directed to GATE. When the last header has been written, CZAYF converts and formats its BWQ data, line by line, and writes each line to SYSOUT via GATWR. When these entries are exhausted, CZAYF returns to its calling routine.

ERROR CONDITIONS: If, for a non-operator task, the BWQ could not be accessed, a diagnostic message is directed to GATE.

SYSTEM CONTROL BLOCK USAGE: Batch Work Queue.

CANCEL Command Routine (CZABJ)

This routine sends a message to the batch monitor to cancel a specified nonconversational task. CANCEL operates in conversational mode only. (See Chart BD.)

ENTRY: CZABJ1 - normal entry

MODULES CALLED:

NEXTPAR (CZAAC1) Gets and checks batch sequence number.

NUMSTG (CZAAC4) Validates batch sequence number.

MSGWR (CZAAD2) Issues messages to user.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The routine gets and checks the batch sequence number, which is provided as the command's operand, using the SCAN routines NEXTPAR and NUMSTG. CANCEL then builds a request message, including the user's identification and privilege class as well as the batch sequence number, and sends it to the batch monitor, via CALL if executed in the operator task or VSEND if requested by a user.

When the batch monitor receives the message, it will cancel the specified task if it is currently being executed or is awaiting execution, and if the user is privileged to cancel that task. Until the batch monitor responds, an AWAIT puts the CANCEL task in wait status. Upon receiving the batch monitor's response, CANCEL issues an appropriate message to the user, and returns to the calling routine.

ERROR CONDITIONS: The CANCEL processor checks the validity of the parameter format and the BSN entered with the CANCEL command. If either is in error the user is prompted and allowed to try again. An attention will interrupt this prompting.

After the call to the batch monitor, if the CANCEL routine discovers that the batch sequence number is either completed or nonexistent, or that the user is not privileged to cancel the task, it issues an error message before exiting. For the message IDs and exact text of the messages issued, see System Messages.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050203401	Minor	INVALID RETURN CODE FROM A CALLED ROUTINE.	Invalid return code received from called module.
050203402	Minor	MESSAGE TO BATCH MONITOR COULD NOT BE COMPLETED.	VSEND to batch monitor was not completed.
050203403	Minor	INVALID RETURN CODE FROM A CALLED ROUTINE.	Invalid response received from batch monitor.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Message Control Block (CHAMCB)
Message Event Control Block (CHAMEB)

CATALOG Command Routine (CZAEI)

This routine makes a permanent record of a non-VAM data set in a user's catalog, updates an existing non-VAM entry in the catalog, creates an index level in the catalog for a generation data group, or catalogs a data set as a new generation of a generation data group. For VAM data sets, it is used only to change the data set name in the catalog and on direct access volumes. (See Chart BE.)

ENTRIES:

CZAEI1 - command entry
CZAEI2 - macro instruction entry
CZAEI3 - special macro instruction entry for batch monitor
CZAEI11 - BPKD

MODULES CALLED:

SCAN Routines:

NEXTPAR (CZAAC1)	Locates delimiters of input string and scan for invalid characters.
CHEKDS (CZAAC2)	Validates data set name.
CHKNUM (CZAAC5)	Validates generation number.
MSGWR (CZAAD2)	Issues system messages.
FINDDS (CZAEC1)	Locates or creates JFCB for specified data sets.
DELCAT (CZCFD)	Changes data set name in catalog entry, or deletes the old name from the catalog.
RENAME (CZCFZ)	Changes data set name on direct access volume.
ADDCAT (CZCFA)	Creates or updates fields within the catalog.
LOCATE (CZCFL)	Creates index level in catalog for generation data group.
GETMAIN (CZCGA2)	Obtains additional virtual storage.

FREEMAIN (CZCGA3)	Releases virtual storage.
INDEX (CZCFI1)	Creates generation data group.
MTREQ (CZCAA1)	Mounts data set volume.
SETXP (CEAH7)	Makes a DSCB page available for processing.
PGOUT (CZAA1)	Writes out DSCB.
ABEND (CZACP1)	Aborts the task for abnormal conditions.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The command operands are validated using the SCAN routines.

If the new data set name has a relative generation number appended, indicating that it is a generation of a generation data group, LOCATE is called to obtain the absolute form of the generation number. Then the job file control block (JFCB) for the current data set name operand is located, and checked to ensure that the state of the data set (cataloged or not cataloged) as specified by the operand agrees with the state shown in the JFCB.

A non-VAM data set name is cataloged or the catalog data set descriptor (DSD) for a non-VAM data set is updated by calling the ADDCAT routine. Upon successful return from ADDCAT, control is returned to the calling routine. Any attempt to catalog or update the DSD of an existing catalog entry for a VAM data set or to rename a VAM data set which has no catalog entry is rejected by the CATALOG Command Routine.

If new and current data set names are entered as operands for:

- Non-VAM data sets and the current name is not cataloged, RENAME is called to replace the current name with the new name in the DSCB before a catalog entry is created and control is returned to the calling routine.

- Non-VAM data sets and the current name is cataloged, DELCAT is called to replace the current name with the new name in the catalog DSD. If the current data set also resides on a direct access volume, RENAME is called to replace the current name with a new name in the DSCB. Then ADDCAT is called to update fields in the catalog and control is returned to the calling routine.

- VAM data sets and the current name is cataloged, DELCAT is called to replace the current name with the new name in the catalog DSD. Then SETXP is called to make a Format-E DSCB available for processing. After changing the DSNAME in the DSCB, PGOUT is called to write out the Format-E DSCB and control is returned to the calling routine.

- VAM data sets and the new name is a generation data set name, ADDCAT is called to catalog the new name. The current name is replaced with the new name on the DSCB and DELCAT is called to delete the current name from the catalog.

If a catalog index is requested for a generation data group, a parameter list is created for INDEX. Upon successful return from INDEX, control is returned to the calling routine.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code is returned in the fourth byte of general register 15:

Code	Significance
00	Cataloging done as requested.
04	Name cannot be changed since new dsname not unique.
08	Input string error, no cataloging done.
0C	No cataloging for other reason.
10	Data set name already in catalog.
14	No volume of data set mounted, cannot catalog.
20	VAM data set not GDG or RENAME option.
24	Open DCB.

The routine checks for acceptable operands and notifies the user, through user prompter, of any errors found. For the messages issued, see System Messages.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050503302	Minor	SYSTEM ERROR, INVALID RETURN FROM MSGWR.	Invalid return code from MSGWR.
050503304	Minor	SYSTEM ERROR, INVALID RETURN FROM NEXTPAR.	Invalid return code from NEXTPAR.
050503306	Minor	SYSTEM ERROR, INVALID RETURN FROM CHEKDS.	Invalid return code from CHEKDS.
050503308	Minor	SYSTEM ERROR, INVALID RETURN FROM CHKNUM.	Invalid return code from CHKNUM.
050503310	Minor	SYSTEM ERROR, INVALID RETURN FROM FINDDS.	Invalid return code from FINDDS.
050503314	Minor	SYSTEM ERROR, INVALID RETURN FROM DELCAT.	Invalid return code from DELCAT.
050503316	Minor	SYSTEM ERROR, INVALID RETURN FROM RENAME.	Invalid return code from RENAME.
050503318	Minor	SYSTEM ERROR, TOO MANY VOLUMES OR UNABLE TO GETMAIN.	More than 255 volumes or unable to issue GETMAIN.
050503320	Minor	SYSTEM ERROR, INVALID RETURN FROM ADDCAT.	Invalid return code from ADDCAT.
050503322	Minor	SYSTEM ERROR, UNABLE TO COMPLETE COMMAND OR BACKOUT.	ADDCAT unsuccessful but indicator shows that successful change of data set name has occurred in catalog or DSCB through DELCAT or RENAME.

050503324	Minor	SYSTEM ERROR, INVALID RESULTS FROM INDEX.	Invalid return code from INDEX.
050503326	Minor	SYSTEM ERROR, INVALID RETURN FROM LOCATE.	Invalid return code from LOCATE.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Catalog SBLOCK (CHACCC)
Interrupt Storage Area (CHAISA)
Task Data Definition Table (CHATDT)
Symbolic Device Allocation Table (CHASDA)

CDD Command Routine (CZAFS)

This routine retrieves one or more DDEF commands for execution, obtaining them from a line data set that contains prestored DDEF commands. (See Chart BF.)

ENTRIES:

CZAFS1 - normal entry
CZAFS2 - macro instruction entry

MODULES CALLED:

SCAN Routines:	
NEXTPAR (CZAAC1)	Gets command operand.
ALFNUM (CZAAC3)	Validates ddname.
CHEKDS (CZAAC2)	Validates data set name.
MSGWR (CZAAD2)	Issue messages to user and fetch his replies. Also, lists the ddnames processed at the end of operation.
FINDDS (CZAEC1)	Finds or creates JFCB for prestored data set.
Command Analyzer (CZASA1)	Executes DDEF command via OBEY string.
GETMAIN (CZCGA2)	Reserves virtual storage for stack of ddnames.
FREEMAIN (CZCGA3)	Releases virtual storage reserved for ddnames.

EXITS: The routine returns to its calling routine, via the RETURN macro instruction.

OPERATION: If the command operand field does not include ddnames (data definition names), the CD routine will fetch all DDEF commands in the prestored data set, and present them, one by one, to the DDEF command routine for execution. In doing this, the CDD routine first validates the name of the prestored data set (using SCAN routines for this), gets the job file control block for that data set (using the FINDDS routine), then opens the data set and reads it, record by record. Each record containing a proper command verb is passed to the Command Analyzer for processing as an OBEY string. First, however, the CDD routine adjusts the record to take into account the record length, line number, and continuation lines. The routine will continue setting up records and passing them to the Command Analyzer until the prestored data set is exhausted. At that point, control returns to the calling routine.

When one or more ddnames are supplied as operands, the routine validates and then stacks them in temporary storage. Next, it scans sequentially through the prestored data set, looking for a match on a ddname with any that had been stacked. As each match is found, the prestored DDEF command is readied and passed to the Command Analyzer for execution. After the last of the requested ddnames has been executed (or when end of data set is encountered), the routine returns control to its calling routine.

If CDD is entered via a command, the routine accepts attention interrupts by immediately closing the prestored data set, setting a return code and returning control to the calling routine. If all the ddnames have not been processed, CDD releases the reserved storage through FREEMAIN, before returning to its caller.

ERROR CONDITIONS: If entry was by a macro instruction, a hexadecimal code will be returned in register 15:

Code	Significance
00	No error detected
04	Invalid data set name
08	Invalid data definition name
0C	Data definition name not in prestored data set
10	Error return received from DDEF (via Command Analyzer)
14	Not a line data set
18	Attention interrupt handled

The routine checks for acceptable operands and notifies the user, through user prompter of any errors found. For the messages issued, see command description given in System Messages.

The routine may issue these system errors. In each case the severity is minor, and the ABEND message is SYSTEM ERROR, TASK TERMINATED.

SYSER Code	Significance
050506801	Invalid NEXTPAR return code
050506802	Invalid NEXTPAR delimiters
050506803	Invalid MSGWR return code
050506804	Invalid ALFNUM return code
050506805	Invalid FINDDS return code
050506806	Invalid CHEKDS return code

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Data Control Block (CHADCB)
Task Data Definition Table (CHATDT)
Interrupt Storage Area (CHAISA)

CDS Command Routine (CZAFV)

This routine makes a copy of a data set or member of a partitioned data set. It may also copy member(s) of a partitioned data set (with user data and aliases) into a second partitioned data set, replacing duplicate members, if so specified. The original data set or member(s) may be erased after duplication. If the original data set or member is a line data set, the routine also renumbers the new data set when requested. (See Chart BG.)

ENTRIES:

CZAFV1 - command entry
CZAFV2 - macro instruction entry
CZAFV3 - EODAD entry in reading input data set
CZAFV4 - SYNAD from input or output data set
CZAFV5 - EODAD entry from output data set

MODULES CALLED:

CHEKDS (CZAAC2P)	Validates data set name.
CHKNUM (CZAAC5)	Validates line number and DCB operands.
PRMPT (CZATJ1)	Issues system messages to the user.
FINDDS (CZAEC1)	Locates or creates a JFCB.
DDEF (CZAEA1)	Defines the data set if no JFCB exists.
FIND (CZCOJ1)	Finds a member of a partitioned data set.
STOW (CZCOK1)	Updates the POD of a partitioned data set.
ERASE (CZAEJ7)	Erases data sets or members.
GETMAIN (CZCGA2)	Obtains virtual storage work area.
FREEMAIN (CZCGA)	Frees virtual storage work area.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: Upon entry to the macro, the parameter string is obtained, flags are set, and an OBEY is done to the command entry.

If entered as the result of a command, after each data set name is validated, a job file control block (JFCB) is obtained or created for each (by FINDDS). If the output data set JFCB cannot be found or created (that is, it is not cataloged), CDS calls DDEF to define the data set, with the same organization as the input data set. The remaining operands are validated and the erase, renumbering, and replace flags are set accordingly.

If the input and output data sets are both VAM partitioned and no member name has been specified, multiple member processing (copying of members with user data and aliases, if they exist) is assumed.

Otherwise, the input DCB is opened and checked against the output JFCB. Both data sets must have the same organization (VAM or SAM). Any combination of VAM data sets may be copied. If a VSAM data set is being copied to VISAM, the key length, relative key position, and pad must be

specified for the output data set. In all other combinations, the output is given the same DCB parameters as the input. For VSAM format U records, a LRECL of one page is used. The output DCB is then opened.

For SAM data sets, READ and WRITE are used to obtain the input records and place them in the output data set. For VAM data sets, GET and PUT are used. If renumbering is specified, the input record is obtained and the new key is overlayed on the old key before the record is written.

Tests for attention interrupts are made after each WRITE or PUT. Processing ends when the input data set is exhausted or an attention interrupt is received.

If multiple member processing has been indicated, three DCBs are opened; one for input, one for VSAM output, and one for VISAM output.

If no member names have been specified for the input data set, then every member found in the input data set's POD will be copied. Otherwise, only the members specified will be copied.

A FIND for a member is done, which fills in the input DCB and obtains the user data for the member. The output POD is searched to see if a member with the same name exists. Then each alias in the input POD which is associated with the member is checked in the output POD. If a similar alias is found, it must be associated with the same member name in the output POD or processing of the member is ended.

If no invalid duplicate aliases are found, and the user has not specified that duplicate members are to be ignored, the input member is copied into the output data set using the appropriate output DCB.

When the copy is complete, the input member is erased if applicable and the output member is added to the output POD with its user data and aliases, using STOW.

Multiple member processing is complete when all specified members have been copied.

When processing is complete, the DCBs are closed and the input data set is erased, if specified (not applicable to multiple member processing). Control is then returned to the calling routine.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code will be returned in register 15:

Code	Significance
00	Normal processing completed.
04	Invalid operands.
08	Name of original data set not in TDT or catalog.
0C	New data set not cataloged or no ddname specified.
10	JFCB for original data set not consistent with JFCB for new data set.
14	Member name not given for partitioned data set.
18	User does not have write access to new data set.
1C	Original data set organization not VAM or SAM.

20 Data set not on direct access or tape.

24 Member name for new data set already in POD.

28 Data set copied; old data set not erased because user does not have proper access.

2C Data set copied; new data set not renumbered because it is not line data set.

30 Data set copied and renumbered; old data set not erased because user does not have proper access.

34 Data set copied and erased; new data set not renumbered because it it is not a line data set.

38 Data set copied; new data set not renumbered and old data set not erased.

3C SYNAD or EODAD exits have occurred.

The routine checks for acceptable operands and notifies the user, through user prompter, of any errors found. For messages issued, see System Messages.

The routine may issue these system errors. In each case the severity is minor and the ABEND message is TASK TERMINATED. DSCOPY INCOMPLETE.

SYSER Code	Explanation
050508101	An invalid field in a JFCB has been discovered.
050508102	An illogical return code from STOW or FIND was received.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)

CHGPASS Command (CZATI)

This routine allows users to change their passwords by a command. This command will be accepted at any time after LOGON and before LOGOFF. (See Chart BH.)

ENTRY: CZATIC - normal entry

MODULES CALLED:

GATE (CZATC1)	Notifies user (via GATWR macro) to enter his new password (if defaulted) and cutrent password. Response read through GATRD.
User Prompter (CZATJ1)	Notifies the user of error conditions.

EXITS: This routine returns to its calling routine.

OPERATION: Upon entry, CHGPASS prompts the user to enter his new password in an overtyped field (if it was not already entered as a parameter with the command). Once the new password has been checked for correct syntax, CHGPASS prompts the user for his current password (again in an overtyped field). If the current password matches the USEPAS field, the new password is stored in USEPAS.

If the task requesting the change of password is nonconversational, CHGPASS uses a GTWAR macro to obtain the new and current passwords. If the task is conversational, a GATWR/GATRD combination is used.

ERROR CONDITIONS: CHGPASS issues an error message via the User Prompter and returns to its calling routine if (1) the user enters an invalid current password, or (2) the user enters an invalid new password.

SYSTEM CONTROL BLOCK USAGE:

User Table (CHAUSE)
Task Common (CHATCM)
Terminal Control Table (CHATCT)

CLOSE Command (CZCHB)

The CLOSE command is specified to allow the user to close his data sets from the command level. This function is required when the normal path of processing has been interrupted either by the system or by the user himself, and precludes this closure at the program level. This command may be invoked to close a data set, a group of data sets, or all data sets. (See Chart BI.)

ENTRY: CZCHB1 - command entry point

MODULES CALLED:

CHKDS (CZAAC2P)	Validates data set name.
LOCATE (CZCFL1)	Converts relative generation member to absolute.
STOW (CZCOJ, CZCOH, CZCOI)	Deletes data sets, sets and resets interlocks on the RESTBL.
CLOSE (CZCLB)	Closes the DCB.
CLOSEVAM (CZCOB)	Eliminates all traces of the DCB adjuncts (header, buffers, etc.).
PRMPT (CZBTC)	Issues messages to the user.

EXITS: Normal exit is made via return macro with a return code of RC=00 in Register 15.

OPERATION: This routine is coded as a BUILTIN. The BPKD included in the PSECT provides for three parameters -- DSNAME, TYPE, and DDNAME.

If both DSNAME and DDNAME are specified, only the data set for which both names match is closed. If an inconsistency is found, the user is informed.

CHEKDS is called to validate the data set name. The return code from CHEKDS is used to determine if a relative generation number or member name is included; a member name is ignored. The relative generation member is converted to absolute via a call to LOCATE; LOCATE is not called if DSNAME specifies an IBM System/360 Operating System data set name. If the data definition name exceeds eight characters, or the return code from CHEKDS is unexpected, or LOCATE fails to convert a relative generation number to absolute, the command is cancelled, and the user is informed.

The TDT is located, and the TDT header (TDTJ10) is used to find the last JFCB in the chain. The search of JFCBs is determined from the DSNAME or DDNAME parameter. If DSNAME is defaulted, all JFCBs in the

chain are processed, excluding the JFCBs for system data sets and USERLIB. If DSNAME is specified, the data set name field in the JFCB (TDTDS1) is compared against the specified data set name. The length (maximum of 35) for the comparison is obtained from the byte preceding the data set name pointed to by the BPKD. If a match is found, the character following the field on which the comparison was made (length + 1) is examined for a period or a blank if less than 35 characters have been compared. If a period is found, the name is partially qualified, and the search resumes after the current data set is closed. If a blank is found, the name is fully qualified, and processing is complete when the data set is closed.

Special consideration must be made for IBM System/360 Operating System data set names, which may be 44 characters long. Only JFCBs with TDTDS1=C'*' are examined and the name compared with TDTDS2.

If DDNAME is specified, the data definition name field in the JFCB (TDTDDN) is compared against the specified data definition name. The number of characters to be compared is determined from the byte preceding the data definition name pointed to by the BPKD. If the name is less than eight characters, the search continues through the JFCBs after the current data set is closed.

If a search of the TDT fails to find the specified data set name or data definition name, the command is terminated, and the user is informed that the data set is not defined in the current task. If the data set is found to be a system data set, the command is terminated, and the user is informed that a system data set may not be closed.

After the JFCB is located, the open count (TDTODN) is examined; if the count is zero, the command is terminated and the user informed that the data set is not open.

If the TYPE=T option is specified for a VAM data set that is specified by name and found to be duplexed (TDTDUP=0), the command is cancelled; the user is informed that TYPE=T cannot be specified for a duplexed data set.

When the data set has been properly closed, the user is informed. The actual close processing of a data set is determined from the data set organization (TDTDSV).

VAM Organization: If the data set is duplexed (TDTDUP=0), under defaulted names or partially qualified name processing, and the TYPE=T option is specified, the JFCB is ignored and the user informed that a duplexed data set cannot be closed with TYPE=T option.

The RESTBL pointer is obtained from the JFCB (TDTDEB). For a shared data set, the RESTBL is locked via CZOOH prior to a search of the DCB Headers. The first DCB Header is located from the pointer in the RESTBL Header.

The DCB Headers for a task (TCMTID=DHDTSK) are processed one at a time; from the DCB Header, the DCB address (DHDDCB) is obtained. Further testing is required prior to the actual call to CLOSE (CZCLB).

1. Before processing an individual DCB Header, all DCB Headers for the task are counted. This count is used to update, if necessary, the DCB open count in the JFCB (TDTOPN).

2. For a non-TYPE=T CLOSE, the existence of a valid DCB must be determined before the call to CLOSE (CZCLB). This is done via a CKCLS macro instruction for the space assigned to the DCB. A code of 0 returned in register 0 indicates that the DCB no longer exists.

CLOSEVAM (CZCOB) is called at a new entry point (CZCOB2) to eliminate all trace of the DCB adjuncts (DCB Header, buffers, etc.).

When the DCB exists, a check is made for a valid DCBID and the OPEN flag set. If the DCBID is not valid or the DCB is not flagged as open, CZCOB2 is called to perform the necessary cleanup.

When the DCB cannot be closed for any of the three reasons above, the user is informed that the data set could not be closed because the DCB was missing or invalid.

CZCOB2 is called with a pointer in register 1 to a parameter list containing a pointer to the DCB Header to be processed.

A call to CZCOB2 doesn't preclude the possible normal closure of other DCBs opened for the data set; for example, the RESTBL must not be released prior to processing of all DCB Headers.

When the data set is duplexed and CZCOB2 must be called, it is called twice to process each DCB in the same manner. If the DCBs are the last to be closed for the data set in this manner, the duplex pointer and indicator in the JFCB (TDTDUP and TDTDCI) must be cleared.

3. For a TYPE=T CLOSE the same checks must be made for an existing DCB that is open and contains a valid DCBID. If these conditions are not found, the close is not attempted and the user is informed that the temporary close was not done because the DCB was missing or invalid.

4. If the JFCB is in the JOBLIB chain, the DCB address is used to search the LIBE MAINT DCB chain to determine if the DCB is the JOBLIB DCB. If it is, the header is bypassed and, upon completion of processing of the data set, the user is informed that the JOBLIB DCB was not closed.

5. If the data set is duplexed (DHDDUP=0), the address of the second DCB is obtained from the DCB Header pointed to from DHDDUP and both DCBs are passed to DUPCLOSE.

6. For a CLOSE TYPE=T, the address of the next DCB Header for the task must be retained prior to the actual call to CLOSE. This address is used to resume processing of the next DCB Header rather than the pointer in the RESTBL Header used for non-TYPE=T CLOSE which will change as DCBs are closed and DCB Headers are deleted.

7. When the data set is partitioned and a new member is checked out, a STOW (type N) must be performed. The name used has the form:

   MYDDDNNN

   where M is the character M to conform to naming conventions, YDDD is the current date and NNN are three numeric characters to provide uniqueness to the name.

   If STOW passes back a return code indicating a non-unique member name, the NNN portion of the name is incremented and STOW is recalled.

   After the STOW has been performed, the user is informed that a new member (MYDDDNNN) has been stowed for the data set.

For shared data sets, the lock on the RESTBL must be released by CZCOI prior to the call to CLOSE and as soon as the DCB address is obtained.

SAM Organization: When DDNAME=ddname is specified for concatenated data sets, all JFCBs with the same ddname are examined and a close performed on each for which the DCB open count (TDTOPN) is greater than zero.

The DEB address in the JFCB (TDTDEB) is used to locate the DEB and from that, the DCB (DEBDCB). This address is passed to CLOSE to perform the actual closure of the data set.

Prior to the call to CLOSE (CZCLB) a check must be made for an existing DCB that is open and contains a valid DCBTD (see "VAM Organization" above). If these conditions are not found, the close is not attempted. For a non-TYPE=T CLOSE, the user is informed that the close of the data set failed because the DCB was missing or invalid.

ERROR CONDITIONS: The following messages are issued:

CZCHB001 -- invalid
CZCHB002 -- data set not defined in this task
CZCHB003 -- data set not open
CZCHB004 -- system data set may not be closed
CZCHB005 -- TYPE=T option invalid for duplexed data set
CZCHB006 -- DSNAME and DDNAME inconsistent
CZCHB007 -- joblib DCB not closed
CZCHB008 -- duplexed data set not closed with Type=T option
CZCHB009 -- close of data set failed; DCB missing or invalid
CZCHB010 -- new member (MYDDDNNN) stowed for data set
CZCHB011 -- close Type=T not done; DCB missing or invalid
CZCHB012 -- closed
CZCHB013 -- CSECT, Privileged, Public, READONLY, Reentrant

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Data Event Control Block (CHADEB)
DCB Header (CHADED)
Interrupt Storage Area (CHAISA)
Member Header (CHAMHD)
RESTBL Header (CHARHD)
Task Common (CHATCM)
Task Definition Table (CHATDT)

DATA Command Routine (CZADF)

This routine creates either a VISAM line data set or a VSAM data set. During creation of a line data set, the routine can also be used to insert, delete, and replace lines in that data set. (See Chart BJ.)

ENTRIES:

CZADF1 - normal entry
CZADF2 - SYNAD entry
CZADF3 - EODAD entry

MODULES CALLED:

NEXTPAR (CZAAC1)	Locates delimiters of input string and scan for invalid characters.
CHEKDS (CZAAC2)	Validates data set name.
ALFNUM (CZAAC3)	Validates member name.
CHKNUM (CZAAC5)	Validates line number.

FINDDS (CZAEC1)	Locates job file control block (JFCB).
DDEF (CZAEA1)	Creates JFCB for new data set if needed.
FORTRAN Conversion (CZAWC1)	Converts a record of FORTRAN data set from cardboard to keyboard.
ERASE (CZAEJ1)	Erases VSAM data set in case of attention interrupt.
MSGWR (CZAAD2)	Issues system messages.
SYSIN (CZASC7)	Issues number sign and fetch input records.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: After validating its input parameters by means of the SCAN routines, DATA checks to see if a job file control block (JFCB) exists for the data set. If so, the JFCB must show either (1) that the data set is partitioned or (2) it has VSAM or VISAM organization. For a partitioned data set, a new member will be created. If there is no JFCB for the data set, the routine generates one by calling DDEF. In that case, the data set name and organization are as specified in the input parameter, and the data definition name is derived from a value maintained for this purpose in task common. If JFCB already exists, for the data set name given, the DSORG value in the JFCB is assumed regardless of the value in the DATA command.

The routine now opens the data set, with a provision for both reading and writing. For a partitioned data set member, a search is made through the partitioned organization directory (POD) to ensure that the member name is unique. All further processing depends on the type of data set to be created, VSAM or VISAM.

For a VSAM data set, the routine issues a number sign (#) to invite each new line, and then puts the line into the new data set -- or member -- in this format:

length	card/keyboard	user text
(4 bytes)	(1 byte)	(120 bytes)

The keyboard/card indicator shows the origin of the line as either a terminal keyboard (code is 1) or a card reader (code is 0). In a conversational task, SYSIN indicates the origin of each line upon return to DATA (SYSIN is used to fetch the lines). In a nonconversational task, where input lines are prestored in the SYSIN data set, the indicator setting is part of the input record itself. The exception to this is fixed-length lines. The origin of those lines is assumed to be the card reader, and the keyboard/card indicator is set accordingly. DATA continues creating the VSAM data set (or member), one line at a time, until it finds an input record whose only significant characters are %E or an underscore followed by a command. The %E (or an underscore followed by a command) marks the end of input.

The building of a VISAM data set is similar, but the line format is:

length	line number	card/keyboard	user text
(4 bytes)	(7 bytes)	(1 byte)	(120 bytes)

Line numbers are assigned in ascending sequence, in the order lines are received. The other major difference from VSAM creation is that each VISAM input record is checked for modification indicators. If the input record contains %D followed by a line number, the line bearing

that number is deleted from the data set being built. If the line number is preceded by a % only, the text following it is written as either a replacement or an insertion line, depending on the line number specified. When a %E (or an underscore followed by a command) record appears for VSAM or VISAM data sets, DATA closes the completed data set, without writing the %E (or an underscore followed by a command) in the data set. If a new member was created, a member descriptor for it is added to the POD before the data set is closed. Control then passes to the calling routine. If the attention key is pressed while the DATA command is in operation, the path taken depends upon how far processing has advanced. If the new data set has not yet been opened, DATA merely returns control, leaving the JFCB set up (if it was generated). Otherwise, DATA closes the opened data set (the member is stowed if it is a VPAM data set), and then returns control to the calling routine. The VISAM data set, once opened, thus exists and may be added to later, if desired, by means of the MODIFY command routine.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through user prompter, of any errors found. For the messages issued, see System Messages.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050504502	Minor	SYSTEM ERROR, INVALID RETURN CODE FROM CALLED ROUTINE.	Invalid return code from a called module.
050504504	Minor	SYSTEM ERROR, TROUBLE WITH MEMBER NAME.	CHEKDS indicates a member name but left parenthesis was not found or, when adding member name, new data set name length is greater than 45.
050504505	Minor	SYSTEM ERROR. TROUBLE WITH CHKNUM ROUTINE.	Invalid return code. from CHKNUM, or all zeros in line number.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)

DDEF Command Routine (CZAEA)

To define a data set to the system, this routine creates a job file control block (JFCB) entry in the task data definition table (TDT). In addition, the routine issues requests for device allocation and direct access external storage, as needed by the newly defined data set. (See Chart BK.)

ENTRIES:

CZAEA1 - user command entry
CZAEA2 - FINDJFCB entry
CZAEA3 - user macro instruction entry
CZAEA4 - system macro instruction entry
CZAEA5 - fully qualified dsname entry
CZAEA6 - FINDDS entry

MODULES CALLED:

CHEKDS (CZAAC2)	Validates the syntax of data set name.
LOCATE (CGCFL)	Finds the catalog entry for the data set.
MTREQ (CZCAA1)	Mounts a volume for a non-VAM data set.
MTVOL (CZCAM1)	Mounts volumes for VAM data set.
RELEAS (CZCAD1)	Releases devices allocated for private volumes.
ALLOCATE (CZCEA)	Allocates pages for new data sets on direct access.
GETMAIN (CZCGA2)	Gets a new page for JFCBs.
MSGWR (CZAAD2)	Issues system messages to SYSOUT.
LIBMAINT (CZCDH)	Opens the DCB for a data set defined as a JOBLIB.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: Operation of this routine may be divided into four broad steps:

1. The first step is to read the command (or macro instruction) operands, and set these operand values in the new JFCB.

2. The second step is to set default values -- where defaults are indicated -- in the new JFCB. If the data set being defined is cataloged, values available in the catalog entry for the data set are also moved to the JFCB. Information available is in the catalog will override system default information.

3. The third step is to effect volume mounting and space allocation for data sets, if necessary.

4. Tne last step before returning control is to link the newly created JFCB into the proper chain or chains in the TDT.

Operand Handling: The routine first sets up the new JFCB. It locates the JFCB by checking the TDT header. In most cases, the JFCB fields are blanked or zeroed.

Operands in the input string are now inspected one by one. The first operand is the ddname (data definition name), which if syntactically correct is moved to the JFCB. At the same time a check is made through the TDT for a matching ddname. If one is found, the address of the matching ddname is saved for later tests and for possible use in concatenating. The second operand is assumed to be the dsorg (data set organization) value. It too is validated before being moved to the JFCB.

All subsequent operands are identified by keywords. As each keyword is recognized, the individual elements of that operand are isolated, checked for syntactic correctness, and then processed by an appropriate part of the DDEF routine. In general, handling of UNIT, VOLUME, SPACE, DISP, LABEL, OPTION, and DCB operands involves validating elements for syntax and logic, and then setting them in the corresponding JFCB fields. For the DSNAME, the routine ensures that a fully-qualified data set name is placed in the JFCB. It will allow for partitioned data set members, members of generation data groups, IBM System/360 Operating

System names, normal data set names, and names containing special ASCII characters.

If the first two characters of the data set name parameter are a quote and a blank, in that order, the normal TSS syntax is ignored and the data set name is scanned until the first quote-comma pair is found. All other characters between these pairs are assumed to be the data set name, and it is not validated, except for length. (If the new data set name matches a name already in the TDT, and the old ddname is not a system name, the new ddname will be substituted for the old ddname and the routine will return control to its calling routine without processing any other operands.)

Processing of keyword operands continues until the end of the input string is reached. UNIT, LABEL, and VOLUME are ignored for a cataloged data set; SPACE is ignored if the data set already exists.

If the first two characters in the volume serial number specified in the volume parameter are a quote and a comma, the next six characters are assumed, without checking, to be the volume serial number. They are not validated. One seventh character is checked for a comma or a right parenthesis. If neither, DDEF assumes a user error has occurred.

Default Handling: The routine now checks to see if certain values, such as dsorg, have been defaulted or if other values, such as file sequence number, have been defaulted under certain conditions. As needed, the routine will set standard values in the JFCB to fill out the defaulted fields. Most defaults result in the operand or element being ignored, unless the information is available in the catalog. Only dsorg, disp, file sequence number, and volume sequence number are always considered. Other defaults that depend on the device required by the data set (for example, tape density) will be processed during device or storage allocation.

The routine also checks to see if a data set nas been cataloged. If so, the routine tests to ensure that the catalog and JFCB values for dsorg and disp agree. If the data set is cataloged, and disp is defaulted, disp is set to old. In addition, it uses the catalog information to fill certain fields of the JFCB: label-type, original data set, and volume flags. Note that catalog information takes precedence over corresponding information supplied by the DDEF command or macro instruction.

Volume Handling: For a data set on a private non-VAM volume, the routine makes a request to MTREQ for device allocation and sets appropriate flags in the JFCB.

For a new non VAM data set on direct access storage, the routine moves the data set space requirements to the JFCB, then calls ALLOCATE to reserve space for the data set on a direct access volume. The space requirements may be either default values or values specified in the DDEF command or macro instruction.

For data sets on VAM volumes, the routine calls MTVOL to mount all volumes.

Chaining: The routine places the new JFCB into the main JFCB chain of the TDT. Depending on the data set defined, it may also link the JFCB to the temporary tabulation chain (for a new data set) or to the library chain (for a JOBLIB). When a concatenation is indicated by the CONC option and by matching ddnames for physical sequential organized data sets, except ASCII formatted data sets, forward and backward concatenation pointers are set as well.

Before returning control to the calling routine, the DDEF routine tests to see if the newly created JFCB has used the last available space on the current page. If this is so, GETMAIN is invoked to fetch a new page, which is then formed into 17 JFCBs, all linked together, for future use.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code is returned in the fourth byte of general register 15. Also, the message identification code of the diagnostic which would have been issued if the DDEF had been a command is placed into general register 1, to clarify the meaning of a nonzero return code.

Code	Significance
00	No error detected
04	Data set undefined
08	Data set name not unique
0C	Attention interrupt occurred
10	Inconsistent dsorg
14	Nonexistent generation name specified
18	DSNAME not fully qualified
1C	VOL(s) cannot be mounted
20	Space not available
40	Duplicate ddnames but no concatenation
80	Other invalid specification

The routine checks for acceptable operands and notifies the user, through user prompter, of any errors found. For the messages issued, see System Messages.

The routine may issue these system errors. In each case the severity is minor, and the ABEND message is SYSTEM ERROR - TASK TERMINATED.

SYSER Code	Explanation
050502500	Invalid conversational (TCMCOV) or confirmation (TCMCDF) field in task common.
050502501	Invalid time in system common.
050502502	Invalid data set name from FINDDS or error in DDEF scan routine.
050502503	Invalid return code from CHEKDS when validating data set name.
050502504	Invalid return code from LOCATE (when searching catalog for duplicate data set name).
050502505	Invalid fully qualified name from LOCATE, or SBLOCK is not a data set descriptor.
050502506	Invalid return code from FINDDS.
050502507	Logic error within DDEF in DCB parameters.

050502508 Invalid return code from LOCATE for SBLOCK of next volume.

050502509 Invalid return code from ALLOCATE.

050502510 Invalid return code from MTREQ.

050502511 Return code from MTREQ indicates requested volume could not be found.

050502512 Invalid return code from RELEAS.

050502513 Inconsistency of address with message response area; end-of-field address is not greater than next available address for response.

050502514 Invalid return code from MSGWR.

050502515 Invalid return code from GATRD (used to read parameter list from SYSIN).

050502516 Invalid delimiter from scan routine in DDEF.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Symbolic Device Allocation Table (CHASDA)
System Common (CHASCM)
Interrupt Storage Area (CHAISA)
Catalog SBLOCK (CHACCC)
Task Data Definition Table (CHATDT)
Public Volume Table (CHAPVT)

Routine Control Blocks: The DDEF routine employs several internal tables, detailed descriptions of which may be found in the listing. Below is a summary of each table's function.

1. The parameter address list and the parameter descriptor table control transfers within the routine while operands are being processed. The parameter address list consists of 18 words, and the parameter descriptor table consists of 37 words.

2. The parameter definition table is used for prompting.

3. Two tables are used to validate the individual elements of DDEF. The first table is a list of attribute and flag addresses for each element. The second table is the attribute table which defines what is valid for each element.

4. The consistency table is used to ascertain whether the parameters are consistent with each other. This table is only used for uncataloged data sets, and is only part of the internal consistency checking done by DDEF.

5. DDEF uses a table of logical diagnostics with a description of permitted replies. It consists of two words for each logical diagnostic issued.

DSS?/PC? Command Routine (CZAEL)

This routine processes both DSS? and PC? commands. For DSS?, the status of one or more cataloged data sets is presented. For PC?, the data set name, user's access, and, for shared data sets, the owner's identification of one or more cataloged data sets is presented. (See Chart BL.)

ENTRIES:

CZAEL1 - DSS? entry
CZAEL2 - PC? entry

MODULES CALLED:

GATE (CZATC1)	Writes information on SYSOUT.
MSGWR (CZAAD2)	Issues system messages.
MTREQ (CZCAA1&2)	Mounts private volumes; obtains the symbolic device address of a public volume.
LOCATE (CZCFL)	Retrieves a catalog SBLOCK for a given fully-qualified name.
OBTAIN (CZCFO1)	Obtains a DSCB for a SAM data set on a direct access device.
SCAN Routines:	
NEXTPAR (CZAAC1)	Obtains input parameters.
CHECKDS (CZAAC2)	Checks for a valid dsname.
ALFNUM (CZAAC3)	Checks for a valid userid.
FREEMAIN (CZCGA3)	Releases the pages obtained by LOCFQN for the TBLOCK structure.
ABEND (CZACP1)	Terminates processing after a system error is detected.
BUMP (CZCAB1)	Mounts successive private volumes.
SETXP (CEAH7)	Issues an SVC 244 to read in a DSCB page.
RELEASE (CZCAD1)	Releases private volumes mounted.

EXITS: The routine returns to the calling routine via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The SCAN routines are called to obtain input parameters and validate the dsname and userid. If the parameters are valid, LOCATE is called with the input name prefixed to the userid to obtain the TBLOCK structure. LOCFQN returns with TBLOCKs containing sharing and volume information. Volume information is extracted from the TBLOCKs and printed. For DSS?, MTREQ is called to mount the required volumes. For direct access devices (either public or private), OBTAIN is called to obtain a DSCB for a SAM data set or a SETXP is issued to read in a DSCB page for a VAM data set. Additional information is extracted from the DSCB and printed.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user of any errors found. For messages issued, see System Messages. The routine may issue this ABEND:

'TASK ERROR. CZAEL PROCESSING COMPLETE, BUT TRANSIENT DEVICE NOT RELEASED. DEVICE FREED BY ABEND.'

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
TBLOCK Data Set Descriptor (CHATBD)
TBLOCK Continuation (CHATBC)
TBLOCK Sharer (CHATBS)
Interrupt Storage Area (CHAISA)
TASK Definition Table (CHATDT)
Format-E DSCB (CHADSE)
Public Volume Table (CHAPVT)

ERASE/DELETE Command Routine (CZAEJ)

This routine erases or deletes a data set, as specified by the user. The DELETE procedure removes a data set entry from the catalog. The ERASE procedure erases the direct access storage assigned to the data set as well as deleting the catalog entry. (See Chart BM.)

ENTRIES:

CZAEJ1 - ERASE command entry
CZAEJ3 - ERASE entry from batch monitor
CZAEJ4 - DELETE command entry
CZAEJ5 - DELETE macro instruction entry
CZAEJ6 - ERASE entry for COPY, ABEND, and LOGOFF
CZAEJ7 - ERASE macro instruction entry
CZAEJ8 - ERASE entry for QUIT
CZANER - SYNAD error entry
CZAEOD - EODAD entry

MODULES CALLED:

SRCHSDST (CZCQE1)	Searches for data sets opened by sharers.
NEXTPAR (CZAAC1)	Gets a command operand.
CHEKDS (CZAAC2)	Validates a data set name.
MSGWR (CZAAD2)	Issues system messages.
FINDDS (CZAEC1)	Locates the address of a JFCB.
RELEASE (CZAFJ6)	Closes the data control block.
DDEF (CZAEA5)	Creates a JFCB.
LOCATE (CZCFL)	Finds a data set descriptor for shared data set.
DELCAT (CZCFD)	Removes a data set from the catalog.
SCRATCH (CZCES)	Removes a data set control block from volume table of contents.
FREEMAIN (CZCGA3)	Releases virtual storage page.
GDV (CZASDX)	Obtains a default value for DEPROMPT.
DELVAM	Removes VAM data sets from the catalog and the volumes in ERASE mode.

This routine also uses OPEN, FIND, STOW, CLOSE, and MTREQ.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: Before processing either an ERASE or DELETE request, the SCAN routine NEXTPAR and CHEKDS are used to fetch and validate the command operand. If the operand proves valid, LOCATE is called to search the catalog, and compile a list (TBLOCK) of all data sets qualified by the input data set name. If the task is nonconversational, the data sets are erased or deleted as originally specified by the user.

For any data set to be processed, FINDDS is called to search the task data definition table (TDT) for the address of its job file control block (JFCB). If the data set is cataloged but not found in the TDT, FINDDS calls DDEF to create a JFCB.

It is necessary to check for open data sets before erasing. If a data set has been opened by sharers, or by the owner in another task, the ERASE command will be ignored. The erasure or renaming of an open data set would have resulted in subsequent errors in the system should one of the sharers attempt to use the data set.

Before processing an erase or delete request, GDV is called to determine the value of the DEPROMPT operand in the user profile. If the value is Y, ERASE/DELETE prompts the user for disposition of data sets, when more than one is referenced.

For non-VAM data sets, the ERASE/DELETE routine processes an ERASE request by performing the following functions:

1. RELEASE is called to close the data control block.
2. SCRATCH is called to delete the data set control blocks from the volume table of contents (VTOC).
3. If the data set is cataloged, DELCAT is called to delete the various index levels from the catalog structure.
4. RELEASE is called to release the data set entry in the JFCB.

For VAM data sets, the ERASE/DELETE routine processes an ERASE request by calling DELVAM to delete the catalog entry, remove the data set from the volume and release the JFCB. If only a member of the data set is to be released, ERASE/DELETE calls FIND to locate the member name in the Partitioned Organization Directory (POD) and STOW to remove the member from the POD.

In processing a DELETE request, the ERASE/DELETE routine calls DELCAT to delete the data set's catalog entry. If there is a JFCB, RELEASE is called to close the DCB and release the JFCB.

Final processing of the ERASE/DELETE routine is as follows: FREEMAIN releases the virtual storage previously obtained by LOCFQN. Control is then returned to the command analyzer and executor, or the calling routine.

When the batch monitor calls ERASE/DELETE, only the ERASE procedure is used. This operates basically as described above except that the routine returns error codes rather than printing diagnostic messages and waiting for a response, and DDEF will be called if there is no JFCB and LOCATE has been successful.

Code	Significance
4C	System data definition name found in TDT E
58	Data set opened by another sharer

The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For the messages issued, see System Messages.

The routine may issue these system errors. In each case the severity code is minor, and the ABEND message is SYSTEM ERROR - TASK TERMINATED.

SYSER Code	Explanation
050503401	Nonzero return code from CHEKDS.
050503404	Unsuccessful return code from MSGWR.
050503405	Data set count is nonzero, but no forward pointer in TBLOCK.
050503406	Invalid type of TBLOCK.
050503407	Return code of C from FINDDS. No space is allocated.
050503408	Invalid return code from FINDDS.
050503409	FINDDS not finding JFCB created by DDEF.
050503410	Invalid data set name code returned by DDEF.
050503412	Invalid return code from SCRATCH.
050503413	Invalid return code from LOCATE.
050503414	Unsuccessful return from DELCAT.
050503415	DELCAT unable to delete data set descriptor (DSD).
050503416	RELEASE unable to release JFCB from TDT.
050503418	CHEKDS found invalid member name.
050503419	Invalid return code from STOW.
050503420	Invalid return from NEXTPAR.
050503423	Unknown TBLOCK; not TBD or TBC.
050503424	Validity bit in JFCB not on.
050503426	Invalid return from RELEASE when trying to close DCBs.
050503427	Invalid return code from MSGWR.
050503428	Sharing indicator but no sharing descriptor TBLOCK.
050503429	Label error.
050503430	SYNAD exit.
050503431	EODAD exit.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code will be returned in the fourth byte of general register 15:

Code	Significance
00	No error detected E/D

04 Not class D or batch monitor entry E

08 Invalid return code from NEXTPAR E/D

0C Invalid delimiters in data set name E/D

10 No data set name supplied E/D

14 Invalid return code from CHEKDS E/D

18 Data set name not in catalog or TDT E or volumes could not be mounted

1C Partitioned data set not fully qualified name E

20 Member of partitioned data set not found in POD E

24 Data set not cataloged D

28 Data set on public volume D

2C Data set is member of partitioned data set D

30 User does not own data set in ERASE batch monitor entry E

34 Sharing/access conflicts prevent processing E/D

38 No catalog entry for ERASE batch monitor entry E

3C Data set name undefined-return code from DDEF E

44 Data set not on direct access E

48 Volume not found

4C System JFCB

58 Data set in use

5C Resources exceeded, volume cannot be mounted

60 Loaded DS; cannot erase or delete

64 Bulk I/O pending on data set

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
TBLOCK Data Set Descriptor (CHATBD)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)
Interrupt Storage Area (CHAISA)

EVV Command Routine -- CATVAM (CZCFB)

This routine catalogs data sets on private volumes which were produced on one system to allow their use on another system. In support of this function, the routine will optionally alter the userid portion of the FQN in the Format-E DSCB to make the FQN compatible to the second system. Multi-volume data sets are also processed by this routine. (See Chart BN.)

ENTRY: CZCFB1 - normal entry

MODULES CALLED:

DDEF (CZAEA1)	Sets up a JFCB with the necessary parameters.
MTREQ (CZCAA)	Issues mount requests for the required volumes.
VAMINIT (CZCEQ)	Reads in the PAT table and initializes the volume.
DSCB RD/WR (CZCEM)	Makes a DSCB page available to the task.
ADDCAT (CZCFA)	Creates a catalog entry.
PGOUT (CEAA1)	Writes out a DSCB page.
BUMP (CZCAB)	Dismounts and mounts volumes.
INDEX (CZCFI)	Updates and chains an indexed data set.
LOCATE (CZCFL)	Verifies the userid.
GETMAIN (CZCGA2)	Obtains virtual storage.
FREEMAIN (CZCGA3)	Releases virtual storage.
User Prompter (CZATJ1)	Issues diagnostics to user.
RELEASE (CZAFJ1)	Freeup device and JFCB.

EXITS: Control is returned to the calling routine via the RETURN macro instruction.

OPERATION: After validating the input parameter list, a random ddname is generated for the DDEF instruction. The DDEF macro is then issued with enough parameters specified to create a JFCB which meets the requirements for later processing by ADDCAT, MTREQ and BUMP.

A parameter list is then prepared for the MTREQ routine. The parameters consist of the volume serial number(s) and the address of the JFCB for the task. MTREQ issues a mount request for the first volume to be mounted and GETMAIN requests a block of virtual storage in order to read in and process DSCB pages. The VAMINIT routine is called to initialize the volume and read in the PAT.

The PAT is scanned to find a DSCB page and the DSCB page is read into virtual storage by invoking the CZCEM routine. If the option to alter the userid portion of the FQN was specified, the first Format-E DSCB is located and the userid portion of the FQN is altered and the DSCB checksum is modified. LOCATE is called first in order to verify the old userid and see if the data set is cataloged. If the data set is not cataloged but the userid is valid, ADDCAT is called to catalog the data set (the new userid, if specified, is updated before calling ADDCAT). The condition code returned from ADDCAT is checked to assure that the data set was cataloged successfully. If the data set is cataloged and the userid is valid, the DSD is verified to assure that the data set being processed is the same as the cataloged data set.

If the data sets are the same, the next Format-E DSCB is located and the same procedure is followed as for the previous Format-E DSCB. The next Format-E DSCB is located and the same procedure is followed as for the previous Format-E DSCB.

If the option to alter the userid portion of the FQN was specified, the DSCB WRITE routine (CZCEM1) is called to write the altered DSCB page on the same place on the disk from which the DSCB page was originally

read. (If the write is unsuccessful the user is informed of the failure to relocate the DSCB, and further processing is bypassed.) If the write is successful (or if the DSCB was not changed), the PAT is scanned for more DSCB pages. If any are found, they are read in and the same procedure is followed as for the previous DSCB page until the entire PAT has been scanned and the entire volume cataloged.

After volume processing is complete, a test is made for task completion. If it is not complete, the BUMP routine is called to dismount the current volume and mount the next volume in the list. VAMINIT is then called to initialize the next volume and processing continues as for the previous volume.

After the last volume of the task is processed, the direct access device is released by invoking the REL command. This frees up the device and releases the JFCB. A call to FREEMAIN releases the virtual storage page.

ERROR CONDITIONS: EVV prompts the user and returns to its calling routine under the following conditions:

1. Missing or invalid command parameter.
2. An attempt by other than a manager or administrator to change the userid.
3. A userid that has too many characters or is not joined to the system.
4. Erroneous volume entries.
5. An attempt to build a JFCB fails.
6. On the mount request (a) the user has cancelled, (b) the first volume cannot be mounted due to a MTREQ error, (c) the user is not authorized to use the volume.
7. The requested data set cannot be shared by the current user.
8. DSCBWR cannot relocate the DSCB.
9. ADDCAT cannot catalog the data set.
10. Device release fails after the task has been completed.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Format-E DSCB (CHADSE)
Catalog SBLOCK (CHACCC)

EXECUTE Command Routine (CZABB)

This routine sends a message to the batch monitor to initiate a non-conversational task. The name of the data set that will serve as the task's SYSIN is supplied in the message. EXECUTE operates in conversational mode only. (See Chart BO.)

ENTRY: CZABB1 - normal entry

MODULES CALLED:

NEXTPAR (CZAAC1) Fetches data set name and checks for valid characters.

CHEKDS (CZAAC2)	Validates syntax of data set name.
LOCATE (CZCFL)	Finds catalog entry for SYSIN data set name.
MSGWR (CZAAD2)	Issues messages to user (bridges into user prompter).
VSEND	Sends request to batch monitor (if the task is not the main operator's).
AWAIT	Delays processing until reply to request message is received.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The routine first fetches the name of the SYSIN data set, which is furnished as the input parameter, and checks that it is valid; two SCAN routines, NEXTPAR and CHEKDS, are called for this. EXECUTE then checks, via LOCATE, to see that the valid data set name actually appears in the catalog. If not, or if the name proves invalid, a diagnostic message is issued, asking the user to reenter the data set name.

After the name has been fully validated, EXECUTE builds a request message and sends it, via VSEND, to the batch monitor. When the batch monitor receives the message, it assigns the task a batch sequence number, and adds an entry for the task in the batch work queue. Until the batch monitor responds, an AWAIT puts the EXECUTE task in wait status. Upon return, the batch sequence number is issued to the user and EXECUTE returns control to the calling routine.

Before and after issuing each diagnostic message, and before sending the request message to the batch monitor, EXECUTE polls for attention interrupts. If one is detected, control is immediately returned to the calling routine.

ERROR CONDITIONS: The routine checks for an acceptable operand and notifies the user, through user prompter, of any errors found. For the messages issued, see System Messages.

SYSER Code	Severity	ABEND Message	Significance
050202601	Minor	INVALID RETURN CODE FROM A CALLED ROUTINE.	Invalid return code received from a called module.
050202602	Minor	MESSAGE TO BATCH MONITOR COULD NOT BE EXECUTED.	VSEND to batch monitor not completed.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Batch Work Queue (CHABWQ)
Catalog SBLOCK (CHACCC)
Message Control Block (CHAMCB)
System Message Record (CHAMSG)
Message Event Control Block (CHAMEB)

EXHIBIT Director (CZAYD)

The EXHIBIT Director is called when the EXHIBIT command has been entered from a terminal supported by TSS/360 as a SYSIN device. The Director then performs initialization, validates the request, and calls the applicable display routine to process the request. (See Chart BP.)

ENTRY: CZAYD3 - normal entry

MODULES CALLED:

UID EXHIBIT Processor (CZAYG)	To process requests for a display of all active tasks in the system.
BWQ EXHIBIT Processor (CZAYF)	To process requests for a display of the Batch Work Queue.
GETMAIN (CZCGA2)	To obtain a work area for the display processors.
FREEMAIN (CZCGA3)	To release the work area acquired through GETMAIN.

EXITS: When processing is complete CZAYD returns to its calling routine.

OPERATION: CZAYD's calling routine provides as input a pointer in Register 1 to a parameter list containing pointers to character strings. These are pointers to OPTION, TYPE, and FORM parameters. The option must be specified and valid; it may be either UID or BWQ, requesting displays of either the information about active tasks in the system or the activity on the Batch Work Queue.

CZAYD calls an internal subroutine, SCAN2, to validate the UID or BWQ type. If the UID type is valid, the UID form is also checked. It then sets a branch table index to either the UID or BWQ displacement and provides a 50-page work area for the subsequently called processor (CZAYG or CZAYF). On return, if the task requesting the EXHIBIT function is tne operator task, CZAYD releases the 50 pages allocated as a work area. CZAYD then returns to its calling routine.

ERROR CONDITIONS: All diagnostic messages are directed to GATE. The following conditions cause diagnostic messages:

1. No option specified.
2. EXHIBIT option invalid.
3. UID or BWQ type invalid.
4. UID form invalid.

SYSTEM CONTROL BLOCK USAGE: None.

FINDDS Routine (CZAEC)

Given a data set name, FINDDS searches the task data definition table (TDT) for a job file control block (JFCB), and either provides the user with its address, or creates one if requested by the user. (See Chart BQ.)

ENTRIES:

CZAEC1 - normal entry
CZAEC2 - used when the userid is specified in an input parameter list rather than in Task Common

MODULES CALLED:

DDEF (CZAEA6)	Creates new JFCB using information from the catalog.

LOCATE (CZCFL1) Converts relative generation number to absolute number.

CHKDS (CZAAC2) Validates dsname.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: FINDDS checks the data set name for a relative generation number or member name. A member name is removed, since it is meaningless to this routine. A relative generation number is converted to its absolute form (via LOCATE). If a relative generation data set name is not a FQN of a GDG, relative generation zero (0) is appended to the data set name.

The task data definition table (CHATDT) is then searched for a JFCB containing the given data set name and, if it is found, its address is returned to the user. If a JFCB is not found, and the user requested one, a data definition name is generated from a value maintained for that purpose in task common. However, if FINDDS is entered at CZAEC2, it does not fetch the userid from Task Common, but uses the userid passed to it in Word 4 of the input parameter list. DDEF is called with this ddname to create a JFCB in the TDT. Its address is then supplied to the user.

ERROR CONDITIONS: A hexadecimal code is returned in register 15:

Code	Significance
00	JFCB found or created as requested
04	No JFCB found; no request to create one
08	No JFCB found; request to create one, but DDEF could not find data set name in catalog
0C	No JFCB found; DDEF could not create one because space unavailable.
10	DSNAME invalid.
14	No JFCB found; DDEF could not create one because volume could not be mounted.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050502702	Minor	SYSTEM ERROR, FINDDS. NO RIGHT PAREN IN DSNAME.	No right parenthesis after relative generation number.
050502704	Minor	SYSEM ERROR, UNEXPECTED RETURN FROM LOCATE.	Return code from LOCATE is invalid or LOCATE did not replace relative generation number with absolute one.
050502706	Minor	SYSTEM ERROR, FINDDS. INVALID RETURN FROM DD.	Invalid return code from DDEF.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)

FINDJFCB Routine (CZAEB)

This routine locates the job file control block (JFCB) for a given data definition name (ddname) and, if requested, checks that the proper volumes are mounted. (See Chart BR.)

ENTRY: CZAEB1 - normal entry

MODULES CALLED:

MSGWR (CZAAD2) Issues messages to user.

DDEF (CZAEA2) Defines ddname.

MTREQ (CZCAA1) Requests volume mounting.

BUMP (CZCAB1) Replaces mounted volume with requested volume.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction, with a pointer to the JFCB set in a caller-designated area. If the JFCB cannot be found and a normal return is made, the caller-designated area will be set to zero. If a system error occurs, or if a JFCB cannot be found when it is mandatory, the routine exits to ABEND.

OPERATION: Three option codes (0, 1, and 2) control FINDJFCB processing:

0 = find JFCB, mount volumes, and do not return unless JFCB is found;

1 = find JFCB and mount volumes, if possible, but return in any case;

2 = find JFCB, if possible, and return, but do not mount volumes.

For an option code of 0 or 1, the task data definition table (TDT) is searched for a JFCB with the specified ddname. If the JFCB is not found, the conversational user is asked if he wants to define the ddname. If he indicates yes, DDEF is called to built the JFCB. If he indicates no, or if the task is nonconversational, the action taken depends on the option code. For option 0, a diagnostic is issued and the task is terminated via ABEND. For option 1, the output area is set to zero and return is made to the calling routine.

When the JFCB is found or created, FINDJFCB next checks that the proper volumes are mounted. For a SAM data set, only the first volume of the data set has to be mounted. For VAM data sets, the volume(s) are mounted at OPEN time. If the proper volumes cannot be mounted immediately, MTREQ is called to reserve the necessary devices and place the task in the WAIT state until the devices become available. MTREQ then mounts the necessary volumes. When volume mounting is completed, FINDJFCB sets a pointer to the JFCB in the output area and returns to the calling routine.

For option code 2, there is no attempt to mount volumes; only a search of the TDT is made to find the JFCB. If found, a pointer to the JFCB is placed in the output area and return is made to the calling routine. Otherwise, the output area is set to zero before returning to the calling routine. Option code 2 does not prompt the user, nor does it check that the proper volumes are mounted.

ERROR CONDITIONS: The routine takes this action if the JFCB cannot be found:

Option Code	Action
0	Issue diagnostic D002 DDNAME (name) UNDEFINED, then terminate task via ABEND.
1,2	Set output area to zero, then return to calling routine.

FINDJFCB may issue the message B022 WANT TO DEFINE DDNAME (name)? ENTER Y IF YES, N IF NO, when an undefined ddname is discovered in a conversational task and the option code is 1 or 0.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050502600	Minor	SYSTEM ERROR INVALID RETURN CODES	Invalid return code received from BUMP.
050502601	Minor	Same as above	Invalid return code received from MTREQ.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)

FLOW Command Processor (CZAGD)

The FLOW Command Processor allows the operator or manager/administrator to regulate the number of each different task type (batch, remote, conversational, MTT appplication, and MTT aplication users) that tne system will process concurrently, by allowing the manipulation of the various job type limit fields in the system. (See Chart BS.)

ENTRIES:

CZAGD1 - normal entry
CZAGE1 - subparameter processing entry

MODULES CALLED:

User Prompter (CZATJ1)	Informs the user of error conditions.
GATE (CZATC1)	Prints the buffer.
GETMAIN (CZCGA2)	Gets one page of storage for a buffer, if requested.
UFLOW Macro	Obtains or adjusts the conversational task limit or MTT application user limit.

EXITS: This routine returns to its calling routine.

OPERATION: On entry this routine checks for the appropriate authority code and privilege class. Only those with privilege class A, B, or F, and authority O or P can use the FLOW command.

This routine then begins to check the parameter values for reasonableness. The parameters that may be entered are BATCH, CONV, BACK, BULKIO, MTT, and APP. This routine checks the BATCH, BACK, and MTT parameters for exceeding maximum value. If they do not exceed the maximum

value, this routine inserts the value entered for them in the System Activity and Resources Table (CHASAR). Otherwise the maximum value is inserted in CHASAR. The CONV parameter is validated by issuing the UFLOW macro with action code 1 passed in Register 15. Action code 1 requests UFLOW to adjust the conversational task limit to the value specified in the low-order 2 bytes in Register 0.

If the APP parameter is entered, this routine links to CZAGE1 to build a list of subparameter addresses. Then, if there are active MTT users, this routine links again to CZAGE1 to build a list of application name addresses. A relative application number (RAN) may be associated with the APP parameter. If so, it is validated; if not, a default value is used. When the application names are exhausted, this routine uses the UFLOW macro with action code 3 passed in Register 15 to adjust the MTT user limits for the different application entries. This routine then calls GATE to print all the application names it has accumulated in the buffer. On return from GATE, this routine returns to its calling routine.

If no parameters are entered with the FLOW command, the user is requesting a display of task type counts and limits. This routine will extract the appropriate fields for BATCH, BULKIO, BACK and MTT from CHASAR and format them into a message. To complete the message, this routine executes UFLOW with action code 2. This operation obtains the current number of conversational tasks in execution and the conversational task limit, and returns it to the FLOW Command Processor, which includes this information in the formatted message. This routine then calls GATE to print the buffer. If there are active MTT administrators, this routine acquires another page of storage for its buffer, and executes UFLOW with action code 4 to obtain the relative application number (RAN) and current number of MTT users in execution for each application entry. On return this routine loads this print buffer with the MTT information and calls GATE to print it. When this operation is finished, this routine returns to its calling routine.

ERROR CONDITIONS: The FLOW Command Processor prompts the user and continues processing if the BATCH, CONV, BACK, or MTT parameter is specified with a value that exceeds the maximum allowed. The maximum limit is inserted in the table in these cases. The FLOW Command Processor prompts the user and exit immediately when the user is not authorized to use the FLOW command, or when it receives a nonzero return code from UFLOW.

SYSTEM CONTROL BLOCK USAGE: This routine references and manipulates the System Activity and Resources Table (CHASAR). The following limits, maximums, and count fields, as well as the BULKIO Suppress Flag, in CHASAR can be set or displayed by this processor:

BATCH	SARBTL	Batch task limit.
	SARNRM	Batch tasks in execution.
	SARMBT	Maximum number of batch tasks.
BACK	SARRML	Remote task limit.
	SARREM	Remote tasks in execution.
	SARMRM	Remote task maximum.
BULKIO	SARBAS	BULKIO Suppress Flag. If set to X'01' indicates BULKIO will not be initiated. If set to X'00' BULKIO will be initiated.
MTT	SARMAL	MTT Administrator task limit.
	SARMCA	Actual count of active MTT Administrators.
CONV	SARMCN	Maximum number of conversational tasks allowed.

IF String Comparison Routine (CZBLT)

This routine is a fast substitute for PCS IF processing when EBCDIC strings are being compared. (See Chart BT.)

ENTRY: CZBLT1 - normal entry

MODULES CALLED:

Source List Handler (CZASC6)	Updates the source list to an end-of-line or command.
PCS IF (CZAME1)	Processes IF statements which require additional processing by PCS.

EXITS: This routine exits normally to the calling routine, via the RETURN macro instruction.

OPERATION: The routine is entered at CZBLT1 from the CA&E when the IF verb is encountered. It tests '=' and '¬=' for true or false comparisons and sets a return code indicating the results of the test. Blanks on either side of equal signs are removed. If any of the following conditions are present, CZBLT calls PCS IF for additional processing:

- The phrase being processed is dynamic
- Argument is not a quoted string
- No terminating quote found in current string
- Embedded EOB encountered
- A character other than = or ¬ = is discovered between strings (such as <)
- Semicolon is not the delimiter

The source list is updated to the next command or end-of-line, whichever occurs first, and the routine exits with either a successful true or successful false condition.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE:

Source List Page Header (CHASLP)
Source List Sublist Header (CHASLH)
Source List Marker (CHASLM)

JOIN/REJOIN Command Routine (CZAFK)

These routines operate only in conversational mode to process the JOIN command by defining a legitimate new user to the system or the REJOIN command by redefining any parameter (exclusive of userid) and changing the user table entry accordingly. These commands can only be issued by a system manager or system administrator. (See Chart BU.)

ENTRIES:

CZAFK1 - JOIN entry
CZAFK2 - REJOIN entry

MODULES CALLED:

ALFNUM (CZAAC3P)	Validates user identification.

DELCAT (CZCPH1)	Deletes an entry in SYSVCT.
READ/WRITE (CZCPE1)	Operates on the User Table and the the User Limits Table.
FIND (CZCOJ1)	Locates the User Limits Table.
User Prompter (CZATJ1)	Accessed via the PRMPT macro; issues messages to user.

EXITS: The routine normally returns to the Command Analyzer and Executor, via the RETURN macro instruction.

OPERATION: After each operand of the JOIN command is fetched, it is validated according to the restrictions placed on the particular operand. Then, after validation and successful preliminary processing, the operand is stored, and the next one is fetched. When all operands have been successfully fetched, validated, and processed, they are placed in the user table so that the new user will be defined to the system as a legal user. Certain operands (userid, charge number, priority, privilege class, authorization, ration, batch, and RJE) require special checking and processing.

REJOIN follows the same processes as above except that only given parameters are checked and inserted. Defaulted parameters are ignored. This command makes it possible to alter a user's entry in the user table without first QUITing him, thereby losing all his data sets.

Userid: If the new user is being joined by a system administrator, the first two characters of his userid are prefixed to the userid operand. The operand is then checked for syntactic validity. A further check is made to ensure that the userid does not already exist in the system.

Password: This operand, if specified, is verified as having no invalid characters (such as parenthesis, comma, period, blank, etc.). If defaulted, an 8-byte all-blank password is assigned. A blank password may be specified by enclosing blanks within quotes.

Charge Number: Each user has a charge number. The charge number is prefixed with the first two characters of the user's newly established userid and checked for syntactic validity. No default value is permitted.

Priority: This operand, if specified, is verified as a value between 0 and 9 inclusive, and then converted from EBCDIC to binary. If the operand is defaulted, an installation-defined value is assigned.

Privilege Class: This operand, if specified, is first verified as an alphabetic character. Further checks are made to ensure that the user can assign this class and that this privilege class is acceptable to this installation. If defaulted, an installation-defined value is assigned.

Authorization: This operand, if specified, is verified as either U, P, or O. If defaulted, the operand is assigned an installation value.

Ration: This operand, if specified, is verified as a value between 1 and 9, inclusive. If defaulted, a value of 2 is assigned.

Batch: This operand, if specified, is verified as either Y or N. If defaulted, N is assigned.

RJE: This operand, if specified, is verified as either Y or N. If defaulted, N is assigned.

When all operands have been processed, a first-level catalog entry is created for the new user and the entry is placed in the user table. Finally, control is returned to the command analyzer and executor.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through the User Prompter, of any errors found.

SYSTEM CONTROL BLOCK USAGE:

Catalog Common (CHACDS)
System Common (CHASCM)
Task Definition Table (CHATDT)
Task Common (CHATCM)
Shared Data Set Member (CHASDM)
Data Control Block (CHADCB)
Task Status Index (CHATSI)
User Table (CHAUSE)
User Limits Table (CHAULT)
Interrupt Storage Area (CHAISA)
Active User Table (CHAAUL)

JOINRJE/QUITRJE Command (CZABS)

This command creates or deletes a station ID entry from the RJE acknowledgement data set TSS*****.RJEACK. (See Chart BV.)

ENTRIES:

CZABS1 - JOINRJE command entry
CZABS2 - QUITRJE command entry
CZABS3 - ABEND recovery entry

MODULES CALLED:

WRITE (CZCPE1)	VISAM write to enter a station id in member VALIDSTA.
ALFNUM (CZAAC3P)	Validates input parameters.
DDEF (CZAEA5)	Defines acknowledgement data set.
OPEN (CZCLAO)	Opens acknowledgement data set for update.
FIND (CZCOJ1)	Finds member of acknowledgement data set-VALIDSTA; finds VISAM member of STATIONID.
STOW (CZCOK1)	Creates or deletes acknowledgement data set member.
DELREC (CZCPH1)	Deletes VISAM records in member VALIDSTA.
User Prompter (CZATJ1)	Issues messages to SYSOUT.
SHAREUP (CZCFS1)	Permits the acknowledgement data set.
CLOSE (CZCLB)	Closes acknowledgement data set.
RELEASE (CZAFJ3)	Releases acknowledgement data set.
PAIR (CZACS1)	Queues entry for ABEND Interlock Release Table.
PAIR (CZACS2)	Dequeues entry from AIR table.

EXITS: The routine exits to the calling routine via the RETURN macro.

OPERATION: On non ABEND entry, this routine validates parameter input, utilizing ALFNUM. After parameter validation, the various options (TYPE, MRF, TAB, BRK, and REC) are checked and set. Then the DATADEF routine (CZAEA5) IS INVOKED TO DEFINE THE ACKNOWLEDGEMENT DATA SET TSS******.RJEACK, the data set is opened, and the Task Definition Table (TDT) is interrogated to determine disposition.

For disposition=new, the request must be for JOINRJE or the command is canceled via diagnostic message. The station id is entered utilizing VISAM WRITE, and the member VALIDSTA created via STOW. SHAREUP (CZCFS1) is then invoked to PERMIT/SHARE the TSS*****.RJEACK data set for later system use.

For disposition=old, on a JOINRJE request the member VALIDSTA is located via FIND, and the requested station id is entered into the data set using WRITE. On a QUITRJE request, the BULKCOMM table (CHABCT) is searched to determine if the station id being quit is still active in the Bulkio task. If active, a diagnostic is output canceling the command. For an inactive station id, entry point CZABS3 is queued on the AIR table for possible ABEND recovery with the BULKCOMM locked. A FIND is done on the requested station id to determine if acknowledgements are pending for that station id. If acknowledgements are queued in that member station id, the user is requested to specify their disposition. Based on the response, the command will be ignored, the acknowledgements will be deleted, or the acknowledgements will be left enqueued. When there are no acknowledgements queued, the station id is deleted from VALIDSTA by DELREC.

ERROR CONDITIONS: If an error is found during processing, the command is canceled and an appropriate diagnostic message is displayed on SYSOUT.

SYSTEM CONTROL BLOCK USAGE:

BULKCOMM Table (CHABCT)
BULKCOMM S-Entry (CHASET)
Task Common (CHATCM)
Task Definition Table (CHATDT)
Data Control Block (CHADCB)

JOBLIBS and DDNAME? Commands (CZAEK)

These commands allow a user to list the chain of DDNAMES and associated DSNAMES and to move any one of his JOBLIBS to the logical top of the list. (See Chart BW.)

ENTRIES:

CZAEK1 - for manipulating the JOBLIB chain
CZAEK2 - for reviewing user's JFCB chain

MODULES CALLED:

User Prompter (CZATJ1) Writes diagnostic messages.

GATE (CZATC1) Writes output to user.

EXITS: Returns to calling routine.

OPERATION: CZAEK1 is the entry point for manipulating the JOBLIB chain. Upon entry the parameter is checked for the presence of a DDNAME. If one does not exist, a diagnostic is initiated, followed by an exit from the module. Once a DDNAME has been obtained, the JOBLIB portion of the TDT chain is searched for this JFCB. If no match occurs, a diagnostic is issued and the command canceled. If the DDNAME is found in the

chain, it is removed from the chain by adjusting the appropriate backward pointer of the previous JFCB. Then it is put at the top of the list by placing its address in TDTPL1, and replacing the backward pointer with the address which was at TDTPL1.

Once the TDT chain is adjusted, the chain of DCBs in the Loader's PSECT is searched for the associated DCB. When it is found, the appropriate forward and backward pointers are changed as the JFCB pointers were, so that the DCB ends up at the top of the chain. This completes normal processing and the module exits.

If there were no JOBLIBS in the chain or the corresponding DCB could not be found, a diagnostic will be issued to the user and further processing canceled. In the latter case, the TDT chain will be rearranged as requested but the DCB chain will remain as is.

CZAEK2 is the entry point for reviewing the user's JFCB chain. Upon entry, the JOBLIB parameter is checked for 'N' or default. If such is the case, the TDT chain is addressed. If the JOBLIB parameter is 'Y', then the JOBLIB portion of the TDT is addressed. Next, a header message is initiated followed by the list of DDNAMES/DSNAMES of the JFCBs requested. For JOBLIB lists, the order of listing is from top to bottom; last DDEFed to first DDEFed. Once the end of the chain is reached, the module exits.

ERROR CONDITIONS: If there are no JOBLIBS assigned, or if the DDNAME does not exist in the JOBLIB chain, or if there is an invalid or missing parameter, the command is canceled. If no DCB is found, only the JOBLIB chain is reordered.

SYSTEM CONTROL BLOCK USAGE:

Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)

KEYWORD Command Routine (CZATH)

This routine is used to display keywords or parameters of commands in a user's SYSLIB and USERLIB. (See Chart BX.)

ENTRIES: The Display Keyword Format command has entry point CZATHC and is called by BUILTIN CZATH11.

MODULES CALLED:

GATE (CZATC1)	Outputs commands with their parameter keywords.
FIND (CZCOJ)	Searches USERLIB and SYSLIB.
Common OPEN (CZCLA)	Opens data sets.
Common CLOSE (CZCLB)	Closes data sets.
User Prompter (CZATJ1)	Outputs error messages.

EXITS: After all requested output has been printed (or queued up for nonconversational tasks), this routine returns to its calling routine.

OPERATION: This routine first determines if all or a specific command in USERLIB (SYSPRO) is being requested. If all, USERLIB (SYSPRO) is opened and a pointer positioned to the beginning of the data set. Then this routine searches through the data set, seeking out all command names and their parameter keywords (up to a maximum of 152 bytes). For PROCDEFs, the keywords are retrieved starting from line 100 of the procedure. For BUILTINs, the module ID is taken from the SYSPRO entry and

stored in an adcon group. Then the adcon is armed for subsequent issuance of a LOAD macro. Once the module is loaded, the parameter keywords are retrieved from BPKD. In any case, all commands, with their parameter keywords, are printed one command string per line.

When the user has asked for a particular command, this routine searches USERLIB (SYSPRO) first, and, if not found there, searches SYSLIB (SYSPRO). Once the command is found, its parameter retrieval process is the same as described above for PROCDEFs and BUILTINs. If the command cannot be found, this routine issues a diagnostic (via the PRMPT macro) indicating the situation, and processing terminates.

ERROR CONDITIONS: If the command cannot be found, the user is prompted, and this routine exits to its calling routine.

SYSTEM CONTROL BLOCK USAGE: Interrupt Storage Area (CHAISA)

LINE? Command Routine (CZAEM)

This routine presetns the contents of specified lines from a user's line data set or language processor list data set. (See Chart BY.)

ENTRIES:

CZAEM1 - normal entry
CZAEME - EODAD error entry
CZAEMS - SYNAD error entry
CZAEM2 - DCB macro instruction entry

MODULES CALLED:

NEXTPAR (CZAAC1)	Locates operands and scan for invalid characters and delimiters.
CHEKDS (CZAAC2)	Validates data set name.
ALFNUM (CZAAC3)	Validates user identification.
NUMSTG (CZAAC4)	Validates line numbers.
MSGWR (CZAAD2)	Issues system messages.
GATWR (CZATC1)	Writes line on SYSOUT.
DDEF (CZAEA5)	Creates a JFCB.
FINDDS (CZAEC1)	Locates a JFCB.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The data set name is fetched through NEXTPAR. If the user is class B or F, the data set name must be prefixed with a user identification; this userid is removed, validated (via ALFNUM), and saved for future use. For a class B user, a further check is made to ensure that the first two characters of the userid are the same as the first two characters of the userid of the administrator issuing the command. The data set name is validated using CHEKDS. If the name is a valid partitioned data set name, the member name is removed and saved for future use.

After the data set has been successfully opened, and if the first qualifier (excluding userid) of the data set name is LIST, data set control block fields are checked to verify that the data set is in language

processor list format. (A language processor list data set must be index sequential, must have a key length of 7 bytes, a logical record length of 140 bytes, and record format must be fixed length.) If the first qualifier is not LIST, the data control block fields are checked to vierfy that the data set is in line format. (A line data set must be index sequential or partitioned index sequential, must have a key length of 7 bytes, must have RKP=4, a logical record length of no more than 132 bytes, and the record format must be variable length.) If the line data set is partitioned index sequential, the FIND macro instruction is used to locate a specified member.

If the entire data set (or member of a line data set) is to be presented, the SETL macro instruction is used to designate the start of the data set (or member). The first logical record is then fetched by GET and presented via GATWR. Before and after each line is presented, a test is made for an attention interrupt. If one is detected, the data set is closed and control is returned to the command analyzer and executor. If no attention interrupt has been received, GET obtains the next logical record and the process is repeated. Sequential reading and presenting continues until GET tries to obtain a record that is past the end of the data set. At this point, the user is informed that the end of the data set has been reached, the data set is closed, and control returns to the command analzyer and executor.

If only a single line is to be presented, NEXTPAR fetches the line number; SETL points to the line; GET reads the line; and GATWR writes the line on SYSOUT. If a range of lines is to be presented, the process is the same except that NEXTPAR fetches both the beginning-of-range and end-of-range line numbers. When GET obtains a record with a key higher than the end-of-range line number, processing is complete.

After each line or range of lines has been processed, a check is made to determine if there are any further lines to be processed from the current data set. If there are, processing continues. If not, the data set is closed and control is returned to the command analyzer and executor.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through MSGWR, of any error found. The routine may issue the following system errors. In each case the severity is minor.

SYSER Code	Significance
050503700	Response switch set with incorrect value.
050503701	Invalid return code from NEXTPAR.
050503702	DCBEX2 field contains unexpected value.
050503703	Privilege class indicator set incorrectly.
050503704	Indicator for subfield of data set name set incorrectly.
050503705	Beginning-of-range indicator set incorrectly.
050503706	Beginning-of-range indicator set incorrectly.
050503707	Beginning-of-range indicator set incorrectly.
050503708	Prompting switch set incorrectly.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Task Common (CHATCM)

LOGOFF Command Routine (CZAFN)

This routine logs a task off the system. That is, it disposes of any uncataloged data sets that were defined by the task, releases all devices allocated to the task, and frees the task's virtual storage for system use. When LOGOFF is completed, the task no longer exists. (See Chart CA.)

ENTRIES:

CZAFN1 - normal entry
CZAFN2 - shutdown entry
CZAFN4 - 'END' exit function

MODULES CALLED:

VSS DISCONNECT (CZHNEB)	Disconnect VSS from the task.
FINDJFCB (CZAEB1)	Finds the SYSOUT DCB for a nonconversational task.
Loader Logoff (CZCCD4)	Unloads the user routines; releases the virtual storage assigned to the task.
Cleanup (CZCJCU)	Refresh the Task Monitor Interrupt Table.
RELEASE (CZAFJ2) and (CZAFJ3)	In stages, releases all but the system data sets, releases SYSULIB, releases all data sets.
Common Close (CZCLBC)	Closes the SYSIN and SYSOUT DCBs for a nonconversational task.
BIO Preprocessor (CZABD3)	Prints SYSOUT for a nonconversational task.
CATFLUSH (CZCFX1)	Flushes the catalog.
VMTI-II (CZATD4)	Logs on the next subtask for express batch mode.
RCR VACATE	Decrements the user count in the User Table for a nonconversational task.
RCR LOGOFF	Computes and saves all resource information for the task and updates the User Table.
RCR CLOSE	Subtracts the user's entry from the User Table.

The LOGOFF routine also uses FREEQ, DCON, DELVAM, WTL, EBCDTIME, CHANGE, ACCTSUBR, XTRTM, XTRACT, and ATCS during its processing.

EXITS: Should an error occur during LOGOFF processing, this routine exits to the ABEND processor. The normal exit from LOGOFF processing is via the DLTSI SVC, which deletes the task status index for the task, thus eliminating the task from the system.

OPERATION: An entry at CZAFN2, indicating that a system shutdown is in progress, causes LOGOFF to act as if it were processing a nonconversational task and prevents prompting messages. An entry at CZAFN1 is con-

sidered a normal entry; however, if the operator task enters here, control returns to the command system -- the operator cannot successfully execute a LOGOFF. Operation of this routine is divided into two major parts. The first part is always performed; the other part is only performed when the task has uncataloged, defined data sets.

LOGOFF always carries out a series of actions to eliminate the task from the system. Initially, it sets an indicator in the interruption storage area (ISA) to prevent any recursions of the ABEND routine. For nonconversational tasks which are not in Express Batch mode, LOGOFF informs the Batch Monitor that it may permit updating of the batch work queue. A complete logoff is then performed for tasks which are not in Express Batch mode by:

1. Issuing a FREEQ to logically or physically disconnect this task from MTT. If SHUTDOWN is in progress, a physical disconnect is done. Otherwise a logical disconnect is issued.

2. Issuing a FIND on the sysout JFCB for a nonconversational task.

3. Checking STE level for error, and issuing the CHANGE macro if the STE level is in error. A bad return code from CHANGE results in system error 050506305.

4. Issuing the LOGOFF accepted message.

5. Calling the accounting subroutine -- ACCTSUBR -- to record this task's use of the system resources.

6. Closing the SYSIN and SYSOUT data sets for a nonconversational task, and issuing a print on sysout via PRINT.

7. Issuing the ATCS macro for conversational tasks only; a logical disconnect is normally performed. In the case of a shutdown a physical disconnect is issued.

8. Purging the terminal from the TSI to allow a new task to be created at this terminal.

9. Releasing the task's virtual storage via a call to LOADER LOGOFF.

10. Releasing all data sets used by the task by a call to RELEASE command.

11. Updating the user table entry for the current user and deleting the AUL entry for the task. This is done by RCR LOGOFF.
Note: A nonconversational task will call RCR VACATE previous to calling RCR LOGOFF.

12. Closing the user table.

13. Calling VSS DISCONNECT (CZHNEB) if the task has been connected to VSS.

14. And finally, issuing the DLTSI SVC to erase the task status indicator and thereby eliminate the task completely.

A complete logoff is not performed for an express batch subtask. A partial logoff is performed, the first four of the functions above are done, then the partial logoff is completed by:

1. Calling LOADER LOGOFF (CZCCD4) to unload user called routines.

2. Calling RELEASE (CZAFJ2) to release all but system data sets and performing a special release of SYSULIB.

3. Performing an RCR VACATE for the task and an RCR CLOSE to update the user's entry.

4. Calling VMTI-2 (CZATD4) to logon the next subtask.

Note: When the SYNAD exit is reached in express batch (that is, no more subtasks to be logged on), CZAFN4 is entered. The userid in task common is changed to SYSOPER0 and a complete logoff is performed (steps 5-14 in a normal logoff sequence).

LOGOFF disposes of uncataloged data sets created by the task (found in the temporary tabulation chain of the task data definition table).

ERROR CONDITIONS: The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050506305	Minor	ERROR ATTEMPTING TO CHANGE STE LEVEL	Error return code from CHANGE.
050506306	Minor	NO JFCB IN TDT SYSOUT	FINDJFCB determined that no JFCB was included for SYSOUT.
050506307	Minor	UNABLE TO RELEASE ALL	Error return code from RELEASE.
050506308	Minor	INVALID RETURN FROM PRINT	Error return code from PRINT.
050506310	Minor	MINOR SOFTWARE ERROR	Error condition detected by RCR LOGOFF; or RCR CLOSE for a express batch subtask.

SYSTEM CONTROL BLOCK USAGE:

Terminal Control Table (CHATCT)
Interrupt Storage Area (CHAISA)
Symbolic Device Allocation Table (CHASDA)
Task Common (CHATCM)
User Table (CHAUSE)
Task Data Definition Table (CHATDT)
Data Control Block Header (CHADHD)
RESTBL Header (CHARHD)
Active User Limits Table (CHAAUL)
Task Accounting and Statistical Table (CHAACT)

LOGON (CZAFM)

The LOGON command routine validates the user's LOGON parameters (his user identification, charge number, etc.) and carries out certain task initialization functions, such as setting values in task common. The user cannot proceed with his task until the LOGON routine has been successfully completed. (See Chart CB.)

ENTRIES:

CZAFM1 - normal entry
CZAFM2 - privileged entry for SYSOPER0, ABEND, BULKIO (TID=0002)
CZAFM4 - privileged entry from private BULKIO

MODULES CALLED:

NEXTPAR (CZAAC1)	Locates next operand.
DDEF (CZAEA4)	Creates JFCB for the user library (USERLIB).
RCR OPEN	Reads an entry in user table and update fields; create AUL entry.
RCR RATION	Processes task, CPU and connect time.
RCR VACATE	Vacates resources assigned to task.
RCR CLOSE	Resets and writes user table entry.
XTRCT	Fetches authority from Task Status Index.
SETUP	Stores information in Task Status Index.
SCHED	Alters task's internal priority.
EBCDTIME	Gets current time.
WTL (CZABQ)	Issues logon message to operator log and issues ABEND message for a nonconversational task ABEND in LOGON.
GATWR (CZATC)	Issues shutdown message.
LOGOFF (CZAFN1)	Logs a task off when specified auxiliary space is not available.
User Prompter (CZATJ)	Writes the initial logon message on the user's SYSOUT.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error is detected, the routine exits to ABEND.

OPERATION: LOGON first determines what kind of task is logging on. For any conversational task, an ATTACH macro is issued which returns the address of the terminal control slot for this user. If the TCT slot's address is zero, LOGON exits to ABEND. Otherwise, this address is stored in GATE's PSECT. The LOGON parameters are pointed to by a field in the TCT slot for every conversational task except an abended task. GATE's PSECT is then initialized with information retrieved from the TCT slot, including the device type and a maximum SYSOUT line length. For nonconversational and abended tasks, the parameter string is pointed to in register one. For other than BULKIO and main operator tasks, LOGON then calls the User Prompter (CZATJ) to print the initial logon message.

A complete LOGON consists of checking the LOGON command operands. Actual fetching and validating of each operand for proper syntax is done through calls to the SCAN's NEXTPAR routine. First the userid is validated by a call to RCR OPEN. When the userid has been determined to be valid (that is, a read on the user table is good), an AUL entry is built for the user, an AUL entry is constructed for USERID=TSS*****, and the user table is updated. If the task is nonconversational, a call is made to CZBTBA to open the SYSOUT data set for this task. The routine next tests the 'first time through' switch for an express batch task. If on, a GATWR is issued to write the EXPRESS card to the task's SYSOUT. This is followed by another GATWR (issued for all nonconversational tasks) which writes the user's LOGON parameters to SYSOUT.

The LOGON parameters are then validated one at a time. If the task is conversational and the user table entry corresponding to the userid contains a valid password (not blanks), the password is checked. If the user has been joined with a valid password, but has defaulted this at LOGON time, a special prompt will be issued conversationally to have this parameter entered. The addressing operand is processed next and fields in the ISA are set accordingly. Then if a charge number is supplied, it is placed in task common; if the charge number has been defaulted, the charge number from the user table entry is placed in task common. The CSECT packing operand is checked, followed by the AUX space control parameter, the pristine parameter, and lastly the XIVM parameters.

As each command operand is accepted, a part of the LOGON routine's initialization is performed. Thus the userid, password, and charge number are moved one at a time to task common. The CSECT packing operand is moved into new task common and also into the dynamic loader's PSECT. If a PRISTINE LOGON has been requested the character string 'pristine' is moved into new task common. If any of the LOGON operands are invalid, the user is informed of his error by an appropriate message and can reenter the entire operand string. This prompting loop will continue until the system's prompting limit has been reached; at this time the task will be abnormally terminated.

When all parameters have been determined to be valid, the routine performs the following additional functions:

1. Completes fields in the task status index (TSI) by moving userid, external priority, and authorization values into it;

2. Adds further information to task common. Values are set to show the user's privilege class, completion of logging on, no confirmation messages and full messages.

3. Calculates the internal priority of the task via a call to the SCHED SVC;

4. Creates a job file control block (JFCB) for the user library (SYSULIB) through a call to DDEF. Note: This is not done if the user has specified PRISTINE=X in the LOGON operand string.

After completing all its initialization functions, the routine issues two messages. One of these is to the task's SYSOUT noting date and time of LOGON. The other message is issued to the system log noting the userid, time, and date of LOGON initialization. Control is then passed to the calling routine.

ERROR CONDITIONS: The routine will check for acceptable operands and notify the user, through the User Prompter, of any errors found. For the messages issued, see the description of the LOGON command in Part I.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050506207	Minor	DDEF UNABLE TO CREATE A JFCB FOR SYSULIB	DDEF was unable to create JFCB for SYSULIB.
050506208	Minor	UNABLE TO WRITE THE USER TABLE	An attempt was made to write in user table but SYNAD exit was invoked.

050506211	Minor	NO JFCB FOUND FOR USER TABLE	No JFCB exists for user table in TDT chain.
050506212	Minor	RCR CLOSE FAILURE	RCR CLOSE failed.
050506213	Minor	RCR OPEN DETECTED SYSTEM ERROR	RCR OPEN could not perform its function.

SYSTEM CONTROL BLOCK USAGE:

Interrupt Storage Area (CHAISA)
Task Status Index (CHATSI)
System Common (CHASCM)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)
User Table (CHAUSE)
Task Dictionary Table (CHATDH)
Terminal Control Table (CHATCT)
Task Common (CHATCM)
New Task Common (CHANTC)
Shared Data Set Member (CHASDM)
Active User Limits Table (CHAAUL)

LOGON2 Command Routine (CZBTB)

This privileged routine completes the LOGON functions that must be done within the user's task. LOGON2 functions are accomplished in three main routines: TASKOPEN, TABINIT, and GATEOPEN. TASKOPEN opens all system data sets needed by the user's task (except SYSIN and SYSOUT) and finds all needed members. TABINIT retrieves members from SYSLIB and USERLIB, and constructs the command system dictionaries and tables in virtual storage. GATEOPEN opens the SYSIN and SYSOUT devices for non-conversational tasks and stores information needed by GATE. (See Chart CC.)

ENTRIES:

CZBTB1 - LOGON2 control
CZBTB2 - TASKOPEN
CZBTB3 - TABINIT
CZBTB4 - GATEOPEN (SYSIN only)
CZBTB5 - VAM EODAD routine
CZBTB6 - VAM SYNAD routine
CZBTB7 - SYSLIB DCB
CZBTB8 - USERLIB DCB
CZBTB9 - PROCLIB SCAN
CZBTBA - GATEOPEN (SYSOUT only)
CZBTBB - EXPRESS BATCH
CZBTBX - DICTEXTR

MODULES CALLED:

Dictionary Handlers:

STARTVAR (CZASD2)	Initializes the dictionary; called by CZBTB3.
RFR (CZASD3)	Uses the hash chain to locate an entry in the dictionary; called by CZBTB3 and CZBTBX.
NEXTRFR (CZASD4)	Checks the next entry in the dictionary; called by CZBTBX.
ENTR (CZASD5)	Makes a new entry in the dictionary; called by CZBTB3 and CZBTBX.

RELEASE (CZAFJ3)	Release the SYSOUT JFCB; called by CZBTBA.
DDEF (CZAEA4)	Defines a new SYSOUT JFCB; called by CZBTBA.
FINDJFCB (CZAEB1)	Locates the SYSIN JFCB; called by CZBTB4.
User Prompter (CZATJ1)	Sends messages to the user.

Various routines within LOGON2 also issue FIND, OPEN, GETMAIN, ESETL, SETL, GET, PUT, STOW, and CLOSE. CZBTB1 executes a SIR macro.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: LOGON2 is called to complete the logon functions. New Task Common (CHBNTC) is initialized; then all system data sets needed by the user task are processed. LOGON2 processes four of the system data sets: SYSLIB, USERLIB, SYSIN, and SYSOUT. If the user has entered the "pristine" operand with his LOGON command, USERLIB DCBs are not processed in LOGON2; only SYSLIB DCBs are opened, otherwise both USERLIB and SYSLIB DCBs are processed.

A pristine request will cause SYSLIB (SYSPRO) to be opened. LOGON2 will issue a series of FIND command for the SYSPRO and the SYSPRD members of SYSLIB, so that the DCBs will be correctly initialized and the members ready for reference by the Procedure Expander (CZATE). Where several members of the same data set are needed simultaneously, a DCB is generated for each member. A normal logon request will cause the above to be done, in addition the SYSPRD, SYSPRO, and SYSMLF members of USERLIB will be opened, and the FIND command issued on the SYSPRO and SYSMLF members. SYSMLF must be initialized for reference by the User Prompter.

LOGON2 then builds the combined dictionary in the Dictionary Handler (CZASD) PSECT. The origin of the dictionary is indicated by a one-word pointer and the pointer is saved in the external symbol CZASD9. The initial size of the dictionary is one page. If it must be expanded, it is done by a Dictionary Handler routine. GETMAIN is used to obtain the referenced pages and the CZASD9 pointer is updated accordingly. If the pristine option has been requested, the combined dictionary will be built from SYSLIB only; otherwise, both SYSLIB and USERLIB information will be used.

The combined dictionary contains entries from four main sources.

1. The system procedure dictionary -- SYSLIB (SYSPRD).

2. The user procedure dictionary, if any exists -- USERLIB (SYSPRD). If there is no user procedure dictionary, one will be constructed from the user's procedure library -- USERLIB (SYSPRO) -- if one exists. When built, this dictionary will be used and written into USERLIB.

3. The primary dictionary from the user profile (SYSPRX).

4. The primary dictionary from the system profile for a pristine task or if no user profile exists.

The input character translation table is moved into GATE's PSECT (CZATC5) and the PCTCTT pointer is set. The output character translation table is moved into GATE's PSECT (CZATCT) and the PCTOCT pointer is set. The profile character and switch table is moved into new task common. The SYSLIB and USERLIB DCBs opened for SYSPRD and SYSPRX are then closed.

The ZLOGON procedure is executed using the OBEY macro instruction. Then LOGON2 enables the CSII Attention Handler (CZASB1) by a SIR macro.

ERROR CONDITIONS: The following error conditions result in an ABEND:

1. USERLIB (SYSPRD) - Invalid format.
2. SYSULIB (SYSPRD) - Invalid hash pointer.
3. USERLIB (SYSPRX) - Invalid format.
4. Read past end of dictionary.
5. VAM error, SYNAD routine called.
6. USERLIB (SYSPRD) - Invalid hash pointer.
7. Return code from FIND of USERLIB (SYSPRD) of 8, C, or 10.

SYSTEM CONTROL BLOCK USAGE:

Editable Data Set (CHACVF)
Data Control Block (CHADCB)
Control Dictionary Header (CHADCT)
Control Dictionary Entry (CHADEN)
New Task Common (CHANTC)
Profile Character and Switch Table (CHAPCT)
Task Common (CHATCM)
User Profile (CHAPFL)
Symbolic Device Allocation Table (CHASDA)
Task Data Definition Table (CHATDT)

MCAST/MCASTAB Routine

This routine makes changes in the Profile Character and Switch Table as specified in the parameters of the MCAST command or macro instruction. It also allows the user to specify, through the MCASTAB command, his own Input and/or Output Translation Table. (See Chart CD.)

ENTRIES:

CZATU1 - MCAST macro entry
CZATU2 - MCAST command entry
CZATU3 - MCASTAB command entry

MODULES CALLED: This routine calls the User Prompter (CZATJ1) via the PRMPT macro to inform the user of error conditions during command processing.

EXITS: This routine returns to its calling routine via the RETURN macro instruction.

OPERATION: The input parameter list for MCAST specifies the address of each replacement value for the Profile Character and Switch (CHAPCT) Table. For macro processing, this routine moves each value from its location in the calling module to the proper location in CHAPCT. For MCAST command processing, this routine does some error checking. Unless the parameter string is hexadecimal, the parameter length must equal 1 to be moved into the CHAPCT table. Only nine separate parameters will be processed. The MCAST command processor will translate hexadecimal input via its internal subroutine HEXTRAN.

To process the MCASTAB command, this routine validates the input parameter and, if valid and if the user wishes to specify his own table, sets the address to that table in CHAPCT. If the user wishes to discon-

nect his Input and/or Output Translation Table, this routine sets a pointer to the system's Translation Table in CHAPCT.

ERROR CONDITIONS: This routine prompts the user and exits if, during MCASTAB command processing, it recognizes an invalid parameter. During MCAST command processing, this routine prompts that it is ignoring any parameter found to be invalid but does not exit until the parameter list is exhausted.

SYSTEM CONTROL BLOCK USAGE:

Profile Character and Switch Table (CHAPCT)
Input Translation Table
Output Translation Table
TABLEA (CHAAAA)

MODIFY Command Routine (CZAEG)

This routine inserts, deletes, replaces, and reviews records in a VISAM data set or VISAM member of a partitioned data set; or builds a new VISAM data set or member. (See Chart CE.)

ENTRIES:

CZAEG1 - normal entry
CZAEG2 - SYNAD entry
CZAEG3 - EODAD entry

MODULES CALLED:

NEXTPAR (CZAAC1)	Locates delimiters of input string and scan for invalid characters.
CHEKDS (CZAAC2)	Validates data set name.
ALFNUM (CZAAC3)	Validates member name.
CHKNUM (CZAAC5)	Validates line number.
MSGWR (CZAAD2)	Issues system messages.
GATWR (CZATC1)	Writes on SYSOUT.
SYSIN (CZASC7)	Reads data from the source list.
FINDDS (CZAEC1)	Finds or builds JFCB.
DDEF (CZAEA4)	Defines a new data set and creates JFCB for it.
STOW (CZCOK)	Updates POD when new member is created.
GETMAIN (CZCGA2)	Gets additional virtual storage.
FREEMAIN (CZCGA3)	Releases virtual storage.

EXITS: The routine normally returns to the calling routine via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: After operands are fetched using BPKD and validated using the SCAN routines, an attempt is made to obtain a job file control block (JFCB) for the data set. If the data set has been previously defined or cataloged, a JFCB will be created for it. In either case the JFCB must show that the data set has VISAM or VAM partitioned organization and is write-accessible to this user.

When the JFCB is successfully located or created, the data set is opened with a provision for both reading and writing by the user. If the data set is partitioned, the partitioned organization directory (POD) must be searched for the specified member name. When the data set or member name is found, a check is made to ensure that it has VISAM organization. If the name is not found, a new data set or member is created with this name and VISAM organization.

Input records containing the user's modifications are obtained one at a time, via SYSIN, until the end-of-input record (%E or an underscore followed by a command) is reached. Input characters are EBCDIC unless a X% is found. Hexadecimal mode is assumed until a nonhexadecimal character (for example, G) or end-of-block (EOB) is found. The characters found within the hexadecimal string are converted to hexadecimal, character for character. When the end-of-input record is reached, the data set is closed. For a partitioned data set, the POD is updated to reflect any alterations before the data set is closed.

Modifications are effected by employing the user-supplied key (a line number, if the data set is in line format) to point out the location of the specified record (line). When the first character of input supplied by the user is numeric, the record is written into the data set as an insertion or replacement for an existing record. When the first character supplied by the user is D, the record at the specified location is deleted from the data set. When the first character is R, the record at the specified location is reviewed (presented to the user). If review of all modifications is requested, the existing record which is being replaced or deleted is presented to the user before the actual modification is done. In the case of an insertion, the record immediately preceding the insertion is presented. In confirmation mode the new record (either an insertion or replacement) is also presented.

When a new nonline data set is to be created, an additional page is added to MODIFY's virtual storage since the nonline input record may be as much as 4K bytes in length. This type of data set is built of successive readings of input records containing continuation characters. To build a line data set, each input record is processed individually with continuation characters disregarded.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For messages issued, see System Messages.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050503102	Minor	SYSTEM ERROR. INVALID RETURN CODE	Invalid return code from called module.
050503104	Minor	SYSTEM ERROR. TROUBLE IN WRITE MACRO	SYNAD error when trying to write replacement line in data set.
050503105	Minor	SYSTEM ERROR. TROUBLE IN WRITE MACRO	SYNAD error when trying to write insertion in data set.
050503108	Minor	SYSTEM ERROR. MEMBERNAME	No left parenthesis in member name.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)

MSGWR (Message Write) Routine (CZAAD)

This routine issues system messages and, if requested, fetches responses to those messages, using the facilities of the USER PROMPTER (CZATJ). (See Chart CF.)

ENTRIES:

CZAAD2 - privileged entry
CZAAD3 - nonprivileged entry

MODULES CALLED:

User Prompter (CZATJ) Issues messages and obtains response (if any).

EXITS: The routine returns to the calling routine, via the RETURN macro.

OPERATION: MSGWR is entered to issue system messages. Either of two entry points is used, depending on the object code generated by the MSGWR macro. The macro expansion checks whether the macro user is privileged or not, then generates linkage to MSGWR accordingly.

MSGWR converts the parameter list it receives to a form compatible with that which the PRMPT macro would generate for an equivalent call to USER PROMPTER. MSGWR then branches to the USER PROMPTER, who handles the actual message output and any expected response.

After the message has been issued, via USER PROMPTER, the response (if any) is collected and MSGWR returns to its caller. A return code is set to indicate results.

ERROR CONDITIONS: A hexadecimal code will be returned in register 15:

Code	Significance
00	No error detected.
04	Truncation of output to terminal; or truncation of response from terminal. Truncated response not moved to user-provided area.
08	I/O terminated by ATTN interruption.

SYSTEM CONTROL BLOCK USAGE: None.

Place Address in AIR Table (CZACS) - PAIR

This routine adds and removes addresses of interlock release routines in the ABEND Interlock Release (AIR) Table. PAIR also controls an area within the AIR table for temporary storage of interlock release routine control information. (See Chart CG.)

INPUT: Register 1 points to a three-word parameter list.

Word 1: VCON of interlock release routine.

Word 2: RCON of interlock release routine.

Word 3: CZACS1 returns the address of a doubleword that the caller may use to store variable information. CZACS2 uses the contents of this word (the address CZACS1 had returned) to determine which entry to delete from the AIR table.

ENTRIES:

CZACS1 - entry to place address in table
CZACS2 - entry to remove an address from table

MODULES CALLED:

GETMAIN (CZCGA2) Obtains additional virtual storage.

EXITS: This routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: To place an interlock release routine address in the AIR table, PAIR searches the table until it finds a pair of zero AIRVCN and AIRRCN fields. It then places the VCON address of the interlock release routine in the AIRVCN field, and the RCON address of the routine in the AIRRCN field. The temporary storage area (AIRINF) address associated with the AIRVCN and AIRRCN field is placed in the input parameter list and control is returned to the calling routine.

To erase an interlock release routine address from the AIR table, CZACS2 verifies the address of the AIRINF field which it gets from the third word of the parameter list and then sets this field to zero along with its associated AIRVCN and AIRRCN fields. Control is then returned to the calling routine.

ERROR CONDITIONS: A hexadecimal code is returned in register 15:

Code	Significance
04	No space available in AIR table (for CZACS1)
08	AIRINF address invalid (for CZACS2)
0C	Parameter list not on a word boundary
10	Invalid or zero parameter passed to PAIR routine.

SYSTEM CONTROL BLOCK USAGE: ABEND Interlock Release Table (CHAAIR)

PERMIT Command Routine (CZAFH)

This routine enables a catalog owner to authorize shared use of some or all of his cataloged data sets by some or all other users. The owner may subsequently change or retract such authorization. (See Chart CH.)

ENTRY: CZAFH1 - normal entry

MODULES CALLED:

NEXTPAR (CZAAC1) Gets command operand.

CHEKDS (CZACC2) Validates data set name.

SHARE (CGCFS) Adds user identifications to sharer's list.

UNSHARE (CGCFV) Removes user identifications from sharer's list.

MSGWR (CZAAD2) Issues system messages.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The first operand read in is either a data set name or the word *ALL, which indicates that the user is offering to share all of his cataloged data sets.

The sharer's identification is next in the input string, however the sharing access is checked first. It is validated and a coded version of it is stored in the parameter list being built for catalog services. If the access qualifier is R, meaning that sharing privileges are being revoked, PERMIT calls UNSHARE rather than SHARE. If the access qualifier is defaulted, this is so indicated and the last entered access on the sharing list being updated is used.

The next operand shows the identifications of the users being granted access. If none exists and the owner indicates he has purposely defaulted the operand, or if the operand is *ALL, the universal sharing mode is indicated in the parameter list being built. If a list of user identifications has been entered, each userid is read, validated, and then added to the sharer's list. A count of the number of sharers (which is limited to 25 for each PERMIT command) is kept and supplied, along with the address of the start of the list, as input to SHARE or UNSHARE.

Either SHARE or UNSHARE makes the appropriate changes in the owner's catalog. Note that PERMIT calls the catalog service's SHARE (CGCFS), which should not be confused with the SHARE command (CZAFI).

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For messages issued, see System Messages.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Explanation
050505702	Minor	OWNER ID NOT IT CATALOG	Owner's user identification not found in catalog.
050505703	Minor	INVALID RETURN CODE FROM CATALOG SERVICE ROUTINE	SHARE or UNSHARE returned an invalid code.
050505704	Minor	IMPROPER HANDLING OF STATE AND MODE BY PERMIT	SHARE return code invalid.
050505705	Minor	INVALID RETURN CODE FROM NEXTPAR	NEXTPAR return code invalid.
050505706	Minor	INVALID RETURN CODE FROM CHECKDS	CHECKDS return code invalid.

SYSTEM CONTROL BLOCK USAGE

Task Common (CHATCM)
Interrupt Storage Area (CHAISA)
Data Control Block (CHADCB)

▶POD? Command Routine (CZCOX)

This routine is used to print the member names (and, optionally, the aliases and other member-oriented data) of individual members of a cataloged VPAM data set. (See Chart CI.)

ENTRY: CZCOX1 - normal entry

MODULES CALLED:

User Prompter (CZATJ)	Writes diagnostic messages on task SYSOUT.
GATE (CZATC1)	Issues output to task SYSOUT via GATWR.
LOCATE (CGCFL1)	Determines the accessibility of the requested data set.
GETMAIN (CZCGA2)	Acquires a work area in storage for the POD and optionally the PMD.
FREEMAIN (CZCGA3)	Frees the work area previously acquired.
Set Interlock (CZCOH1)	Denotes the RESTBL of the requested data set being interlocked.
Release Interlock (CZCOI1)	Removes the interlock from the RESTBL of the required data set.
FINDDS (CZAEC1)	Locates the JFCB for the requested VPAM data set using dsname.
RELEASE (CZAFJ3)	Releases the JFCB and private devices.
RELEAS (CZCAD1)	Releases private device.
CHEKDS (CZAAC2)	Validates input dsname.

EXITS: All exits are to the Command Analyzer and Executor (CZASA1) via type-I linkage.

OPERATION: Upon entry, POD? finds a pointer in register 1 to a four-word parameter list. The first word contains the dsname address; the second word is the address of the DATA option; the third word contains the address of the ALIAS option; and the fourth word is the address of the module option. An analysis is made of the parameters. Diagnostics are issued and the routine returns if any of the parameters are incorrect. If they are all correct, LOCATE is called to see if the data set is accessible to the user; if it is not accessible, a message is issued via User Prompter and the routine returns. Otherwise, FINDDS is called to create or locate a JFCB. A DCB is provided in the POD PSECT, and its ddname is set to that of the JFCB. The OPEN subroutine is then called to provide a link between the DCB and the RESTBL DCB header. The POD address is acquired from the DCB header, the data set is write interlocked, virtual storage space is allocated, the POD is moved into the work area, and the interlock is released.

The hash table entries of the POD are examined for hash chaining, and each hash chain is examined for member entries. As each member is found, it is printed along with POD and user data, alias entries and module information, if these options have been selected. After all member names have been printed, the user is informed and the data set is closed. If a JFCB was created, it is released. If no device was mounted and no JFCB created, a call to CZCAD1 is made to free the device. The area required for the POD, and optionally the PMD, is freed and the routine returns to the Command Analyzer and Executor.

ERROR CONDITIONS: User Prompter is called to issue diagnostic messages if any of the following conditions exist:

1. No data set name supplied.

2. Data set does not exist.

3. Data set cannot be shared by user.

4. Data set not partitioned.

SYSTEM CONTROL BLOCK USAGE:

Catalog SBLOCK (CHACCC)
Data Control Block (CHADCB)
Data Control Block Header (CHADHD)
Partitioned Organization Directory (CHAPOD)
POD Alias Descriptor (CHAPOE)
POD Member Descriptor (CHAPOM)
RESTBL Header (CHARHD)
Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Task Dictionary Table (CHATDY)

PRMPT Command Routine (CZBTC)

PRMPT allows the user to display the standard or extended form of a message in SYSMLF, the message file. He may insert up to five parameter strings if the message has variable parameters. (See Chart CJ.)

ENTRIES:

CZBTC1 - normal entry
CZBTC2 - BPKD macro entry

MODULES CALLED:

User Prompter (CZATJ1) Finds and writes the message.

EXITS: The routine returns to the calling routine, via the RETURN macro instruction.

OPERATION: The parameters from the BPKD macro are used to issue a PRMPT macro with up to five inserts. If the MSGID is null, a blank MSGID is assumed. If it has less than eight characters, the field is padded with blanks. If it has more than eight, the excess is truncated. The inserts are checked to see if any exceeds the length limit of 40 characters. If so, the insert is truncated. There are no error return codes.

ERROR CONDITIONS: None of the user prompter errors are checked.

SYSTEM CONTROL BLOCK USAGE: None.

PROCDEF Routine (CZATP)

This routine, called by the Command Analyzer and Executor when either a BUILTIN or PROCDEF command is recognized, maintains the procedure library and makes corresponding changes to the combined library. (See Chart CK.)

ENTRIES:

CZATP1 - entry point for either a BUILTIN or a PROCDEF command
CZATP2 - an internal symbol which is passed as the END entry during an LPCINIT call

MODULES CALLED:

Dictionary Handlers (CZASD):

STARTVAR (CZASD2)	Initializes a VM page in dictionary format.
RFR (CZASD3)	Finds a PROCDEF or BUILTIN entry.
NEXTRFR (CZASD4)	Adds and deletes entries from the dictionary at the direction of the procedure.
ENTR (CZASD5)	Adds an entry to the dictionary.

User Controller (CZAMZ):

LPCINIT (CZASW1)	Initializes the procedure as an LPC.
LPCEDIT (CZASW4)	Begins execution of Text Editor commands.

FINDDS (CZAEC1)	Finds or creates a JFCB.
DDEF (CZAEA4)	Creates JFCB if FINDDS fails to locate one for the input dsname.
CHEKDS (CZAAC2)	Validates the data set name.
User Prompter (CZATJ1)	Informs user of error situations.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: The principal activities are carried on at entry point CZATP1. After parameter validation, LPCINIT is called to activate the LPC for PROCDEF. On the first call the SYSPRD member is erased from USERLIB if one exists. If the data set to which the BUILTIN or PROCDEF is to be added is not USERLIB, a call to CZASD2 is made to initialize a page, obtained by GETMAIN, for use as a variable length dictionary. This dictionary will contain the SYSPRD member for the BUILTIN/PROCDEF. On each call, a line corresponding to the BUILTIN or PROCDEF line is built and written into SYSPRO as line zero of the corresponding region, and a corresponding entry is added to the combined dictionary.

Transactions are enabled and the routine to modify the dictionary is entered. The end entry for CZATP is CZATP2. At this entry CZATP issues a SETL to line zero of the current region to determine if it still exists. If not, the corresponding dictionary entry is removed. The transaction table is marked as processed. SYSPRO is closed to save a PROCDEF if one was created.

ERROR CONDITIONS: The routine checks for any name or symbol which is too long, and the situation where there is no name for PROCDEF or BUILTIN. If one of these errors or a System Fault occurs, this routine calls the User Prompter to print an appropriate message.

SYSTEM CONTROL BLOCK USAGE:

Control Dictionary Entry (CHADEN)
Control Dictionary Heading (CHADCT)
Data Set Control Block (CHADCB)
New Task Common (CHANTC)
Task Data Definition Table (CHATDT)
Transaction Table (CHATRN)

Procedure Expander Routines (CZATE)

The Procedure Expander is called by the Command Analyzer and Executor when it identifies a verb as a procedure name. The Procedure Expander consists of seven modules which:

- Locate the procedure in the Procedure Library (PROCLIB).
- Move the procedure into the Source List (SL).
- Construct (if parameters are present) a table of parameters, equivalent names, and calling values, with default values inserted where required.
- Substitute calling parameters in the text as the procedure is moved into the Source List. (See Chart CL.)

ENTRIES: Each of the seven modules is defined by one entry point:

CZATE1 - entry into main line processing
CZATE2 - BUILDLIST entry
CZATE3 - LISTEQ entry
CZATE4 - DEFSEARCH entry
CZATE5 - procedure parameter scan entry
CZATEL - entry into the procedure expander from routines that have been entered via macro
CZATE10 - subparameter search entry

Only CZATE1 and CZATE6 serve as entry points to the Procedure Expander from routines outside of it.

MODULES CALLED: The Procedure Expander, during its main line processing, calls three of the other six modules within itself:

BUILDLIST (CZATE2)	Builds an ELIST and ILIST from dummy parameters.
LISTEQ (CZATE3)	Builds a PLIST from the calling parameters.
DEFSEARCH (CZATE4)	Inserts default values in the PLIST if no calling parameters are present. Also called by LISTEQ.

In addition, the BUILDLIST routine and the LISTEQ routine call the fifth Procedure Expander routine (CZATE5) to find and isolate each procedure parameter in turn and put it into the Source List.

When entered at CZATE6, Procedure Expander processing includes a call only to LISTEQ. The subparameter processing routine, CZATE10, is a logical extension of LISTEQ and is called only by it.

The modules (outside themselves) upon which the Procedure Expander routines rely are:

Source List Handlers:	
Buffer Fetch (CZASC2)	Extends the size of the source list.
Update (CZASC6)	Updates the source list pointers.
Dictionary Handlers:	
RFR (CZASD3)	Searches dictionary for parameter names.
GDV (CZASDX)	Determines a default value.
Verb Scanner (CZASA2)	Isolates the verb for examination.
User Prompter (CZATJ1)	Writes messages to the user.

The Procedure Expander routines also use OPEN, FIND, READ, GET, ESETL, and HASH.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: Upon entry to the Procedure Expander, a search for the procedure in the PROCLIB is initiated via the READ macro instruction, after assuring that the USERLIB DCB is open (if the procedure definition -- PROCDEF -- resides in USERLIB. If the procedure does not exist, an immediate return is made to the calling routine with a return code that indicates this fact. If the search was successful, a test is made to determine if there is enough space in the source list for the procedure's first line. If not, a Source List Handler is called to expand the SL by another page. In either event, the current end point (EP) value is saved for a later test, and a P marker is placed in the SL at the saved EP address. The current start address (SA) value, which points to a position past the identified name in the SL, is inserted into the P marker.

The procedure's first line is moved into the SL after the P marker, via the GET macro instruction. If the procedure is null, the SL is left in its original state, and control returns to the calling routine. Otherwise, the EP, SA, and available byte count are updated in the SL, and the SA value is saved. Also, the IDENT field of this line is saved in order to determine the end of this procedure. The Procedure Expander now calls the Verb Scanner to determine if the first statement is PARAM.

The lack of a PARAM statement results in moving the procedure into the SL one line at a time. Initially, an E marker, which contains a pointer back to the P marker, is placed at the end of the first line. The EP and available byte count are reset, and the value of EP is saved. A test is made to determine if there is enough room in the SL for the next line. If not, the SL is extended another page, via the Source List Handler routine. In any event, GET is used to obtain the next line, and move it into the SL. If an EODAD occurs, or the IDENT field is found not to match the initial IDENT value, the process is complete, and an exit is made after resetting the SA to the first byte after the P marker. Otherwise, an E marker, with a pointer set to the value of the previous line's E marker pointer, is placed at the end of the current line. The previous E marker pointer is reset to the previously saved EP value; that is, the address of the new line. The EP and available byte count are updated after saving the current EP, and the process is repeated. The last line, whose E marker points back to the P, or beginning marker, is an exception. If during the process, the source list must be expanded, the last E marker on the page will point to the input area on the new page.

If the first line is a PARAM statement, the end result is the same; however, the expansion process is different. Initially, a test for a line continuation marker is made. If one exists, the next line is obtained via the GET macro instruction, and stored in the SL after the first line's E marker (current EP). A new E marker is placed after the continuation line. The pointer in the previous E marker is placed in the current E marker; then the previous E marker is changed to point to the next line. The EP and available byte count are updated, and the process is repeated until all continuation lines of the PARAM statement reside in the SL. The last EP is saved for future linkage of succeeding procedure lines. Due to the previous Verb Scan call, the SA is currently set to permit an immediate dummy parameter scan.

BUILDLIST Routine: To analyze the procedure's dummy parameters, control passes from the main line routine to the BUILDLIST routine (CZATE2). BUILDLIST calls the Procedure Parameter Scan routine (CZATE5) to:

- Isolate (search for a delimiter) the parameter beginning at the current start address (SA) of the source list (SL).

- Store the parameter, preceded by a one-byte length characteristic, in the SL at the current end point (EP).
- Update the SA, EP, and available byte count in the SL accordingly.
- Determine whether to strip the parameter's quotation marks before storing them in the SL.
- Return the address of the parameter's new SL location; return a signal identifying it as a normal or special (quoted) string, and a return code (RC) designating its delimiter (= or ,).

On return, BUILDLIST scans and identifies the parameters as to type, constructing a TYPE table in which each parameter is designated as normal (N) or special (S), where the special form is a quoted string.

In addition to scanning and identifying the parameters, BUILDLIST constructs two pointer tables, the ELIST and the ILIST.

ELIST Contains the address of each external parameter name.

ILIST Contains the address of each name represented in the keyword form.

Of course, the ILIST may not have all its entries filled. BUILDLIST now returns to the main line processor.

The main line Procedure Expander now fills vacant ILIST entries with the corresponding ELIST values, to obtain a final table with pointers to parameters which will be used in substitution procedures. An additional table containing the first letter of each parameter in the ILIST is constructed. This table will be used to make tests for the possible existence of a dummy parameter in the procedure text. The EP is reset to EP_2 in order to provide for an extension of the SL with calling parameters.

The Procedure Expander now tests for the presence of calling parameters. If there are any calling parameters, the processor links to the LISTEQ routine (CZATE3) to build a table of pointers to the calling parameters (PLIST); if not, the processor links to the DEFSEARCH routine (CZATE4) to build the PLIST from default values.

LISTEQ Routine: After first clearing the PLIST, LISTEQ calls the Procedure Parameter Scan routine (CZATE5) to isolate a parameter (as it did for BUILDLIST, above). If, on return, LISTEQ finds the parameter null it links back to CZATE5. This process continues until a non-null fragment is found or the end of the input string is reached.

If the parameter is not null, the delimiter becomes important in determining the type of keyword at hand. If the delimiter is an equals sign, LISTEQ clears the keyword list (ELIST) index, and begins searching for a match between the input parameter and the ELIST entries. Errors exist if (1) the ELIST is exhausted before a match is found, or (2) an asterisk precedes the match in the ELIST. Errors are marked by flag settings and may result in the user being prompted.

If a match is found, the ELIST entry is checked further for length, subparameters, and indefinite form. Flags are set representing the conditions found. In all cases, error and non-error, LISTEQ ends its search loop by returning to the point in its logic where it calls CZATE5 to isolate the next parameter.

When the delimiter is not an equals sign, LISTEQ makes different tests. If the input value is the mate of an invalid keyword, it sets a flag, and (barring end of input string) loops back for another call to CZATE5. If the input is the mate of an asterisk keyword, it is invalid; the CZATE5 loop continues. The parameter fragment may have been preceded by an equals sign. If it is not, LISTEQ begins searching for an asterisk keyword that matches the input parameter. If one is found, its location in the PLIST is noted and the CZATE5 loop rejoined. If the ELIST entries are exhausted before a match, but the keyword positions are not exhausted, or if the parameter was preceded by an equals sign, further tests are made for indefinite form. Occurrence of a parameter in the indefinite form results in the creation of an indefinite form parameter sublist. Then, if the parameter string is not exhausted, LISTEQ calls CZATE5 again and continues processing.

When the parameter string is exhausted, LISTEQ may, if necessary, call DEFSEARCH to seek synonyms, and will, on return, link to the Subparameter Search routine (CZATE10) to seek subparameters.

Subparameter Search: This routine is a logical extension of LISTEQ. It uses LISTEQ's recursion index to orient itself. This routine loops through the ELIST/PLIST index, locating, identifying, and flagging subparameters, if they exist, and creating a subparameter list for analysis by LISTEQ. When the keywords are exhausted, this routine finally exits to LISTEQ.

When the Subparameter Search routine returns to the LISTEQ routine (after all parameters are exhausted), LISTEQ sets its return code and returns to its calling routine. This may be the main line processor or it may be that part of the Procedure Expander defined by entry point CZATE6. In creating the PLIST, LISTEQ has ensured that the procedure's calling parameters and subparameters have a one-to-one correspondence with the entries of a desired BUILTIN or textual procedure dummy table.

DEFSEARCH Routine: This routine is used to determine if default values exist for dummy parameters without corresponding call values, and, if they do, to provide default substitutions. DEFSEARCH tests each entry in the calling parameter pointer table (PLIST), and, if it is zero, identifies the values through the corresponding entry in the dummy parameter pointer table (ELIST). This value is passed to the Dictionary Handler routine (CZASD3), which searches for the default value.

Such a default value, if found, is moved (preceded by its length characteristic) by DEFSEARCH into the Source List (SL) at the current end point (EP), and a pointer to the value is placed in the PLIST. The EP and available byte count (ABC) in the SL are updated to reflect the length of the default value. When PLIST entries have been tested (whether or not default values are found), DEFSEARCH returns to this calling routine. This can be either LISTEQ or the main line processor.

Mainline Processing Continues: The Procedure Expander now moves the entire procedure from the PROCLIB into the SL, one character at a time. Each character is initially tested to determine if it is an end of line (EOL). If not, it is then compared with the first character of each dummy parameter in the ILIST (previously constructed first character table). If no match occurs, the character is stored in the SL; the EP and available byte count are updated, the PROCLIB character pointer is advanced and control seeks to obtain the next procedure character. Had a first character match occurred, the entire dummy parameter (obtained via a pointer in the ILIST) would be compared with its corresponding number of characters in the PROCLIB. If a match does not occur, the logic for no match on the first character is followed. If a match did occur, this indicates a string substitution and the available byte count is tested to determine if there is enough room in the SL for the string (length characteristic is retrieved from the SL preceding the calling

parameter to be substituted). If required, the SL is extended by another page, via the GETBUFSL routine. In any event, the corresponding calling parameter is obtained, via the pointer in PLIST, and inserted into its new position in the SL. The pointer to the PROCLIB text is updated to the next character after the dummy parameter, the EP, and available byte count are updated and control returns to interrogate the next PROC character.

As previously mentioned, while the above logic is used to move the procedure into the SL, each character is tested for an EOL. If an EOL is encountered, an E marker is placed in the SL at the current EP, and the previously saved EP is tested to determine if it points at the P marker. This test will be true only for the first procedure line after the PARAM statement, and its E marker will contain a pointer back to the P marker. The new EP is saved to allow for the entry of additional lines; the previous pointer will assume the saved EP value. This will result in a string of lines with markers pointing to their successors with the exception of the last line which will point to the P marker. After each line is properly entered and connected in the SL, a test is made to determine if a EODAD or IDENT not-equal condition has occurred (the comparison is made for the previously saved IDENT value). If not, control passes back to process the first character in the next line; otherwise, the process is complete, and the SA is updated to point back to the first executable procedure statement; control returns to the calling routine.

Entry at CZATE6: The Procedure Expander, at this entry point, accepts parameter analysis requests from routines which have been entered via macros. Three parameters are passed to it: the address of the BPKDS, the address of the string to be scanned, and a flag governing dictionary reference. Upon entry, this routine sets a flag indicating that no PRMPT macros should be issued. If the BPKDS and string prove valid, they are used to form a parameter list for LISTEQ. This routine then calls LISTEQ. On return it exits, passing the return code from LISTEQ to its calling routine. This routine provides its own return codes if the BPKDS is invalid (X'C'), the string is invalid (X'8'), or the dictionary search parameter is invalid (X'10'). In these three cases, this routine does not call LISTEQ.

ERROR CONDITIONS: If the main line processor discovers that the requested procedure does not exist in the PROCLIB, a return code designating this fact is sent immediately to the calling routine. The Procedure Expander calls the User Prompter (CZATJ1) to send a diagnostic message to the user if:

1. Parameter line is erroneous.

2. More calling parameters than dummy parameters when the PREXPAND flag is not set to 'Y'.

3. A synonym loop occurs while BUILDLIST is scanning the parameter line values.

In these cases the command is canceled. In addition, a message is sent to the user and processing continues, under the following conditions:

1. BUILDLIST finds that a SYN/DEFAULT parameter on the left side of an equals sign is a quoted string. The parameter will be treated as an unquoted string.

2. BUILDLIST discovers that, in a string with more than 2 parameters, all parameters are equal. The last parameter has precedence in the ILIST.

3. BUILDLIST or LISTEQ discovers that the maximum number of parameters allowed in a list has been exceeded. Any excess parameters are lost.

4. LISTEQ, while constructing the PLIST, notes redefinition of any parameter. The new definition has precedence.

5. LISTEQ discovers that a calling parameter keyword does not exist in the ELIST. The parameter is deleted from the parameter list.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Control Dictionary Entry (CHADEN)
Sublist Header (CHASLH)
Source List Marker (CHASLM)
Source List Page Header (CHASLP)
New Task Common (CHANTC)
Profile Character and Switch Table (CHAPCT)

QUIT Command Routine (CZAFL)

This routine processes the QUIT Command, which may be issued only by a system manager or administrator. QUIT removes the specified user from the system, and erases, reassigns, or stores (on a private volume) his data sets. In nonconversational mode, all the user's data sets are erased. (See Chart CM.)

ENTRY: CZAFL1 - normal entry

MODULES CALLED: During its processing, QUIT calls some of these modules several times.

NEXTPAR (CZAAC1)	Locates and validates operands and delimiters.
ALFNUM (CZAAC2)	Validates userid.
DDEF (CZAEA5)	Defines JFCBs for data sets to be manipulated.
FINDJFCB (CZAEB1)	Locates JFCBs for SYSUCAT, etc.
FINDDS (CZAEC1)	Locates JFCB for a data set, for example, a JFCB that DDEF has just created for a data set to be recataloged.
LOCATE (CZCFL)	Finds all the user's data sets.
CATALOG (CZAEI3)	Recatalogs a data set.
ERASE (CZAEJ6)	Erases a data set.
Vam Tapes (CZAET6)	VVs a data set from public storage onto a private volume.
RELEASE (CZAFJ3)	Releases the JFCB of a data set QUIT has worked with.
MSGWR (CZAAD2)	Prompts the user for additional or missing information; prints error messages.
User Prompter (CZATJ1)	Prints error messages.
DELVAM (CZCFT1)	Deletes catalog entries, as well as the user entry from the User Table.

DELREC (CZCPH1) Deletes SYSSVCT entry.

The QUIT command processor also uses OPEN, READ, WRITE, TSEND, RELEX, STOW and FREEMAIN during its operation.

EXITS: The QUIT routine normally returns to its calling routine, via the RETURN macro instruction.

OPERATION: The routine first validates the user identification (userid) supplied as the command operand. If the command was issued by a system administrator, the first two characters of the userid are checked to ensure that they match the initial two characters of the administrator's userid; only the administrator who joined the user can quit him.

The User Table is now opened, and the entry corresponding to the userid operand is read. QUIT sets an indicator in the entry so that the user cannot initiate any new tasks, then checks the entry to see if the user has any currently active tasks. If he does have an active task, a message is issued to the administrator (or manager), asking if he wants to cancel the QUIT command -- by pressing his ATTENTION key -- or wait until the user's tasks are completed. If he chooses to wait, a TSEND SVC is issued to end the time slice. When control returns to QUIT, the user entry is again tested to see if the user has an active task. Processing of the QUIT command continues when the user's tasks have been completed.

LOCATE is called to obtain a list of the user's data sets, and the administrator (or manager) is asked if he wants to erase all of those data sets or dispose of them individually. If he chooses to erase them, all data sets owned by the user and residing on direct access volumes will be eliminated through successive calls to ERASE. Other data sets are ignored.

If individual disposition is selected, QUIT presents the data set names one by one, and the administrator indicates his choice for each: erase, recatalog, or copy onto a private direct access volume.

An erase request is processed by a call to ERASE, provided the data set is owned by the user and resides on direct access storage. Otherwise, the erase request is ignored.

If the data set is to be assigned to another user's catalog, QUIT fetches and validates the userid of that other user and the new name under which the data set will be cataloged. DDEF is then called to create a JFCB; FINDDS locates this JFCB; and CATALOG then catalogs the data set, under the new name, in the specified user's catalog. The data set is closed through a call to RELEASE.

For a copy request, QUIT fetches and validates the identification of the volume upon which the copy is to be written, the name to be assigned to that copy data set, and the type of volume (2311/2314). DDEF and FINDDS are then called to create and locate JFCBs for the old data set and the copy, VV is called to make the copy, and RELEASE closes the copy. If the old data set is on direct access storage, it is eliminated by a call to ERASE; otherwise, it is closed by RELEASE. All copied data sets are given to TSS*****.

When disposition of the user's data sets is completed, the userid is eliminated from the catalog -- thus eliminating that user's catalog; the user's entry is deleted from the User Table, and the table is closed; and, finally, the work area obtained by LOCFQN is released. Control then returns to the calling routine.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the administrator (or manager), through MSGWR, of any errors found.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Catalog TBLOCK (CHATBD)
Task Common (CHATCM)
System Common (CHASCM)
Task Data Definition Table (CHATDT)
User Table (CHAUSE)
Active User Table (CHAAUL)
Interrupt Storage Area (CHAISA)

RELEASE Command Routine (CZAFJ)

This routine deletes job file control blocks (JFCB) from the task data definition table. It may be used to release the devices associated with a data set, to deconcatenate one or all data sets of a given concatenation, and to remove a job library from the program library list. (See Chart CN.)

ENTRIES:

CZAFJ1 - normal command entry
CZAFJ2 - nonprivileged macro instruction entry
CZAFJ3 - privileged entry
CZAFJ4 - entry for LOGOFF command routine
CZAFJ5 - second entry for ERASE/DELETE command routine
CZAFJ6 - first entry for ERASE/DELETE command routine (used to close data control blocks only, not to delete JFCBs)

MODULES CALLED:

NEXTPAR (CZAAC1)	Gets and validates input operands.
CHEKDS (CZAAC2)	Validates data set name.
ALFNUM (CZAAC3)	Validates data definition name.
FINDDS (CZAEC1)	Finds JFCB for a given data set name.
MSGWR (CZAAD2)	Prompts user for additional or missing information; prints error messages.
RELEAS (CZCAD1)	Releases the devices associated with a data set.
LOCATE (CZCFL1)	Sets up a fully qualified name for a generation data group name.
Loader Release (CZCCD2)	To UNLOAD modules loaded from job libraries.
User Prompter (CZATJ1)	Prompts user for additional or missing information; prints error messages.

The RELEASE Command routine also uses VSEND, FREEMAIN, CLOSE, and DUPCLOSE.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: This routine first determines if the drive is to be released and sets a flag accordingly. Then, the routine determines if its input, fetched via NEXTPAR, is *ALL or a data definition name (ddname). The *ALL requests that all JFCBs in the Task Data Definition Table (TDT) be released. If the *ALL option has been specified, RELEASE locates the

first JFCB to be released, tests for a SCRATCH or HOLD option, and honors it if specified. Then the routine searches the entire TDT. As each JFCB is found, RELEASE tests its ddname to see if the JFCB is reserved; reserved JFCBs have ddnames that begin with $$$ or SYS, can be released only if the calling routine is privileged.

If the JFCB is not reserved or if the calling routine is privileged, RELEASE then closes all data control blocks associated with the JFCB. The request queue is scanned to determine if the symbolic device address or the device type code entries are matched in the JFCB. Unless entry was made from ERASE/DELETE at CZAFJ6, an included subroutine -- DJFCB -- is now called to perform the deletion. RELEASE continues this process until the entire TDT has been scanned. RELEASE searches the entire SDAT table to insure that all devices assigned to the task are freed. Each SDAT entry which contains the task's taskid is deleted by RELEAS3, an included subroutine. Control is returned to the calling routine.

When the *ALL option is not specified and a ddname is given, the routine validates it, via ALFNUM, locates that ddname in all chains, and then looks for a data set name (dsname). If the dsname is defaulted, the JFCB for the specified ddname is located in the TDT and checked to see if a concatenation is involved. If no concatenation is involved and the data set is a library, the JFCB is checked to determine if any modules were loaded from it. If not, processing continues. If so, Loader Release (CZCCD2) is called to unload all modules. If Loader Release is unable to unload all modules after being called by the Release command, the user is prompted. Under all conditions, a return code of X'24' is returned to the caller. Then the request queue is scanned for a match on symbolic device address of device type code. If the data set is open, the DCBs associated with the data set's JFCB are closed and the included subroutine DJFCB is called to delete the JFCB. If concatenation is indicated, all JFCBs in the concatenated chain will be deleted after their DCBs have been closed by the CDCB subroutine.

A dsname supplied along with the ddname indicates that deconcatenation of one data set is desired. In this case, RELEASE validates the data set name via NEXTPAR and CHEKDS, and uses FINDDS to locate the JFCB associated with that name. The JFCB will then be deleted (by DJFCB) from the concatenation, but the rest of the concatenation will remain intact.

DJFCB does the actual deletion, handling one JFCB each time it is called. DJFCB will delete the JFCB from every chain in which it is included (concatenation, library, temporary tabulation, as well as the main TDT chain) and link its area to the free area chain. If the JFCB to be deleted is on a private volume, DJFCB first validates the volume field. For multivolumes, DJFCB finds the end of the chain. If the volume field is invalid, the device is released, but no PAT page is freed. Through the RELEASE service routine, DJFCB also releases the devices associated with the JFCB. Note, however, that it will not release devices for a public, uncataloged data set until ERASE/DELETE has made the request.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code will be returned in register 15:

Code	Significance
00	Normal return
04	Ddname not given
08	Attention interrupt
0C	Reserved ddname
10	Unknown ddname
14	Uncataloged data set on public volume
18	Unknown dsname

20 Invalid operand
24 All modules could not be unloaded

The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For the messages issued, see System Messages.

The routine may issue these system errors. In each case the severity is minor, and the ABEND message is SYSTEM ERROR - TASK TERMINATED.

SYSER Code	Significance
050505901	Invalid information from NEXTPAR, ALFNUM, or CHEKDS.
050505902	Invalid information from MSGWR.
050505903	Invalid information from FINDDS.
050505904	Invalid return from RELEASE.
050505905	Invalid information from LOCATE.
050505906	Pointer to RESTBL (for VAM) or DEB (for SAM) data sets said to be open; ddname taken from TDT, but not subsequently found. Error in TDT.
050505907	DCB header missing.

SYSTEM CONTROL BLOCK USAGE:

Catalog SBLOCK (CHACCC)
Data Extent Block (CHADEB)
Task Common (CHATCM)
Data Control Block Header (CHADHD)
RESTBL Header (CHARHD)
Request Queue (CHARQU)
Symbolic Device Allocation Table (CHASDA)
Task Data Definition Table (CHATDT)

RET Command Routine (CZAEN)

This routine makes it possible for the user to modify the mode field in the Data Set Descriptor (DSD) and the equivalent Job File Control Block (JFCB) field which contains the storage type, deletion and owner/user access attributes of the data set. (See Chart CO.)

ENTRIES:

CZAEN1 - command entry
CZAEN2 - macro entry

MODULES CALLED:

DSCB RD/WR (CZCEM)	Pages in DSCBs.
SRCHSDST (CZCQE)	Gets count of current data set users.
FINDDS (CZAEC1)	Finds JFCB for a given dsname.
ADDCAT (CZCFA2)	Updates the catalog DSD.
User Prompter (CZATJ1)	Sends prompting and diagnostic messages to the user.

To update the user accounting information, RET updates RCR UPDATE, RCR RATION, and RCR VACATE.

EXITS: The routine returns to the calling routine via the RETURN macro instruction.

OPERATION: The RET command routine is entered with a pointer to a parameter string containing pointers to the data set name and the values to be entered into the mode field. When RET has finished processing, the new values will be reflected in both the DSD and the JFCB for the data set.

RET processes the input parameter DSNAME, then calls FINDDS to locate the JFCB or, if necessary, create one.

A check is made that no more than one DCB is open for this JFCB. If a DCB is open, a check is also made to determine whether the DCB belongs to the same user who issued the RET command. If it is not the same user, RET issues a diagnostic, via PRMPT, and returns to the caller. If it is the same user, a check is made to determine if the data set is shared.

For shared data sets, RET checks the owner/user access and permits sharers to use the RET option only if their access privilege is unlimited. RET calls SRCHSDST to determine if there is any current user of this data set and sends a diagnostic, via PRMPT, if there is a current user.

RET then scans and validates the input values, making only those changes to the JFCB that the user has specified. Unspecified values remain unchanged. If invalid values are entered, the routine issues a diagnostic and returns to the caller.

RET calls ADDCAT to update the DSD mode field entry. If a public VAM data set is changed from a permanent to a temporary data set (or vice versa), RET calls RCR UPDATE, RCR RATION, and RCR VACATE to update the User Table and either links the JFCB into the temporary tabulation chain or removes it, if necessary.

If invalid parameters are entered in conversational mode, RET issues a diagnostic via PRMPT, and the command is canceled. In nonconversational mode, a message is written on SYSOUT and control is returned to the user.

ERROR CONDITIONS: When RET is entered through a macro call, error codes are returned to the calling program in register 15:

Code	Significance
00	Normal return
04	Data set not cataloged
08	Invalid dsname
10	Incorrect values (code given)
14	Open DCB
1C	Data set not found or not available to this user
20	Insufficient resources
24	Non zero return code from ADDCAT

28 Invalid temporary tabulation chain in JFCB

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
User Table (CHAUSE)
Catalog SBLOCK (CHACCC)
Task Data Definition Table (CHATDT)
Format-E DSCB (CHADSE)
Active User Limits Table (CHAAUL)
System Common (CHASCM)
Public Volume Table (CHAPVT)

RPS/CVV/LPDS/CPS Command Routine (CZAXX)

This routine provides the means for obtaining information about the status of public storage (LPDS); deleting certain invalid data sets on public storage (CPS); and recreating public storage when the catalog and/or public storage no longer provides serviceability to support the Time Sharing System (RPS and CVV). (See Chart CP.)

RPS: The RPS command is processed in three phases:

1. Phase 1 recovers multivolume data sets from public storage (without the use of the catalog) and copies them to tape. The RVN0 (ACV volume) used in the search for multivolume data sets can be either the ACV in use by the system or an old ACV that has been mounted as a private pack.

2. Phase 2A copies data sets from private volumes (usually those previously mounted as public) into currently mounted public storage (which, by definition, catalogs them).

3. Phase 2B copies output produced by Phase 1 or created by VAM TAPE (CZAET) into public storage.

CVV: The CVV (Catalog Vam Volume) command catalogs data sets on public volumes from the previous system. Old public volumes may be mounted during Startup, provided consistency is maintained in relative volume numbers.

LPDS: The LPDS (List Public Data Sets) command searches public storage and lists each data set found. Any data set that does not pass certain edit tests is flagged on the list as being invalid. All others are marked valid.

CPS: The CPS (Clean Public Storage) command searches public storage the same as LPDS, checks each data set the same as LPDS, and then deletes the data set if it fails any of the edits. The output listing flags each data set as "retained" or "erased."

ENTRIES:

CZAXX1 - RPS command entry
CZAXX11 - RPS BPKD
CZAXX2 - CVV command entry
CZAXX12 - CVV BPKD
CZAXX4 - SYNAD for output list data set
CZAXX6 - EODAD for RPS Phase 2B tape input
CZAXX7 - SYNAD for RPS Phase 2B tape input
CZAXX3 - CPS command entry
CZAXX13 - CPS BPKD
CZAXX8 - LPDS command entry
CZAXX14 - LPDS BPKD

RESTART: All commands have a START parameter which is used to indicate where a restart is to take place. It can be specified as 'CONT' which will process from the point of last interruption; or it can be specified as a specific DSCB or FSQ number.

MODULES CALLED:

MTREQ (CZCAA1)	Issues mount requests for volumes as required.
VAMINIT (CZCEQ1)	Reads in Page Assignment Table (PAT).
OBTAIN (CZCFO1)	Reads volume labels and DSCBs.
PAIR (CZACS1)	Puts entry in AIR table.
LOCATE (CZCFL1)	Determines whether data set is already cataloged.
ERASE (CZAEJ6)	Erases a new data set if the data set is not indexed or renamed successfully.
DDEF (CZAEA5)	Creates input and output JFCBs and reDDEFs TSS*****. USERLIB if it was released during processing.
VV (CZAET6)	Copies data set from private to public volume.
VT (CZAET4)	Copies data set to tape.
TV (CZAET5)	Copies data set from tape into public storage.
INDEX (CZCFI1)	Creates generation index.
ADDCAT (CZCFA1)	Catalogs data set.
RELEASE (CZAFJ3)	Releases input and output JFCBs and USERLIB JFCB.
CATALOG (CZAEI2)	Renames a new data set to original name, replacing generated name.
DELCAT (CZCFD1)	Deletes catalog entries when CVV deletes a cataloged data set.
DELVAM (CZCFT1)	Deletes DSCBs and pages of processed (RPS Phase 1) and error (CVV) data sets without accessing the Catalog.
GATWR (CZATC1)	Issues status message after attention interrupt or ABEND.
MSGWR (CZAAD)	Issues messages on SYSOUT.
GETMAIN (CZCGA2)	Reserves virtual storage.
FREEMAIN (CZCGA3)	Releases virtual storage.
SETXP (CEAH7)	Makes DSCB pages available to a task.
PGOUT (CZAA1)	Returns updated DSCB pages to disk.
PAIR (CZACS2)	Removes AIR table entry.

EXITS: This routine always exits to the calling routine via a RETURN macro instruction.

OPERATION: Initialization procedures are similar for all commands and phases. All parameters are checked for validity and proper combinations. All functions call PAIR (CZACS1) to place an entry in the ABEND Interlock Recovery table (CHAAIR). All functions locate a JFCB for an RPSOUT, CVVOUT, CPSOUT or LPDSOUT data set; checking for VI data set organization; opening the data set and writing the header record. Exceptions in initialization procedures are as follows:

1. Initialization for RPS Phase 1 with the ACV parameters includes creating a new PVT to be used by the system when copying multivolume data sets to tape. This PVT will contain the ACV volume that was requested by RPS. A call is made to MTREQ in Phase 1 with the ACV parameter to mount and initialize the old ACV volume.

2. Initialization for Phase 2A includes calls to MTREQ to mount volume, VAMINIT to read in the PAT, and OBTAIN to read the volume label.

3. Initialization for RPS Phase 1, RPS Phase 2A, CVV, CPS and LPDS includes scanning the PAT(s) for DSCB pages and issuing the SETXP macro to make DSCB pages available to the task.

4. Initialization for all commands is effected by the START parameter. For all commands and phases except RPS 2B, the starting point of the search is initialized to the Volume, DSCB page and slot; one beyond the point that an interruption took place (this information is saved in the CSECT). This occurs if START = CONT is specified and the CSECT does contain information from the last run. If the CSECT does not contain information from the last run and START = CONT is specified, processing begins at the beginning of the volume specified. The starting point is also modified if a specific DSCB is specified in the START parameter. The initialization is similar for RPS Phase 2B except that File Sequence Number (FSQ) is used.

Recreate Public Storage (RPS)

Phase 1 locates a Format-E DSCB. All DSCBs for a data set are validated before they are processed. If the Format-E DSCB is valid, the data set name, DSCB slot address, data set organization, reference data, number of data pages, overflow pages, directory pages, total pages assigned and volume serial number are stored in the RPSOUT buffer. If invalid conditions are detected, the data set may be skipped.

The relative volume number fields in the Format-E DSCB entents (and in any Format-F entents) are checked to determine whether the data set is multivolume. Single volume data sets are skipped. (Since the system may have been started up with a fresh ACV in order to reclaim any multivolume data sets, the DSCB chain may point to a blank DSCB on the ACV. These data sets are skipped. There is no identification for data pages on the ACV which are referenced by DSCBs on other packs.) If the ACV parameter was specified, this problem is minimized. The RVN0 of public storage that is used is taken from a specially built PVT created by CZAXX and used by VT when multivolume data sets are to be copied to tape.

If there is no tape JFCB, it is created and set for multivolume output. Task Common is updated with the userid of the data set. The input (disk) JFCB is created, if necessary, and updated. VT copies the data set onto tape. Input JFCB fields are cleared to disconnect the data set from the system.

Phase 2A locates and validates a Format-E DSCB and stores DSCB information (data set name, DSCB slot address, data set organization, reference date, number of data pages, overflow pages, directory pages, total pages assigned and volume serial number) in the RPSOUT buffer. A

test is made for the SYSCAT or SYSOPER0 data sets and the data page count and total pages assigned are checked for a null data set. If any of these conditions are present, the RPSOUT record is updated with the IGNORED disposition and the record is written. The next DSCB is then processed.

Phase 2A Format-E DSCB processing continues by checking every extent and DSCB pointer against the relative volume number (RVN) which was read in from the volume label. This is done to guard against processing multivolume data sets. The RPS Flag in the JFCB is set to 'C0' to indicate to OPENVAM that it should disregard the RVO of each entry in the DSCB. Task Common is updated with the userid of the data set. The LOCATE is called with the data set name from the DSCB to determine whether or not the data set is already cataloged. If the data set is cataloged and not a USERLIB, it is ignored and the RPSOUT record is updated with the IGNORED disposition. If the data set is a USERLIB, the SYSULIB JFCB is released and ERASE is called to remove the USERLIB from public storage. (The USERLIB from the private volume replaces the USERLIB in public storage which should be a null data set.) Otherwise, the TDT chain is searched for the current data set name, and if a JFCB is found with the same name, it is RELEASEd to prevent later interference. If a JFCB does not exist for the input data set, DDEF is called to create an input JFCB. If a JFCB does exist, the data set name fields in the JFCB are updated with the current data set name. All the necessary input JFCB fields are filled in from known data and the TDTRPS field is set to X'80' as a signal to other modules to pick up information from the JFCB and avoid the Catalog. An output JFCB is created or updated, and fields are cleared so that they may be filled by other modules. TDTRPS remains X'00' in the output JFCB. The output data set name is USERID.RPSNNNNN, where NNNNN is a number generated to create a unique dsname. VV is called to copy the data set. Once the copy is created, the data set must be renamed according to its original name.

A check is made on the last qualifier of the original name to determine whether or not the data set being processed is a generation data group. If it is a generation data group, the SBLOCK returned by LOCATE is checked to see if a generation index exists. If a generation index does not exist, one is created by a call to INDEX (maximum of 2 generations for LIST data sets; 5 for all other). CATALOG is then called with the rename option to rename the data set in the Catalog and the DSCB on public storage according to their original name. If at any point after the copy is made, processing is not concluded normally, the data set copy is ERASEd and the RPSOUT record is updated with the IGNORD disposition. The volume serial number of the public volume on which the correctly completed copy resides is placed in the RPSOUT buffer, and the RPSOUT record is updated with the PROCESSED disposition.

Phase 2B begins by checking to see if a tape has been mounted either by the user's DDEF or a previous RPS cycle. If no tape has been mounted, DDEF is called to mount it. The tape DCB is updated to link to the JFCB and the DCB is opened, moving the tape to the next data set if the previous one was skipped. GETMAIN is called to reserve a buffer page if one is not available and the first record of the data set is read. (The first record is the VAM TAPE control record.) Task Common is updated with the userid of the data set and the RPSOUT record is filled in. LOCATE is called to determine whether the data set is in the Catalog. If the data set is cataloged and is a USERLIB, the existing USERLIB is erased. If the data set is cataloged and not a USERLIB, the data set is skipped. If the data set is not cataloged, the tape is closed with the REREAD option which returns the tape to the beginning of the file. The output JFCB is created or updated. A check is made on the last qualifier of the original name to determine whether or not the data set being processed is a generation data group. If it is a generation data group, the SBLOCK returned by LOCATE is checked to see if a generation index exists. If a generation index does not exist, one is

created by a call to INDEX (maximum of 2 generations for LIST data sets; 5 for all other). TV copies the data set from tape to public storage, cataloging it directly under its correct name. The RPSOUT buffer is completed, and the record is written. At the end of the tape, FREEMAIN releases the buffer page.

Termination for both RPS, CVV, CPS and LPDS are handled by a single routine. PAIR (CZACS2) is called to remove the AIR table entry. The RPSOUT/CVVOUT data set is closed. JFCBs are released and the original userid is restored in Task Common. If SYSULIB has been released, TSS*** **.USERLIB is reDDEFed; the DCBs for its members are reopened and FIND macros are issued for the members. RELEASE is called to release the device for RPS Phase 2A or RPS with ACV parameter. Completion messages are sent to SYSOUT via MSGWR.

Catalog VAM Volume (CVV)

Once a Format-E DSCB is located, the DSCB information (the data set name, DSCB slot address, data set organization, reference date, number of data pages, overflow pages, directory pages, total pages assigned and volume serial number) is placed in the buffer to be written to the CVVOUT data set. If the data set is null or invalid, it is marked for subsequent deletion, unless it is the CVVOUT data set (which appears to be null if it is new). The CVVOUT data set is marked IGNORED and the record is written to it. LOCATE is called to determine whether or not the data set is already cataloged.

If a data set is cataloged but its DSCB pointer is pointing to another DSCB, the data set is marked for deletion. Cataloged data sets marked for deletion are deleted from the Catalog by DELCAT. Current system data sets which may be open are ignored even if marked for deletion. Other cataloged data sets are processed by updating the CVVOUT buffer with the IGNORED disposition and then writing the record. If a SYSOPER0 data set is not cataloged, the data set is marked for deletion. Other cataloged data sets are processed by creating a JFCB for the data set or updating the JFCB with the dsname. If the deletion flag is on, DELVAM is called with TDTRPS set to X'80' to delete the data set pages and DSCBs. Otherwise, a check is made on the last qualifier of the name to determine whether or not the data set being processed is a generation data group. If it is a generation data group, the SBLOCK returned by LOCATE is examined for an existing generation index. If a generation index does not exist, one is created by a call to INDEX (maximum of 2 generations for LIST data sets; 5 for all others.) ADDCAT then creates the Catalog entry for the data set. If the data set is cataloged properly, the CVVOUT buffer is updated with the CATALOGED disposition and the record is written. If the data set is not cataloged correctly, the CVVOUT record is updated with the IGNORED disposition and the record is written. Termination procedures are described in the last paragraph of the preceding section.

List Public Storage (LPDS)

Processing for LPDS is similar to CVV in that each data set is examined to determine its validity (passes certain edit tests listed below). An indication is then made in the LPDSOUT data set to indicate the validity of the data set, and if it is invalid, the reason.

Clean Public Storage (CPS)

Processing for CPS is similar to LPDS in that each data set is examined to determine its validity. When a data set is found to be invalid, it is erased from Public Storage and deleted from the catalog.

ERROR CONDITIONS: RPS/CVV/CPS/LPDS issue the following error messages on SYSOUT via MSGWR:

B403 - Memory allocation request rejected. Command ignored.
EE04 - Data on tape invalid. Command canceled.
EE04 - Return from Vam Tape invalid. Command canceled.
EE04 - Parameter passed to Vam Tape invalid. Command canceled.
EE0D - Volume for (dsname) not 9 track tape. Command canceled.
EE0F - Unrecoverable tape error writing (volid). Command canceled.
EE10 - Unrecoverable tape error reading (volid). Command canceled.
EE14 - Punctuation error in operand field. Command canceled.
EE19 - (Dsname) not PS data set. Command canceled.
EE30 - Use of (RPS/CVV) command unauthorized. Command canceled.
EE31 - Volume parameter missing. Command canceled.
EE32 - Volume parameter invalid. Command canceled.
EE33 - Device type code invalid. Command canceled.
EE34 - Public volume invalid. Command canceled.
EE35 - Unable to mount volume. Command canceled.
EE36 - Use EVV command for private volumes. Command canceled.
EE37 - Hardware error processing volume. Processing terminated.
EE38 - Error in (ddname) data set. Define new data set and restart with this volume.
EE39 - (Ddname) data set not found. Command canceled.
EE40 - Cannot read volume label. Command canceled.
EE41 - Organization of (ddname) data set not VI. Command canceled.
EE42 - Error is Userid. LOGOFF and LOGON before proceeding.
EE43 - Opt parameter invalid. Command canceled.
EE44 - Opt and volume parameters inconsistent. Command canceled.
EE45 - Unable to determine volid of input tape. Command canceled.
EE46 - Input DDEF failed. Command canceled.
EE47 - ACV parameter invalid. Command canceled.
EE48 - Start parameter invalid. Command canceled.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Interrupt Storage Area (CHAISA)
Symbolic Device Allocation Table (CHASDA)
System Common (CHASCM)
Format-E DSCB (CHADSE)
Format-F DSCB (CHADSF)
Public Volume Table (CHAPVT)
Catalog SBLOCK (CHACCC)
Data Control Block (CHADCB)
ABEND Interlock Release Table (CHAAIR)

EDITS: The following edits are performed for all data sets by checking the Format E DSCB:

- DSCB CHECKSUM
- VALID DSORG
- Zero total pages
- Zero data pages (OK FOR RPS, CVV, CPS, LPDS OUT D/S)
- Total pages less than total in use
- VS data set with Directory pages
- VS data set with Overflow pages
- VS data set valid record format
- VS data set record lengths a page multiple (For Format 'U')
- VI data set valid record format
- VI data set key length zero
- VI data set with more than 255 directory pages
- VI data set with more than 240 overflow pages
- VI data set keylength + keyoffset greater than record lengths
- VI data set invalid record length

The following data sets are ignored:

- CATALOG
- SYSOPER0 D/S
- TSS USERLIB
- TSS SYSOUT
- BWQ

The SYSLIB data set is renamed CUV.SYSLIB, or RPS.SYSLIB.

SECURE Command Routine (CZAFU)

This routine is called once by a nonconversational task to reserve all devices that will be needed during execution of that task. The routine is also called by BULKIO tasks to reserve the devices required by them. If the SECURE command is invoked by EXPRESS BATCH, a diagnostic message is issued and the task exits to ABEND. SECURE operates in nonconversational mode only. (See Chart CQ.)

ENTRIES:

CZAFU1 - normal entry
CZAFU2 - batch monitor entry

MODULES CALLED:

MTREQ (CZCAA1) Allocates requested device(s).

NEXTPAR (CZAAC1) Fetches and validates command operands.

MSGWR (CZAAD2) Issues system messages.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The command operands are fetched and validated using the SCAN routine, NEXTPAR. These operands describe the number and type of devices needed by the task. If entry is from CZAFU1, the routine will reserve any of these devices:

1. 7-track tape (with or without data conversion)
2. 9-track tape
3. Direct access devices (2311 and 2314)

In addition to the above, the routine will reserve any of these devices for an entry from the batch monitor or a class E user.

1. Card reader
2. Punch
3. Printer

After verifying operands, SECURE creates a device type code for each requested device and inserts these codes in a parameter list for MTREQ. If any of the requested devices are not available when MTREQ is called, the request remains enqueued and the task is suspended until all required devices become available.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For the messages issued, see System Messages.

SYSER Code	Severity	ABEND Message	Significance
050507001	Minor	SYSTEM ERROR, INVALID RETURN FROM MTREQ.	MTREQ returned an invalid code.
050507002	Minor	SYSTEM ERROR. TASK TERMINATED.	Error has been found in command received from batch monitor.

SYSTEM CONTROL BLOCK USAGE:

System Common (CHASCM)
Task Common (CHATCM)

SHARE Command Routine (CZAFI)

This routine enables an authorized user to access a data set in another user's catalog. (See Chart CR.)

ENTRY: CZAFI1 - normal entry

MODULES CALLED:

NEXTPAR (CZAAC1) Gets command operand.

CHEKDS (CZAAC2) Validates data set name.

ALFNUM (CZAAC3) Validates user identification.

MSGWR (CZAAD2) Issues system messages.

SHAREUP (CGCFU) Creates sharing descriptor in owner's catalog.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: The routine obtains its input parameters, and checks them one at a time for validity. The first check is made on the name that will appear in the user's catalog pointing to the owner's catalog. This name must be both legitimate and unique (see note below). A pointer to this name is put in a parameter list for SHAREUP.

The owner's user identification (userid) is validated next, and then the owner's data set name is checked. If, instead of a data set name, *ALL has been entered (indicating that all of the owner's cataloged data sets are to be shared), a pointer to the owner's userid is put in the parameter list as his fully qualified name. For all other cases, the owner's userid, followed by the data set name, is moved to a save area and a pointer to this area is put in the SHAREUP parameter list.

SHAREUP is called and if all entries are error-free, it creates the sharing descriptor entry in the user's catalog.

Note: If SHAREUP finds that a nonunique data set name is already pointing to the correct owner's data set, a normal return is effected.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For the message issued, see System Messages.

The routine may issue these system errors:

SYSER Code	Severity	ABEND Message	Significance
050505800	Minor	INVALID RETURN CODE FROM CATALOG SERVICE ROUTINE.	SHAREUP returned invalid code.
050505801	Minor	INVALID RETURN CODE FROM NEXTPAR AFTER SCANNING USER DSNAME.	NEXTPAR returned invalid code.
050505802	Minor	INVALID RETURN CODE FROM CHEKDS AFTER SCANNING USER DSNAME.	CHEKDS returned invalid code.
050505803	Minor	INVALID RETURN CODE FROM NEXTPAR AFTER SCANNING OWNER ID.	NEXTPAR returned invalid code.
050505804	Minor	INVALID RETURN CODE FROM NEXTPAR AFTER SCANNING OWNER DSNAME.	NEXTPAR returned invalid code.
050505805	Minor	INVALID RETURN CODE FROM CHEKDS AFTER SCANNING OWNER DSNAME.	CHEKDS returned invalid code.
050505806	Minor	USERID NOT IN CATALOG.	Userid of user issuing SHARE command is not in catalog POD.
050505807	Minor	INVALID RETURN CODE FROM ALFNUM AFTER SCANNING OWNER ID.	ALFNUM returned invalid code.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Interrupt Storage Area (CHAISA)

SYNONYM/DEFAULT Catalog Routine (CZATR1)

This routine is used to enter or delete synonym and default values in the combined dictionary (CD). (See Chart CS.)

ENTRY: CZATR1 - normal entry

MODULES CALLED:

Procedure Expander (CZATE2) — Constructs tables of pointers to the synonym or default names.

Dictionary Handler:

DELENT (CZASD6) — Deletes entries from the combined dictionary.

ENTR (CZASD5) — Adds entries to the combined dictionary.

RFR (CZASD3) — Scans the combined dictionary for matching entries.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: Upon entry to SYN/DEF, a test is made to determine if synonym values are to be cataloged. If they are, a flag, CZATR9, is set on. The Procedure Expander routine BUILDLIST is now entered to con-

struct two tables. The ELIST table contains pointers to the SYN/DEF names, and ILIST pointers to their corresponding parameters or values, where each is preceded by its length characteristic. During execution of BUILDLIST, each isolated parameter is checked for a synonym value; thus SYNFLG must be enabled to allow a name, which may already be a synonym name, to be redefined. Since BUILDLIST expands the source list (SL) while interrogating the SYN/DEF parameter strings, its existing state must be saved to be restored prior to exit.

After the tables are constructed, each ELIST entry (that is, the value to which it points) is hashed via the HASH macro instruction, and the end result is input to the RFR routine to determine if the name currently exists in the combined dictionary. If not, the corresponding ILIST entry is checked for a null value, If it is, a warning message is sent to the terminal, and the next ELIST entry is processed; otherwise, it is entered in the combined dictionary via the ENTR routine prior to processing the next entry.

If the ELIST entry already existed in the combined dictionary, a test is made to determine if it is the right type (that is, a SYN or DEF type). If not, it is ignored, and the search through the combined dictionary continues; otherwise, its corresponding ILIST entry is tested for a null value. If null, the name is deleted from the dictionary via the DELENT routine; otherwise, the new value replaces the existing value, via ENTR.

After all of the entries have been processed, CZATR9 is disabled, the SL is returned to its state prior to entry, and an exit is made to the calling routine.

ERROR CONDITIONS: There are no error conditions for this routine.

SYSTEM CONTROL BLOCK USAGE:

Source List Page Header (CHASLP)
Sublist Header (CHASLH)
Profile Character and Switch Table (CHAPCT)
New Task Common (CHANTC)
Control Dictionary Entry (CHADEN)

System Activity and Resources Display (SARD) Processor (CZAYE)

This routine is called to gather information relative to System Activity and Resources. It summarizes current system activity, pending work, and current resources into a convenient display, which is written to SYSOUT. (See Chart CT.)

ENTRY: CZAYE2 - normal entry

MODULES CALLED: None.

EXITS: CZAYE returns to its calling routine.

OPERATION: This routine may be validly entered only as a result of a command from an operator task or a manager/administrator task. If the SARD table (CHASAR) is not current (SARCUR=X'01'), CZAYE sets the CHASAR lock byte (SARLCK), and proceeds to calculate and extract the latest system values for the CHASAR fields that are not dynamically updated. The data is gathered into CHASAR from various system tables. Some of the data in the SARD table will be updated dynamically by the Batch Monitor and may be extracted at face value.

When CZAYE has completed gathering and tabulating data in the SARD table, it issues a GATWR to write a header. Subsequently, CZAYE converts and formats each line, and issues a GATWR to write each line.

After the last line is processed, CZAYE unlocks the SARD table and returns to its calling routine.

If CHASAR is current (SARCUR=X'01'), CZAYE locks CHASAR, but bypasses the gathering of information, assuming the most recent time-activated accumulation of data for the display. This is accurate to within 30 seconds.

ERROR CONDITIONS: If this routine is invoked by other than an operator or manager/administrator task, a diagnostic message is written to GATE.

SYSTEM CONTROL BLOCK USAGE:

System Common (field:SCMTTS)
VAM2 Public Volume Table (CHAPVT)
Active User Table (CHAAUL)
Symbolic Device Address Table (SDAT)
System Activity and Resources Table (CHASAR)

SYSXPAT Command Routine (CZATF)

This privileged routine calls a command analyzer and executor command directly and passes its parameters in original textual form. (See Chart CU.)

ENTRY: CZATF1 - normal entry

MODULES CALLED: None.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: This is a BUILTIN procedure. If the SYSXPAT command ended with a semicolon, SYSXPAT transfers the parameter list beginning on the remainder of the line from the source list to the buffer in task common. If not, it passes the parameter list beginning on the next line. If the line is a continuation line the following line (2) is also copied. SYSXPAT contains a table of names and external symbols which it uses to determine the symbol to be called.

ERROR CONDITIONS: There are no error conditions for this routine.

SYSTEM CONTROL BLOCK USAGE: Task Common (CHATCM)

Compatibility Handler (CZATF1)

This module allows execution of CLI commands (for example, DDEF, PRINT, EXECUTE, etc.) through the Release II Command System. (See Chart CU.)

ENTRY: CZATF1 - normal entry

MODULES CALLED:

User Prompter (CZATJ1)	Prompts the user.
SYSIN (CZASC7)	Retrieves the parameter string from the source list and place it in task common.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction.

OPERATION: Upon entry to CZATF1, the input command name is compared to a table of compatibility names. If a match doesn't occur, the user is notified, via the PRMPT macro instruction, and an immediate exit is

taken. If a match does occur, the SYSIN macro instruction is executed in order to retrieve the routines parameter string from the source list (SL), and place it in the task common. If the string has continuation lines (that is, last character in the preceding line is a dash), they are also read from the SL until there is a final continuous parameter string in task common. If the total number of bytes passed exceeds 1500, the user is notified via PRMPT, and an immediate exit is taken. A special termination byte (X'27') is now placed at the end of the string in task common, and a series of tests is made to determine if the desired command is FTN, ASM, or LNK. If it is, an immediate exit is taken; otherwise, the routine is executed via a CALL on the adcon pair associated with the input command name. Upon return, an immediate exit to the calling routine is made.

ERROR CONDITIONS: The user is prompted and an exit is taken if the input command name isn't found in the compatibility table. The user is prompted and an exit is taken if the parameter string exceeds 1500 bytes.

SYSTEM CONTROL BLOCK USAGE: Task Common (CHATCM)

TIME Command Routine (CZAVB)

This routine permits a time interval to be specified for the completion of a task. Either of two clocks may be set. System Clock 14 is set to the value specified by a TIME command. It is initialized at LOGON time to the system time specified during System Generation. Express Batch Clock 13 is set to the value, if any, specified by an 'EXPRESS' control card during Express Batch processing. (See Chart CV.)

ENTRIES:

CZAVB1 - The VMTI entry initializes the timer to a value specified by the SCMTIM field of System Common (CHASCM).

CZAVB2 - The normal command entry is entered when a TIME command is issued.

CZAVB3 - The INTERRUPT routine entry prints a message to the conversational user indicating his time has elapsed and resets System Clock 14 to an additional one minute value to permit user continuation. A nonconversational task is ABENDed when it is interrupted by System Clock 14. A task in Express Batch mode is ABENDed by the elapse of Express Batch Clock 13. An indicative diagnostic message is issued for the elapse of either clock.

CZAVB4 - The EXPRESS entry is entered in Express Batch mode to initialize Express Batch Clock 13 to the time value defined on the 'EXPRESS' control card. It is reentered after LOGON for each sub-task within the Express Batch to reinitialize the clock.

MODULES CALLED:

NEXTPAR (CZAAC1)	Scans the parameter for the first nonblank and for the character length.
CHKNUM (CZAAC5)	Converts the parameter, N, into a binary number.
ABEND (CZACP1)	Abends whenever time interval has elapsed while in nonconversational mode.
User Prompter (CZATJ1)	Issues elapsed time and error messages.

EXITS: The routine normally returns to the calling routine via the RETURN macro instruction. There are no error exits. ABEND1 is taken for an interrupt entry due to Express Batch Clock 13 or a background task interrupted by System Clock 14.

OPERATION: Entry via the VMTI entry point at LOGON time causes a time value to be picked up from the SCMTIM field of System Common, as specified during System Generation. This time value is used to initialize System Clock 14.

CZAVB2 is entered as a result of the TIME command. Register 1 contains a pointer to the address of the time parameter, N. NEXTPAR (CZAAC1) is called to scan the parameter. NEXTPAR's output, the length of the parameter and the address of the first nonblank character, is used as input to CHKNUM (CZAAC5). Then CHKNUM validates the parameter and converts it to a binary number. Any unsigned decimal integer greater than zero and less then 451 is valid. If the integer is invalid, PRMPT sends a message to the task and the command is ignored.

The INTERRUPT entry disables the clock not causing the interrupt to inhibit further interrupt action. A message is written to SYSOUT identifying the source of the interrupt. If System Clock 14 caused the interrupt and the task is conversational, System Clock 14 is reinitialized to a one minute interval and the routine returns to the calling routine. If the task is nonconversational, a completion code 1 ABEND occurs after a message is written to SYSOUT.

The EXPRESS entry utilizes the parameter scan and validation routines called by CZAVB2. If the time integer is valid, System Clock 14 is reinitialized to the new value. Then CZAVB4 examines, on the first pass, the time parameter entered on the 'EXPRESS' card and initializes Express Batch Clock 13 accordingly. On subsequent entries the clocks are reinitialized from the value saved from the first pass.

ERROR CONDITIONS: If the integer which is presented with the TIME command does not lie within the limits 0<N<451, the following message is printed on the operator's terminal:

TIME INTEGER N INVALID. REQUEST IGNORED.

SYSTEM CONTROL BLOCK USAGE:

System Common (CHASCM)
Task Common (CHATCM)

Userid Informational (UID) EXHIBIT Processor (CZAYG)

The EXHIBIT Director (CZAYD) calls CZAYG to process requests for an information display depicting all active tasks in the system. (See Chart CW.)

ENTRY: CZAYG1 - normal entry

MODULES CALLED: None.

EXITS: CZAYG returns to its calling routine, CZAYD.

OPERATION: If the task requesting the UID option is not an operator or manager/administrator tasks, this routine flags the form to be presented as 'short'. This means that the number of userids presented per line (UINUMPER) is set to five rather that to one, as it is for the 'long' form. (If the SYSOUT is a teletype, UINUMPER is set to three for short.) The operator or manager/administrator may specify the short form in his command format.

The 'long' form displays, in addition to the userid, taskid, and status, the SDA/BSN, the time the task was activated, how much temporary storage is being used, and the devices in use. All of this information required for the UID display is gathered from the Active User List Table (CHBAUL).

CZAYG moves all active and applicable entries from CHBAUL to the work area provided by its calling routine, CZAYD. Using GATE, CZAYG then writes headers to SYSOUT, and converts and writes via GATWR each line of data in the work area. When the data lines are exhausted CZAYG returns to its calling routine.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE: Active User List (CHBAUL)

UPDTUSER Command Routine (CZAGC)

This routine may only be invoked by a privileged system programmer, authority code '0', to initialize the counts of data sets for the system and for each user. (See Chart CX.)

ENTRIES:

CZAGC1 - command entry
CZAGC2 - macro entry

MODULES CALLED:

Set Interlock (CZCOH)	CZCOH1 locks the User Table RESTBL for WRITE.
Release Interlock (CZCOI)	CZCOI1 unlocks the User Table RESTBL.
LOCATE (CZCFL)	CZCFL1 finds a catalog entry for a given FQN.
User Prompter (CZATJ)	CZATJ1 writes messages to a user; CZATJ2 forms messages to write to the operator.
WTL (CZABQ)	CZABQ1 writes a message to the operator log.
SETXP (CEAH7)	An SVC 244 is issued to read in a DSCB page.
ERASE (CZAEJ)	CZAEJ3 erases a temporary data set.
Virtual Memory Allocation (CZCGA)	GETMAIN (CZCGA2) reserves a page for DSCB processing; FREEMAIN (CZCGA3) frees buffers which were obtained for DSCB processing and TBLOCKs.

EXITS: The routine returns to the calling routine via the RETURN macro.

OPERATION: The authority code of the user is checked. If the authority code is not '0', no processing is performed and the routine returns to the caller. Otherwise, GETMAIN is called to reserve a page for DSCB processing, the User Table data set is opened for update and Set Interlock locks the User Table RESTBL. The following steps are then performed for each User Table entry:

1. A User Table entry is read in.

2. LOCATE generates TBLOCKs for all data sets in the user's catalog and returns the address of the first TBLOCK in the TBLOCK chain.

3. One of the following steps is performed for an entry in the TBLOCK chain:

   - If a TBLOCK is not a DSD entry, or the data set is owned by another user, or the TBLOCK is a DSD entry for a private volume, the TBLOCK is bypassed and the next TBLOCK is located in the chain.

   - If a TBLOCK is a DSD entry for a public volume and the owner of the data set is the user whose User Table entry is currently being scanned, LOCATE is called to obtain the catalog entry for the data set. If the data set is temporary, ERASE is called to delete the data set and the next TBLOCK is located in the chain. If the data set is not temporary, the Format-E DSCB for the FQN is read in via SETXP and a checksum is calculated. If the checksum is in error, a WTL is issued to write a message in the operator log and the next TBLOCK is located in the chain. If the checksum is correct, count fields in System Common and the User Table are updated and the next TBLOCK is located in the chain.

4. When the end of the TBLOCK chain is reached, FREEMAIN is called to release the space reserved for the TBLOCKs.

5. If the userid in Task Common matches the userid of the current UPDTUSER command user, the updated User Table count fields are stored in the user table entry pointed to by TCMVLU in task common.

6. An updated User Table entry is written out.

After all User Table entries have been processed, Release Interlock releases the lock on the User Table, the User Table data set is closed, FREEMAIN releases the page reserved for DSCB processing, a completion message is issued and the routine returns to the caller.

ERROR CONDITIONS: If an error occurs, User Prompter is called to issue a diagnostic and processing continues. When processing is complete, a return code of 4 is set in register 15 and the routine exits to the calling routine.

SYSTEM CONTROL BLOCK USAGE:

Catalog SBLOCK (CHACCC)
Data Control Block (CHADCB)
Format-E DSCB (CHADSE)
Interrupt Storage Area (CHAISA)
Public Volume Table (CHAPVT)
System Common (CHASCM)
TBLOCK Data Set Descriptor (CHATBD)
Task Common (CHATCM)
User Table (CHAUSE)

USAGE Command Routine (CZAGB)

This routine displays on the user's SYSOUT all the non zero accounting information associated with the specified userid. (See Chart CY.)

ENTRIES:

CZAGB1 - macro entry
CZAGB2 - command entry

MODULES CALLED:

SCAN Routines:
   NEXTPAR (CZAAC1) Gets command operand.

ALFNUM (CZAAC3)	Checks for alphabetic first character.
CHKNUM (CZAAC5)	Checks for all numeric characters.
MSGWR (CZAAD2)	Issues system diagnostic.
RCR OPEN	Opens a user entry.
RCR UPDATE	Updates user accounting information.
RCR CLOSE	Closes a user entry.
REDTIM (CEAS6)	Gets current time.
XTRTM (CEAT1)	Gets current CPU time used by task.
User Prompter (CZATJ1)	Writes RESET diagnostics to SYSOUT.
GATE (CZATC1)	Writes data on SYSOUT via GATWR.
WRITE (CZCPE1)	Writes reset user entry to SYSUSE.
SYSER (CEAIS)	Reports system failures.

EXITS: The routine returns to the calling routine via the RETURN macro instruction.

OPERATION: Upon entry an indicator is set to show whether entry was from a command or a macro. Then the user's privilege is checked for F, B, or A. If privileged (F, B, or A) the USAGE command operand list is processed. Otherwise, the user's own user entry address is retrieved from task common and processed as with reset option 'N' specified. In this former case, the USERID operand specified is checked for validity. If the USERID is not valid, a diagnostic message is issued, a return code of 4 is set in register 15 and control is returned to the caller.

For a valid USERID, processing continues by checking to see if it is the same as that in task common. If it is or if no USERID was specified, the user entry pointed to by TCMVLU is processed. If the USERID is different from that in task common, an RCR OPEN is done to OPEN that USERID user entry. A failure by RCR OPEN causes a system diagnostic, invalid USERID, and the command is cancelled.

In both cases, the RESET operand specified or defaulted is analyzed. If an invalid reset parameter (other than 'Y' or 'N') is specified, the conversational user is prompted and reprompted to this effect until he enters a 'Y' or 'N' or defaults. For the nonconversational user, the invalid reset parameter is ignored and processed as the 'N' or default parameter. If 'Y' has been specified, the user privilege is checked for 'F' or 'B'. If either of these privileges pertains, a reset switch is set preparatory to reset processing of the user entry fields. If neither pertains, a User Prompter message is presented indicating the RESET option is not available to the user's privilege class and control is returned to the caller with a return code of 8 in register 15.

Once the user entry to be processed is located and the reset parameter analyzed, an RCR update is performed to insure that the accounting information is current. For each of six RESOURCE PRODUCT fields, the corresponding user entry RATION field, CURRENT ALLOCATION field, and non zero PRODUCT field, in that order, are converted to decimal EBCDIC and moved along with their appropriate resource header to output BUFFER1. If the product is zero, only the RATION is converted and moved to BUFFER1 followed by a semicolon indicating the zero PRODUCT field condition.

Next the active background TASK count RATION and CURRENT TASK COUNT are converted to decimal EBCDIC and moved along with their appropriate corresponding header to OUTPUT BUFFER2. However, if the CURRENT TASK COUNT field is zero, this is indicated by the semicolon following the ration value. The BULKIN and BULKOUT ACCUMULATIVE fields are converted and moved to BUFFER2 along with their respective headers only if they are non zero.

The converted CPU TIME RATION is moved to BUFFER2 and if the user's own entry is being processed (RCR OPEN has not been done), the CURRENT CPU TIME is extracted, converted, and moved to BUFFER2 following the RATION semicolon. Then the ACCUMULATIVE CPU TIME value, if non zero, is converted and moved to BUFFER2.

The converted CONNECT TIME RATION is moved to BUFFER2 and, if RCR OPEN has not been done and the current task is conversational, the current connect time is extracted, converted, and moved to BUFFER2.

Note: CURRENT CPU TIME has the converted format of min:sec:millisec; all other time fields have the format of hr:min:sec.

At this point the RESET parameter switch is checked as set previously. If it is on, ten RESOURCE FIELDS in the user entry are RESET to zero. If an RCR OPEN was done to move the user entry from SYSUSE to VM, an RCR CLOSE is done. However, if RCR OPEN was not done, the RESET user entry is WRITTEN out to SYSUSE.

Finally a check is made for the type of entry. For a command, the two output buffers' contents are written to SYSOUT using two GATWRs. For a macro call, the data in the buffers is moved to the caller's buffer.

At this point processing is complete and the module returns to its caller via the RETURN macro.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code will be returned in register 15:

Code	Significance
00	normal return
04	invalid userid
08	invalid privilege for specified userid
12	attention interrupt during PRMPT I/O

If an invalid userid is specified, a diagnostic is issued and the module returns to the caller. If the RCR CLOSE fails, a minor system error is issued followed by a return to the caller.

SYSER Code	Severity	Explanation
05062501	Minor	RCR CLOSE failed.
05062502	Minor	Significance error in millisecond value of CPU time.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Task Status Index (CHATSI)
User Table (CHAUSE)
Active User Limits Table (CHAAUL)
Data Control Block (CHADCB)
Shared Data Set Member (CHASDM)

Active User List Table (CHAAUL)
Data Control Block (CHADCB)
Shared Data Set Member (CHASDM)
Interrupt Storage Area (CHAISA)

User Control (CZAMZ)

The User Control routines are responsible for preserving the Command System interfaces with nonprivileged programs, for processing the CLIC, PAUSE, CLIP, COMMAND, LPCINIT and LPCEDIT macros, for processing the ENABLE, DISABLE, END and transient commands and for dispatching functions of the Text Editor. In addition, it dispatches a LOAD, UNLOAD or parameterless CALL command. (See Chart CZ.)

ENTRIES: There are twenty-three entry points to this routine:

1. CZAMZ1 is entered by a type-I call from the Command Analyzer's BUILTIN Call Processor and from other routines in CZAMZ to dispatch a nonprivileged module. Register 1 points to a builtin procedure key (BPK).

2. CZAMZ19 is entered when a nonprivileged module issues a RETURN macro to return to the Command System.

3. CZAMZ2 is entered when a nonprivileged module issues an EXIT macro.

4. CZAMZ3 is entered from the Attention Handler when an attention interrupt is received and the user enters a command other than one of the attention response commands or ABEND. It is also entered from DIAGNO (CZAHA).

5. CZAMZ5 processes a CLIC or PAUSE macro.

6. CZAMZ6 processes a CLIP or COMMAND macro.

7. CZAMZ7 is the command entry to RTRN processing. RTRN is a stack manipulating command.

8. CZAMZ8 processes the EXIT command. EXIT is a stack manipulating command.

9. CZAMZ10 processes the STACK command. STACK is a stack manipulating command.

10. CZAMZ11 processes the PUSH command. PUSH is a stack manipulating command.

11. CZAMZ12 cancels attention levels for the input name. Called by CZAMZ1.

12. CZAMZ13 is a service routine for other User Control modules. It obtains a recursive save area.

13. CZAMZ14 deletes LPC programs from CZAMZ's stack.

14. CZAMZ15 is the ABEND and task monitor entry to RTRN processing.

15. CZASW1 processes an LPCINIT macro.

16. CZASW2 processes an END command.

17. CZASW3 processes a transient command.

18. CZASW4 processes an LPCEDIT macro.

19. CZASW5 is entered by the Command Analyzer to dispatch a function of the Text Editor.

20. CZASW9 processes an implicit END situation.

21. CZASW10 processes an ENABLE command.

22. CZASW11 processes a DISABLE command.

23. CZBTG1 dispatches a LOAD, UNLOAD or parameterless CALL command.

MODULES CALLED:

User Prompter (CZATJ1)	Writes error messages to the user's terminal.
Command Analyzer	
CZASA1	Analyzes input.
CZASA6	DIRs the system attention handler (USATT)
CZASA7	SIRs the system attention handler (CLATT)
Source List Handlers	
CZASC6	Deletes remainder of line in source list for EXIT.
GATE (CZATC1)	Writes module names to SYSOUT for STACK.
QLE Processor (CZVJQS)	Puts a QLE to the Command Analyzer (CZASA1)
CLEANUP (CZCJCU)	Performs Task Monitor cleanup for the attention response commands.
Edit Initializer (CZBSE1)	Called by LPCINIT to initialize the Text Editor.
DATALINE (CZASG1)	Prompts for input into a data set.

EXITS: Exits are described in the following section.

OPERATION: CZAMZ1 is invoked to process calls from the Command System to nonprivileged routines. In CZAMZP, it maintains a push-down stack of save areas, with a maximum of ten levels. One level is associated with each call. Each level contains the backward pointer to CZAMZ1's caller, a copy of ISALS1 at the time CZAMZ1 is called, and a save area the called routine uses to store CZAMZ's registers.

The User Controller is called at CZAMZ1 to process calls from the Command System to nonprivileged routines. In CZAMZP it maintains a ten-entry stack (one level of entry for each level of call allowed). Each level contains the backward pointer to the routine that calls CZAMZ1, a copy of ISALS1 at the time CZAMZ1 was called, and other pertinent information about the previously dispatched module called by CZAMZ1. It then sets up the same information in its PSECT for the module to be dispatched.

On entry, if the stack is not full, the count of currently active programs is incremented, and the ISALS1 is saved in the CZAMZP stack. (ISALS1 is saved because certain fields in it are overlaid to dispatch the current program.) The status of PIREC, LPC, USATT, and AETD will also be saved in the stack. If the previous invocations should be canceled, this routine branches out to CZAMZ14 to cancel the levels for this routine. After saving pertinent information in the stack for the previously dispatched module, CZAMZ1 sets in its PSECT information about the module about to be dispatched. Status information saved includes

the names of the routine being invoked (whose address is placed in Register 0), the address of the caller of CZAMZ1, the current source list pointer (CZAMZL), and dispatch option flags (CZAMZF). This routine is marked as active (CZAMZF=X'80') and the pointer to the current stack entry (CZAMZA) is incremented.

To enable the routine about to be dispatched to return to the Command System correctly, this routine set Register 14 in the ISA to point to CZAMZ19. The address of the routine to be dispatched is stored in Register 15. The VPSW also is set to point to the routine to be dispatched. If parameters are included in the call, Register 1 points to the parameter list. The address of the user portion of the current save area is saved in Register 13. This routine then returns control to the Task Monitor, through a field in New Task Common (NTCTMR). The Task Monitor then issues a LVPSW from ISALS1 and dispatches the routine.

CZAMZ19 is entered when the nonpriviliged routine dispatched by CZAMZ1 issues a RETURN macro. Its only function is to issue the EXIT, NOMSG macro to return control to CZAMZ2.

CZAMZ2 is the EXIT SVC processor. If its caller is the Task Monitor, it saves Register 13 (pointer to the save area the Task Monitor used when it last called the Command System) in New Task Common (NTCTMR). Then (or for another caller), if USATT is active, this routine calls the Command Analyzer at CZASA7 to release control of the attention interrupts (CLATT). If no other programs exist in the stack (level=0), this routine performs a CLEANUP, processes a QLE to the Command Analyzer, restores Register 13 from NTCTMR and returns to its calling routine.

However, if other levels are filled in the stack, this routine finds the first active level, moves the stack status information into the current fields, clears the stack entry, and resets the pointer so that this previous entry becomes the current one. If the exiting module issued a PUSH command, this routine exits to a special entry point in the Attention Handler, CZASB3, which causes a return to the Command Analyzer via QLE. Otherwise, Register 13 is restored for CZAMZ1's calling routine and this routine exits.

CZAMZ3 (sometimes called INTERVENE) is called by the Attention Handler to process a command string entered after an attention interruption if that command string is <u>not</u> one of the system-defined responses to ATTN (such as STRING). CZAMZ3 uses the Command Analyzer to process this string, but first, if the user has specified a USATT, this routine calls the Command Analyzer at CZASA7 to SIR the system attention handler (CLATT). It then specifies a non-OBEY entry by setting Register 0 to 0. If a recursive save area is not free, it calls CZAMZ13 to provide a recursive save area.

CZAMZ3 does not expect a return from the Command Analyzer unless the command in the input string is GO. In case it is GO, this routine sets CZAMZL in Register 6 before calling the Command Analyzer at CZASA1. On return, CZAMZ3 returns the recursive save area it acquired, and, if the currently active program had issued a USATT, calls the Command Analyzer at CZASA6 to DIR the system attention handler.

CZAMZ5 processes a CLIC or PAUSE macro; CZAMZ6 processes a CLIP or COMMAND macro. Both call CZAMZ3, passing it a null parameter string. On return, both return to the Task Monitor. (Note: If the task is non-conversational, CZAMZ5 bypasses the call to CZAMZ3.)

<u>Attention Command Processing</u>: The User Controller contains the logic for processing four stack manipulating commands -- RTRN, EXIT, STACK, and PUSH.

- RTRN is processed by CZAMZ7/CZAMZ15. This command is used to return the user to the command system with all previous source lists and programs forgotten, thus allowing him to bypass the normal completion of these programs and source lists. This function is required within a multi-level nonprivileged environment to give the user control over the level at which he is running. This command forces all active nonprivileged programs to exit, deletes all user SIRs and AFTDs. The user is returned to level zero. The Text Editor, if active, remains active on completion of RTRN. A CLEANUP is performed, and the command (or macro) processing ends by processing a QLE to the Command Analyzer and returning to the Task Monitor.

- EXIT is processed by CZAMZ8. This command (or macro) forces the currently active, user-invoked, nonprivileged program to exit. Again the user is allowed to bypass the normal completion of the currently-active nonprivileged program. EXIT checks that the user level is greater than zero (which it must be), and if there is any user SIR routine active (ITBACT does not equal zero). If there is an active routine and SIRTEST=N has not been specified, the operation is terminated. EXIT returns control to the Command System so that the command string from which the nonprivileged module was invoked will be resumed. This means that no commands which follow EXIT will be executed. The Command System will respond with an underscore; the user can resume execution of the previous level of nonprivileged program.

- STACK is processed by CZAMZ10. This command displays on SYSOUT all currently active user-invoked module names. The active level number must be greater than zero. If there are any active nonprivileged SIR routines, the message "SIR ROUTINE ACTIVE" is displayed. The module names are displayed in descending order from the currently active down. SCAN and END routines will be uniquely identified. GATE is used to write the module names, one line at a time. Attention is checked for after each line; attention terminates the operation.

- PUSn is processed by CZAMZ11. This command saves the status of the last non-completed, nonprivileged program in the system save area or stack (providing it is not a Text Editor command). The system save area stack must not be full, and, if there is any user SIR routine active (with SIRTEST=N not specified) this operation is cancelled. The current program is moved into the CZAMZP save area stack, and the currently active level number is incremented by one. In the next entry of the privilege stack, this routine sets the address of a routine which QLEs to the Command Analyzer, and then exits to the Task Monitor. (See CZAMZ2.) This enables a dispatched routine to be interrupted, then PUSHed, then run to completion twice.

Controller Service Routines: Three routines within the User Controller perform services for other Controller routines.

- CZAMZ12 is entered to delete previous entries from the program stack for a particular module name. If a stack entry is found with this name, it is marked inactive and the count of active programs (CZAMZB) is reduced by one. The user is informed of the deletion before CZAMZ12 returns to the calling routine.

- CZAMZ13 expands the save area stack as required through GETMAIN. These recursive save areas are necessary since CZAMZ3 and the Attention Handler (CZASB), both of which obtain save areas, may be reentered before they have been returned to by their called routines.

- CZAMZ14 is entered to delete previous entries from the program stack by type of program (all LPCs could be deactivated, for example). If a stack entry is found marked as a type to be deleted, the entry is

marked inactive and the count of active stack entries is decreased (CZAMZB).

User Controller LPC Processing: The processing of the LPCINIT and LPCEDIT macros, of the END and transient commands, of the text editor functions, etc. is performed by the section of the User Controller whose entry points begin CZASW.

CZASW1, processing the LPCINIT macro, checks whether a new LPC is permitted. A new LPC is permitted if the LPC in progress indicates that interrupts are permitted or if no LPC is in progress. If the LPC in progress indicates that interrupts are permitted, the implicit END routine (CZASW9) is invoked. Then processing continues as if no LPC were in progress. Otherwise or afterwards, the name of the LPC to be initialized is recorded in the Transaction Table for future use. The address of the DCB is stored in the Transaction Table. The address of the END command preprocessor and postprocessor routines, the implicit END routine and the transient command routine are saved if they are supplied by the caller. If no implicit END routine is defined, the END preprocessor is saved as the implicit END routine. The Edit Initialization routine (CZBSE1) is linked to, and, on return, Register 13 is restored from NTCTMR, and control is passed to the calling routine. If a new LPC is not permitted, the user is prompted with the name of the current LPC and asked whether he wishes to terminate it.

To process an END command, CZASW2 checks whether an LPC is in progress. If not, the user is prompted and the routine returns to the caller via the RETURN macro instruction. If the current routine specified no END preprocessor routine, the name of the current LPC in the Transaction Table is cleared, CZAMZ14 is called to clear the LPCs out of the stack, and the END postprocessor is invoked via a call to CZAMZ1. Afterward, an exit is taken. If there is an END routine specified, it is invoked through CZAMZ1, the current LPC name in the Transaction Table is cleared, the END postprocessor is invoked if it exists. Afterward, an exit is taken.

CZASW3 is the transient command processor. If no LPC is in progress, the user is prompted and an exit is taken. If no transient command processor is specified by the LPC, the user is prompted and an exit is taken. Otherwise, it reads a command from the source list via the SYSIN macro instruction, and the transient command processor is invoked, via CZAMZ1, and an exit is taken.

CZASW4, processing the LPCEDIT macro, saves Register 13 (address of the routine that called CZAMZ1) in NTCTMR. Then, if no DCB is open, it calls the implicit END routine (CZASW9), prompts the user that the LPCEDIT has been cancelled, and exits to CZAMZ2. If there is neither an active LPC nor any active programs, this routine also prompts the user and exits to CZAMZ2. However, if all is in order, this routine sets up return linkage to itself for CZAMZ2 to follow and calls CZAMZ2 to pop the stack. In this case, CZAMZ2 returns to CZASW4, which, after acquiring a recursive save area (if one is needed) from CZAMZ13, links to the Text Editor DATALINE routine (CZASG1) to prompt for input. When DATALINE returns, this routine exits to the routine that called CZAMZ1.

CZASW5, entered as a result of a Text Editor command, checks to see if an LPC is in progress. If not, the user is prompted and an exit is taken. If there is an active LPC, but no open DCB, CZASW9 is called to end the LPC, the user prompted, and exit taken. Otherwise, the BUILTIN procedure key (BPK) is altered to contain the adcons for the proper processing routine. The processing routine is then called via CZAMZ1. Upon return to CZASW5, Register 1 is checked to see if a follow-on routine was specified. If no follow-on routine is specified, an exit is taken. If a follow-on routine is specified, CZASW5 loops, calling

follow-on routines as long as they are specified. The routine returns to the calling routine via the RETURN macro instruction.

CZASW10 sets the ENABLE/DISABLE flag, CZASWX, to X'FF' to enable the Text Editor and returns to the calling routine.

CZASW11 sets the ENABLE/DISABLE flag, CZASWX, to X'00' to disable the Text Editor and returns to the calling routine.

CZBTG1 checks to see if there is a name to load or unload. If not, the current name is taken from the AAAMOD field of Table AAA. If Register 1 on entry pointed to part of the UNLOAD BPK, an UNLOAD SVC (SVC 123) is issued and the routine returns to the caller via the RETURN macro instruction. If an unload operation was not requested, the adcon group is armed and a LOAD SVC is issued (SVC 127). If, upon return, the adcon group is still armed, it is assumed that the load operation was unsuccessful and the routine returns to the calling routine. A check is made to see if this was a request for a load operation; if so, the routine returns to the calling routine. Otherwise, a transparent call to CZAMZ1 is issued to dispatch the module to be called. Since this call to CZAMZ1 is transparent, CZAMZ1 returns to the caller of CZBTG1 rather than to CZBTG1 itself.

ERROR CONDITIONS: The routine checks for the following conditions:

1. Pushdown stack capacity exceeded.
2. Page unavailable in which to build recursive save areas.
3. Command requires that LPC be previously defined.
4. Name to be loaded exceeds eight characters in length.
5. DCB not previously opened.

SYSTEM CONTROL BLOCK USAGE:

BPK DSECT (CHABPK)
Data Control Block (CHADCB)
Stack Entry (CHASTK)
Interrupt Storage Area (CHAISA)
TABLEA (CHAAAA)
Profile Character and Switch Table (CHAPCT)
Source List Page Header (CHASLP)
Sublist Header (CHASLH)
New Task Common (CHANTC)
Transaction Table (CHATRN)

VAM Tape Command Routine (CZAET)

This routine processes requests to copy a VAM data set on tape as a physical sequential data set (VT -- VAM to Tape); to restore a physical sequential copy of a VAM data set from tape to direct access storage (TV -- Tape to VAM); or, to produce an identical copy of a VAM data set on direct access storage (VV -- VAM to VAM). (See Chart DA.)

ENTRIES

CZAET1 - VT command entry
CZAET2 - TV command entry
CZAET3 - VV command entry
CZAET4 - VT call entry
CZAET5 - TV call entry
CZAET6 - W call entry
CZAET7 - EODAD entry
CZAET8 - SYNAD entry

MODULES CALLED

CHEKDS (CZAAC2)	Validates data set name.
MSGWR (CZAAD2)	Issues system messages.
GATWR (CZATC1)	Writes messages on SYSOUT.
DDEF (CZAEA1)	Generates JFCBs.
CATALOG (CZAEI1)	Catalogs output tape (copy) data set.
ERASE (CZAEJ1)	Erases output data set if copy not completed normally.
RELEASE (CZAFJ1)	Releases JFCBs generated for the input or output data sets.
LOCATE (CZCFL1)	Finds catalog entry for data set.
GETMAIN (CZCGA2)	Allocates buffer space.
FREEMAIN (CZCGA3)	Releases buffer space.
ADDCAT (CZCFA)	Updates DSORG field in VAM output data set.
WRTM (CZCRBS)	Writes tapemarks.
BUMP (CZCAB1)	Removes mounted volume.
OPEN (CZCLAO)	Opens DCB.
CLOSE (CZCLBC)	Closes DCB.
PAIR (CZACS1)	Creates an entry in the ABEND Interlock Release Table.
BSA LOCK (CZCEJ1)	Set a Virtual Memory Lock.
BSA LOCK (CZCEJ2)	Clears a Virtual Memory Lock.
DSCB RDWR (CZCEM1)	Writes out a DSCB Page. Reads one in.

EXITS: The routine always returns to the calling routine, via the RETURN macro instruction.

OPERATION: The VAM tape module is entered at one of six entry points for the command or CALL specified by the user. A parameter list with a pointer to the address of a character string containing the name of the data set to be copied and, optionally, the name to be assigned to the output data set, is required. Once the parameters have been located, processing continues as required by the particular request.

VAM to Tape Processing: The input dsname is verified. It must be the name of a VAM data set. If a JFCB does not exist and the data set is cataloged, a JFCB is created by a call to DDEF. For the output data set, a JFCB must be found with ddname=DDVTOUT. Specified within this JFCB must be physical sequential organization and a tape volume on a nine-track drive. If this is the first copy operation for the output JFCB, indicated by the absence of the identification *%*% in the TDTDSM field, the output dsname used is the name in the JFCB. The identification is inserted, but no other fields in the JFCB are modified. If not the first copy operation, the output dsname is taken from the operand string, or defaulted, and inserted into the DDVTOUT JFCB. Also updated are the file sequence number (TDTFSQ) and catalog flag (TDTCFL).

Once the data sets are opened, the input JFCB (256 bytes) and the common portion of the input data set's format-E DSCB (37 bytes) are written as the first record (4096 bytes) on the output tape. The remainder of the record is padded with zeros.

Data pages to be copied are located by indexing through the RESTBL for the input data set. For each page to be written, a SETXP SVC is issued with an external address from the RESTBL, to make use of the paging mechanism for input, and is written to tape as a 4096 byte record by BSAM WRITE. Eight buffers are used to overlap processing and input/output time. After the tape operation is completed, processing concludes at a common point. See Final Processing below.

Tape to VAM Processing: The input dsname is verified. It must be the name of a physical sequential data set residing on a nine-track tape. If a JFCB for the data set cannot be found, and the data set is cataloged, DDEF is called to create one. The output dsname must be for a new VAM data set. If a JFCB is found in the TDT chain, it is used; otherwise, one is created by a call to DDEF. An output JFCB is required from the user only if the data set is to be restored to a private VAM volume.

After the data sets have been opened, the first record is read to verify the tape format and to make available the DSCB data necessary to recreate the original data set. Data records from the tape are input by BSAM READ and output by VSAM PUT. (The output data set at this point is treated as a VAM sequential, format U, data set.) Processing continues in this manner until a unit exception occurs. The EODAD routine outputs any remaining records. Processing then concludes as described under Final Processing.

Eight buffers are used. The initial tape read fills all eight buffers. Subsequently, four buffers at a time are filled as the other four are emptied to overlap processing and input/output time.

VAM to VAM Processing: The input dsname is verified. It must be the name of a VAM data set. If a JFCB does not exist and the data set is cataloged, a JFCB is created by a call to DDEF. The output dsname must be for a new VAM data set. If a JFCB is found in the TDT chain, it is used; otherwise, one is created by a call to DDEF. An output JFCB is required only if the data set is to be copied to a private VAM volume.

Both data sets are opened. The common portion of the input data set's format-E DSCB (37 bytes) is retained to recreate the data set once the copy operation has been completed. Data pages to be copied are located by indexing through the RESTBL for the input data set. For each page to be output, a SETXP SVC for the buffer pages is issued with an external address from the RESTBL (input) to make use of the paging mechanism for input, and is written into the output data set by VSAM PUT. (The output data set at this point is treated as a VAM sequential, format U, data set.) When the copy is complete, processing concludes as described under Final Processing below.

Final Processing: Both data sets are closed and all buffers are released. If the data set created on direct access storage is not completed correctly, ERASE is called to delete the partial data set. For normally completed TV and VV operations, the output data set's DSCBs reflect the data set just created by VAM Tape and not the original data set. To correct this, the format-E DSCB for the output data set is read in through a read DSCB routine which is internal to the program. The common portion (37 bytes) of the format-E DSCB is updated from the DSCB data retained from the original data set. Then the output DSCB is rewritten to external storage by PGOUT. Next, the DSORG field in the catalog is updated for the output VAM data set. The output tape data set is cataloged unless the dsname was preceded by an asterisk (*). The

TDTDS2 field for the DDVTOUT JFCB is then filled in with F's. Processing is concluded by releasing any JFCBs created by the routine.

ERROR CONDITIONS: If entry was by macro instruction, a hexadecimal code will be returned in general register 15:

Code	Significance
0C	Input dsname missing.
10	Dsname invalid.
14	Dsname not generation data group.
18	Catalog error.
1C	Dsname not fully qualified.
20	VAM dsname contains *.
24	DISP=OLD in DDVTOUT.
28	Dsname not VAM.
2C	Dsname not extant.
30	DDVTOUT not extant.
34	Volume not 9-track tape.
38	No data to copy.
3C	Unrecoverable output tape error.
40	Unrecoverable input tape error.
44	DSCB error.
48	Unauthorized use of system data set name.
4C	Tape volume not mounted.
50	Punctuation error in operands.
54	Dsname in DDEF not DDVTOUT.
58	New dsname not unique.
5C	No direct access space for new data set.
60	Output VAM data set contains data.
64	Dsname not PS.

The routine checks for acceptable operands and, when called at a command entry point, notifies the user, through MSGWR or GATWR, of any errors found. Messages issued through MSGWR are identified by a message code preceding the message. For precise texts see System Messages.

NORMAL COMPLETION: If entry was by macro instruction, a hexadecimal code will be returned in general register 15:

Code	Significance
00	Normal completion - VT.
04	Normal completion - TV.
08	Normal completion - VV.

The routine, when called at a command entry point, notifies the user, through MSGWR, of a successful completion. The messages are for the VT, TV and VV commands, respectively.

Message Code	Text
BF01	($dsname_1$) COPIED AS ($dsname_2$) ON TAPE, FSQ (fsq), VSN (vsn).
BF02	($dsname_1$) COPIED AS ($dsname_2$) FROM TAPE, FSQ, (fsq), VSN (vsn).
BF03	($dsname_1$) COPIED AS ($dsname_2$).

SYSTEM CONTROL BLOCK USAGE:

Catalog SBLOCK (CHACCC)
Format-E DSCB (CHADSE)
Interrupt Storage Area (CHAISA)

Symbolic Device Allocation Table (CHASDA)
Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)
RESTBL Header (CHARHD)
DCB Header (CHADHD)

VSS Command Routine (CZAVR)

This routine checks the user's authority code to determine if he may use the Virtual Support System (VSS). In order to use VSS, the user must have an authority of O or P. (See Chart DB.)

ENTRY: CZAVR1 - normal entry

MODULES CALLED:

User Prompter (CZATJ1)	Issues diagnostic messages to SYSOUT when errors occur.
READ (CZCPE)	Reads the user table for user's entry.

EXITS: This routine normally exits to TSSS via an SVC 83. Upon return from TSSS, exit is made to the calling routine via the RETURN macro instruction. If an error occurs, the routine exits to the calling routine via RETURN.

OPERATION: Upon entry, the task's userid is obtained. The User Table is READ and the user's authority is examined.

If it is O or P, exit is made to the Time Sharing Support System (TSSS). Otherwise, an insufficient authority diagnostic is issued, and control is returned to the caller. Upon return from TSSS, control is returned to the caller.

ERROR CONDITIONS: The userid of the user issuing the command is checked for the proper authority (O or P). If the user does not have the proper authority, a diagnostic message is issued. If the task calling VSS is nonconversational, a diagnostic is issued, and control is returned to the caller.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Data Control Block (CHADCB)
User Table (CHAUSE)

The command system supports macro instructions which allow object programs, during execution, to use certain command system facilities while still retaining control over processing. Included with these macros is the ATPOL macro instruction, used in checking for attention interrupts before GATE calls. The names and functions of these macro instructions are listed in Figure 12.

Macro instructions supported by the command system may be grouped into four general categories:

1. Those calling command routines.
2. Those calling command support routines (ABEND, FINDJFCB, and FINDDS).
3. Those calling subsystem support routines (GATRD, GATWR, GTWAR, GTWSR, GTWRC and MSGWR).

Macro Instruction Name	Function	Routine to Which Macro Instruction Links
ABEND	To abnormally terminate a task.	ABEND
ATPOL	To check whether an attention interrupt is pending.	None. Macro expansion coding performs action
CAT	To create or update an entry for A data set in the catalog. To create a generation data group index level.	CATALOG Command Routine
CDD	To retrieve and process one or more prestored DDEF commands.	CDD Command Routine
CDS	To make a copy of an already existing data set or member of a partitioned data set.	CDS Command Routine
DDEF	To define a data set and describe its characteristics to the system.	DDEF Command Routine
DEL	To delete one or more data set entries from the catalog.	ERASE/DELETE Command Routine
ERASE	To erase the direct access storage assigned to a data set and remove the data set entry from the catalog.	ERASE/DELETE Command Routine
FINDDS	To locate a JFCB or to create a JFCB for a cataloged data set.	FINDDS

Figure 12. Macro instructions supported by the command system (part 1 of 2)

Macro Instruction Name	Function	Routine to Which Macro Instruction Links
FINDJFCB	To locate a JFCB in the Task Data Definition Table and optionally, insure that correct volume(s) is mounted.	FINDJFCB
GATRD	To read a record from SYSIN.	GATE Routine
GATWR	To write a record on SYSOUT.	GATE Routine
GTWAR	To write a record on SYSOUT, then read the next record from SYSIN.	GATE Routine
GTWSR	To write a record on SYSOUT then read record from terminal.	GATE Routine
GTWRC	To write a record on SYSOUT with carriage control character supplied by program which issued macro.	GATE Routine
MCAST	To make changes in the profile character and switch table.	MCAST/MCASTAB Routine
MSGWR	To issue a message to SYSOUT and, if specified, fetch the response.	MSGWR Routine
PR	To print a data set on a high-speed printer.	PRINT Command Routine
PU	To punch a specified data set onto cards.	PUNCH Command Routine
REL	To delete the JFCB for a data set, to release the I/O devices associated with a private data set, or to remove a job library from the program library list.	RELEASE Command Routine
VSENDR	To send an intertask message and await the response.	XWTO Routine
WT	To write a data set onto tape for subsequent offline printing.	WT Command Routine
WTL	To write a message in the operator log.	XWTO Routine
WTO	To write a message on the operator console and in the operator's log.	XWTO Routine
WTOR	To write a message on the operator console and in the operator's log and get the response.	XWTO Routine

Figure 12. Macro Instructions Supported by the Command System (part 2 of 2)

4. Those used to communicate with other tasks (WTO, WTL, WTOR and VSENDR).

A detailed description of the command routines and command support routines can be found in Section 4, with the exception of the WT, PU and PR command routines. These routines and the XWTO routine can be found in the Operator Task and Bulk I/O PLM, GY28-2047. The GATE routine is described in Section 2.

Macro Instructions that Call Command Routines

Using CAT, CDD, CDS, DDEF, DEL, ERASE or REL macro instructions causes execution of the corresponding command routine. The effect is as if the command itself had been issued; but the code that links to the command routine, as a result of macro expansion, is part of the user's object program and thus active only at object time. The linkage from the object program to the command routine is direct; the command analyzer and executor is not used for this linkage.

The command routine itself performs in the same way whether it is processing a command or a macro instruction. There are two notable exceptions to this statement:

1. The command routine issues no message if it has been entered by a macro instruction. Instead, it supplies a return code to show if an error was found and, in some cases, to indicate the type of error. Error handling is left to the object program that issued the macro instruction.

2. The macro expansion (that is, the object code generated for the macro instruction) links to a special entry in the command routine, not to the entry used in normal command processing. This lets the routine prepare for macro instruction processing; that is, it sets indicators as needed so that no messages will be issued, and makes adjustments to fetch its input parameters. (Unlike the command analyzer and executor, which supplies the address of the input, the macro expansion furnishes a pointer to a location containing the address of the input.)

When the command system routine is finished, it returns control to the calling object program (unless a system error occurred). An appropriate code will be set upon return.

Macro Instructions that Call Command Support Routines

The ABEND, FINDJFCB, and FINDDS macro instructions call upon the ABEND, FINDJFCB, and FINDDS command support routines. Operation of those routines is described in Section 4. Entry to them is always via macro instruction calls, which require no special macro instruction provisions.

The FINDJFCB and FINDDS macro instructions are for system component use only. (A system component is any component supplied as part of the time sharing system or added through a new system generation.) The ABEND macro instruction is normally for system component use, but user programs may also employ it.

Entry to the command support routine occurs during execution of the object program making the call, via linkage generated by the macro expansion. The support routine is then executed as described in Section 4. Return may or may not be made at this point, depending on the macro instruction issued and on whether or not serious errors were found. An ABEND macro instruction terminates the task of the calling object program, so there is no return. Certain errors found during FINDDS or FINDJFCB execution will cause an abnormal termination (via ABEND) of the

calling task, but otherwise return is made to the calling object program.

Macro Instructions that Call Subsystem Support Routines

The GATE routine handles five macro instructions: GATRD, GATWR, GTWAR, GTWSR and GTWRC. These macro instructions may be employed by any user, privileged or not; the GATE routine itself does not check privilege. (Privilege here means the ability to access certain areas of storage. Only systems programs may be privileged; user programs are nonprivileged.)

Entry to GATE is made at the macro entry point; an operation code generated by the macro expansion indicates which of the five macro instructions is being processed. For a detailed description of GATE operation, see Section 2. Upon completion of GATE, return is made to the calling object program.

The ATPOL macro instruction is unusual in that the expansion itself provides the object code that is executed when the ATPOL is honored. No call is made. ATPOL is intended to be used by privileged system programs before linking to GATE. If ATPOL finds an attention interrupt, it returns to the designated location in the calling program. If there is no attention interrupt at this time, it returns to the next instruction in the calling program. Usually, the next instruction is a linkage to GATE.

Macro Instructions That Communicate With Other Tasks

The XWTO routine performs the actions requested by four macro instructions: WTO, WTL, WTOR and VSENDR. WTO, WTL and WTOR each involve some sort of communication with the system operator. The VSENDR macro instruction is used to communicate with any specified task. The expansion of these macro instructions results in linkage to XWTO, with an operation code set to indicate which macro instruction is to be processed. XWTO will then carry out the requested action before returning control to the calling object program.

The Language Processor Controller (LPC) serves as the link between the user and the language processors. It is entirely transparent to the user, for he appears to be communicating directly with the language processor he has chosen. Actually, LPC gathers the input parameters for the language processor, loads the processor, and passes its parameters to the processor. The language processors call upon LPC for source lines, and they use LPC to issue diagnostic messages and invite corrections.

LPC consists of three routines:

LPCMAIN
: has a separate entry point for each language processor, collects the input parameters, calls the language processor and stores the object module in a program library.

GETLINE
: receives the source lines one at a time from the user and creates a source data set (or takes the source lines one at a time from a source data set), and conversationally modifies the source data set if required.

PUTDIAG
: collects diagnostic messages from the language processor, and issues them if appropriate.

When the user calls upon a language processor, LPCMAIN is loaded and entered at the proper entry point for the requested processor. LPC reads the given input parameters, checks them for validity and consistency and, if necessary, issues diagnostic messages and prompts for parameter reentry. LPC then creates data sets for the source and listing lines. It sets up a parameter list and calls the initial phase of the requested processor (thereby loading it).

The processor calls GETLINE whenever it wants an input record. GETLINE either fetches this from SYSIN or from the source data set, depending on whether or not the data set is prestored.

When diagnostic messages have been issued for source statements and the user enters corrections, GETLINE puts these corrections into the source data set and informs the language processor that corrections have been made. The language processor may at this point restart processing at some earlier point in the source data. GETLINE detects this "back-up request" and takes statements from the source data set as long as there are records in it; then GETLINE requests additional input from the user.

PUTDIAG is called by the language processors whenever they want to issue a message to SYSOUT. In conversational mode, PUTDIAG stacks these diagnostic messages until the language processor calls GETLINE for a line of input; then the diagnostic messages are sent to the user, and the keyboard is unlocked so that he can enter corrections. If the diagnostic message stack kept by PUTDIAG becomes full, all diagnostic messages are printed immediately (except for the last one) so that more can be stacked and issued by GETLINE.

When the processor has reached the end of its first phase of processing, it returns to LPCMAIN. The source data set is closed unless the user wishes to modify it. If he chooses to modify it, LPCMAIN calls GETLINE to accept the modifications, and LPC reinitializes the processor

(by calling the appropriate processor entry point) and restarts, using the prestored modified source data set as input. If the user does not wish to modify the source data set, LPCMAIN calls processor continuation, and the language processor continues its activities. It may use PUTDIAG to issue further messages.

When the language processor has finished, it returns to LPCMAIN with an error-level code that indicates the severity of errors. LPC sends a message to the user telling him the level of the error, and stores the object module (if one was created) in a program library. This library is either the last-mentioned JOBLIB data set, or if none has been mentioned, the SYSULIB.

FREEMAIN is called to free some of the storage collected by the language processors and then LPCMAIN returns control to the Task Monitor.

Since LPC and the language processors are all nonprivileged user programs, they can be interrupted by an ATTENTION at any point; thus the user can interrupt a language processor, execute a string of commands, and restart the language processing. If he modifies his source data set while in the command mode, neither LPC nor the language processor will be aware of it, and the results are unpredictable. The user may also choose to terminate one language processor and immediately call another. Before the next language processor is called, checks are made to determine if the current language processor has finished its processing. If it has not, the source data set is closed, the list data set is erased, and any virtual storage reserved for the PMD, TXT, and ISD is released.

The DEFAULT LPCXPRSS = Y option makes it possible to do several compilations without returning to the command system between compilations. Only the new module name is entered at the end of each compilation. The new module uses the same parameters as the original call to LPC. A break character or a numeric module name, inserted before the next command, terminates express mode and causes a return to the command system. DEFAULT LPCXPRSS = N must be entered after returning to the command system or the next assembly or compilation will be in express mode.

LPCMAIN (CFADA)

LPCMAIN is a nonprivileged program called via the ENTER SVC. After validating the operands required by that processor, this routine calls the requested language processor to process a source data set. At the end of processing, LPCMAIN stores the resulting object module and, if requested, prints the listing data set, which includes all output specified by the user. (See Chart DC.)

ENTRIES:

CFADA1 - FORTRAN entry
CFADA2 - assembler entry
CFADA3 - linkage editor entry
SYSASM - alternate assembler entry used by LOAD and RUN
SYSFTN - alternate FORTRAN entry used by LOAD and RUN
SYSLNK - alternate linkage editor entry used by LOAD and RUN

MODULES CALLED:

GATE (CZATC)	CZATC1 writes to SYSOUT or reads from SYSIN.
MSGWR (CZAAD)	CZAAD2 issues system messages.
FINDDS (CZAEC)	CZAEC1 searches the TDT for the JFCB for the source or listing data set.

DDEF (CZAEA)	CZAEA4 creates a JFCB for the source data set, listing data set or system macro library.
DELETE (CZAFJ)	CZAFJ deletes a catalog entry for the list data set.
FINDJFCB (CZAEB)	CZAEB1 searches the TDT for the JFCB for a given ddname.
Virtual Memory Allocation (CZCGA)	CZCGA3 (FREEMAIN) releases virtual storage obtained for PMD, TXT and ISD.
ERASE (CZAEJ)	CZAEJ7 erases a listing data set.
CATALOG (CZAEI)	CZAEI2 establishes a generation data group for a listing data set.
ABEND (CZACP)	CZACP1 terminates processing after an error is detected.
Control Dictionary Handler (CZASD)	CZASDX finds the default value for LPCXPRSS, MODREP.
Source List Handler (CZASC)	CZASC8 gets another line from the Source List.
GETLINE (CFADB)	CFADB1 modifies a source data set.

The following are sections of the language processors that may be entered:

Assember Initiation (CEVPAA), Continuation (CEVPAB), and Early End (CEVPAZ)

FORTRAN Initiation (CEKTAA), Continuation (CEKTAB), and Early End (CEKTAC)

Linkage Editor Initiation (CEYIA1), Continuation (CEYOP1), and Early End (CETEE1)

EXITS: The routine normally returns to the calling routine, via the RETURN SVC. If a system error occurs, the routine exits to ABEND.

OPERATION: This routine first collects the operands required for the specified language processor (FORTRAN compiler, linkage editor, or assembler) either by prompting the user for them or by fetching them from SYSIN. The Command Controller SCAN package (CZAAC), assembled as part of LPCMAIN with minor modifications, is used to validate the operands. If any operands are found to be invalid, the rejected operand is displayed, and the user is prompted to again enter the operand set. The routine then places the validated operands in a list, calls the language processor, and passes the list to it. LPCMAIN also calls DDEF to issue job file control blocks (JFCBs) for the source and listing data sets, and opens the source data set as well as the appropriate job library, before passing control to the language processor's initial entry. If a listing data set was actually produced, LPC determines if a generation data group LIST.symbol exists. If not, LPC creates one by issuing

```
CAT GDG = LIST.symbol,2,0,E
```

and then adding the listing data set just produced to IIST.symbol by issuing

```
CAT LIST.symbol(+1),N
```

The user is now responsible for printing listing data sets.

At the end of the source data set scan, the language processor returns control to LPCMAIN. The routine then issues any diagnostic messages produced for the last statement in the source data set. If the task is nonconversational, LPCMAIN merely closes the source data set and passes control to the language processor at its continuation entry point. For a conversational task, LPCMAIN asks the user if he wants to modify his data set.

If the modify option is selected, LPCMAIN calls the language processor early end routine to terminate the completed processing, then calls GETLINE to collect the user's modifications and place them in his source data set. Upon return, LPCMAIN calls the language processor initialization routine to restart processing. The input for this new scan is the stored source data set with its modifications included.

If the continue option is selected, LPCMAIN closes the source data set and calls the language processor at its continuation entry point.

When normal processing ends, the language processor returns control from its continuation entry point to LPCMAIN. The routine then stores the object module (if one was created) in the user's job library (or user library). Finally, LPCMAIN returns to its calling routine.

LPCMAIN permits a maximum of seven macro libraries to be passed to the assembler; this includes the system macro library and a maximum of six user macro libraries.

When a module is assembled with aliases, duplicates of which already exist in JOBLIB, the user may, at his option, print the duplicate aliases and stow the module in a new JOBLIB.

If the user enters a dsname, LPC DDEFs a new JOBLIB, stows the module, and releases the data set so the next module assembled will go into the original JOBLIB. If the user chooses to DDEF his own JOBLIB, the module will be stowed in the new JOBLIB and all other modules assembled will go into the same JOBLIB. The user may also return to Command System and release the JOBLIB containing the duplicate alias or erase the member containing the duplicate alias.

Note: Any batch assembly or compilation will automatically have the listings printed unless the user specifies LISTDS=Y (create a list data set). This is not true in conversational mode where the default option for LISTDS is Y.

ERROR CONDITIONS: The routine checks for acceptable operands and notifies the user, through MSGWR, of any errors found. For the messages issued, see System Messages.

LPCMAIN may issue these ABEND messages:

ABEND Message	Significance
GATRD RC OR FORMAT KEY	Return code from GATE is invalid or unacceptable, or keyboard/card key of record is invalid.
INVLD RC FROM LP	Invalid return code from language processor.
SCAN RC OR DELIM CODE INVLD	Invalid return code or undefined delimiter code returned from a SCAN routine.

ABNORMAL RC FROM FIND	Abnormal return code received from FIND.
INVLD RC FROM FINDDS	Invalid return code from FINDDS.
GATWR RC INVALID	Invalid return code from GATE.
MSGWR RC INVLD	Invalid return code from MSGWR.
MSNG PARAM	Required parameter missing (issued in non-conversational task only).
STOW RC ABNORMAL	Abnormal return code from STOW macro instruction.
DD RC ABNORMAL	Abnormal return code from DDEF.

If the code returned by the language processor at the end of the scan phase indicates an object module will not be produced, LPCMAIN will issue any remaining messages for the source data set. In nonconversational mode, LPCMAIN will terminate processing as described for the termination option. In conversational mode, the user is asked to terminate processing or modify his source data set; he cannot continue. Termination and modification are handled according to the previous description of those options.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Data Definition Table (CHATDT)
Data Control Block (CHADCB)
Interrupt Storage Area (CHAISA)
TABLEA (CHAAAA)

GETLINE Routine (CFADB)

This routine fetches source lines from SYSIN, or from a prestored source data set, and sends them, on request, to a language processor. It stores the lines to create a source data set if the data was not already stored, and places modification lines in the stored data set. In addition, GETLINE issues diagnostic messages stacked by PUTDIAG for the language processors and (in conversational mode) prompts for corrections. (See Chart DD.)

ENTRIES:

CFADB1 - normal entry
CFADB2 - SYNAD entry
CFADB3 - EODAD entry

MODULES CALLED: Since GETLINE is a nonprivileged program, all calls are made via ENTER SVC.

GTWAR (CZATC1)	Issues line number for next line, and also reads in that line.
GATPD (CZATC1)	Reads in next line of data set.
GTWSR (CZATC1)	Issues correction line number, and also reads in the next line.
GATWR (CZATC1)	Writes diagnostic messages.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: GETLINE has four distinct modes of operation:

1. Source data set is prestored; task is nonconversational.

2. Source data set is on nonconversational SYSIN.

3. Source data set is prestored; task is conversational.

4. Source data set is on conversational SYSIN.

For case 1, the source data set is prestored and the task is nonconversational. In this case, the source lines are read in one at a time and passed on to the language processor. No corrections are made and no messages are issued.

In case 2, the source data set is on nonconversational SYSIN. If the SYSIN is VISAM, the first seven characters of text are the line number (or key); GETLINE replaces this number with a new seven character source data set line number while creating the source data set. The eighth character is assumed to be a format indicator, and is carried with the original text.

If SYSIN is VSAM with variable-length records, it is assumed that it was created via the DATA command, and that the first character of each record is a keyboard/card-reader indicator. The indicator is left intact and a seven-character source data set line number is prefixed to the text.

In case 3, the source data set is prestored and the task is conversational. GETLINE first tests for pending messages, and if any are found they are issued immediately. After all messages are printed, a number sign (#) is issued to invite the user to enter corrections. The user either types in a modification line or presses the RETURN key to indicate that no modifications are desired. Once a correction has been made, or rejected as invalid, GETLINE requests the next correction until the user enters a null line (carriage return only).

All diagnostic printouts may be inhibited by typing an "I" immediately after the number sign (#). A "C", entered immediately after the number sign, allows diagnostic printouts but inhibits printing the number sign, and the user is not allowed to make corrections.

If no corrections were made, GETLINE gets the appropriate record from the source data set. A test is made to see whether the input line number is the same as the last line number sent to the language processor. If it is, the next record is taken from the source data set. If the two numbers are not the same, reorientation within the data set is performed.

In case 4, the source data set is on conversational SYSIN. GETLINE first checks for diagnostic messages and, if any are found, issues them. If messages are issued, the next line number for the source data set is printed to prompt the user to enter his source data line. The keyboard is unlocked so that the user may enter either the new line or a correction line. If no messages are issued, the active terminal component is read.

If the user has entered a correction line, GETLINE requests another line (after making the correction in the data set). This process continues until the user supplies a normal next line, which is put in the data set.

There are two subroutines internal to GETLINE: GVDIAG and SCNCOR. GVDIAG performs the actual issuing of diagnostic messages. It examines the count in the diagnostic message stack, and if it is nonzero, issues

the messages one at a time. SCNCOR validates and makes corrections to data set lines.

The privileged SCAN routines with slight modifications are assembled as part of GETLINE.

ERROR CONDITIONS: If called by a language processor, GETLINE returns a hexadecimal code in register 15. However, LPCMAIN receives no return code from GETLINE.

Code	Significance
00	Normal return
04	Normal return when alterations are made
08	No end card in source data set
12	Abnormal end of task

GETLINE may issue these diagnostic messages to the user:

```
CFADB023  PROCEEDING:  CORRECTION OUTSIDE DATA SET .  LINE IGNORED.
CFADB037  PROCEEDING:  LAST SOURCE LINE IS $01.  MISSING END CARD
          SUPPLIED.
CFADB378  PROCEEDING:  END OF DATA SET .  LAST LINE DELETED WAS $01.
CFADB379  PROCEEDING:  INVALID MODIFICATION REQUEST.
CFADB388  CANCELED:  NEXT LINE NUMBER EXCEEDS 7 DIGITS.
```

The routine may issue these ABEND messages:

ABEND Message	Significance
LINE NO. FROM LP INVLD - CASE 3	In case 3, line number parameter received from language processor invalid.
LINE NO. FROM LP INVLD - CASE 4	In case 4, line number parameter received from language processor invalid.
CASE SW INVLD	Indicator in PSECT of GETLINE has been set incorrectly, showing that PSECT has been erroneously destroyed.
KEY FORMAT INVLD	Keyboard/card-reader indicator of record from GATE is invalid.
TCMCOV OR TCMGRD INVLD	Conversational/nonconversational flag or record length type flag in task common is invalid.
INVLD EXIT TO SYNAD OR EODAD	Exit made to SYNAD or EODAD when reason for exit should not have occurred.
LPC SDS INVALID	A byte within common area of LPCMAIN is invalid, meaning that common has been erroneously destroyed. (Self-checking and debugging aid. Byte actually indicates whether source data set prestored or not.)
SCAN RC INVLD	Invalid return code from a SCAN routine.
MSGWR RC INVLD	Invalid return code from MSGWR.

GATE RC INVLD	Invalid return code from GATE.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Data Control Block (CHADCB)

PUTDIAG Routine (CFADC)

This routine receives diagnostic messages from the language processors and either sends the messages to SYSOUT via the GATE routine, or stacks them as output for GETLINE or LPCMAIN. (See Chart DE.)

ENTRY: CFADC1 - normal entry

MODULES CALLED:

GATWR (CZATC1) To write diagnostic messages to SYSOUT.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: PUTDIAG either stacks the diagnostics or prints them immediately. If the correction request parameter is off (as set by the language processor), the diagnostic message is issued immediately. If it is on, and there is room in the diagnostic message stack, the message is moved to the stack. If there is no room, the messages currently in the stack are immediately written out, and then the current message is stored there.

ERROR CONDITIONS: PUTDIAG may issue these ABEND messages:

ABEND Message	Significances
INVALID GATWR RC	Invalid return code from GATE.
MSG DISPOSITION BYTE INVLD	Invalid parameter from language processor, indicating whether message is to be stacked or sent.

SYSTEM CONTROL BLOCK USAGE: None.

The routines described in this section handle certain interruptions not processed by the task monitor. Those interruptions are:

1. The initial ATTENTION interruption of a conversational task.

2. Program interruptions for which no other procedure has been specified.

3. External interruptions (that is, interruptions caused by messages from one task to another).

Four routines process these interruptions:

Interruption Processor (IAIP)
: performs initialization functions for a conversational task.

Program Interruption Diagnostic Processor (DIAGNO)
: informs the user when a program interruption occurs.

External Interruption Processor (XIP/XIS)
: receives and validates intertask messages, then routes them to the specified subprocessor. This routine also handles special initiation functions for nonconversational, batch monitor, and bulk I/O tasks.

External Interruption Subprocessor (XIMS/XIES)
: writes a message on a task's SYSOUT, and also returns an error message to the sender of an invalid intertask message.

IAIP and XIP make use of the Virtual Memory Task Initialization (VMTI) routine to define system data sets and public devices during task initiation.

The resident supervisor maintains six interruption queues for each task; they are defined in location and extent by fields in the task status index (TSI). When an interruption must be brought to the attention of a task, the task interruption control (TIC) subroutine of the resident supervisor matches the interruption pending flags against the interruption mask in the task's TSI. If the interruption is not masked, TIC moves the MCB, if one exists, to the interruption storage area and and creates a software interruption by manipulation of the virtual PSW, causing entry into the task monitor.

If the interruption is to be handled by any of the routines described here, one of the interruption handling routines of the task monitor will enqueue a linkage entry in the Task Monitor scan table and exit either to the task monitor scanner or the Selective Optimal Procedure (SOP) routine.

Program interruptions for which no other processor has been defined by the user (UPI) are given to DIAGNO, which prints a message on the task's SYSOUT giving the CSECT name, a relative location number for the location beyond the interrupted instruction, and the cause of the interruption.

External interruptions associated with intertask messages are given to the external interruption processor (XIP). This processor validates the message and routes it to its subprocessor.

Because messages cause interruptions, a task may put itself into a wait state pending some external event (such as a tape being mounted), and be brought back into activity upon completion of the event by receipt of a message. The batch monitor creates TSIs for nonconversational tasks, then activates the tasks by sending messages to them. These messages contain a code that causes XIP to route them to XIIS, the task initiation subprocessor, where task initiation is completed.

ATTENTION interruptions associated with a device not currently assigned to a task are given to the initial attention interruption processor (IAIP). (ATTENTION interruptions from SYSIN devices assigned to active tasks cause the Task Monitor to call the Command Analyzer and Executor.)

Program Interruption Diagnostic Processor - DIAGNO - (CZAHA)

This routine issues a diagnostic message to SYSOUT, indicating the cause of the program interruption and where it occurred. DIAGNO services all program interruptions not treated by other processors. (See Chart DF.)

ENTRY: CZAHA1 - normal entry

MODULES CALLED:

MAPSEARCH (CZCCQ) Gets CSECT name and displacement.

MSGWR (CZAAD2) Issues messages to SYSOUT.

INTERVENE (CZAMZ3) Processes program interventions.

EXITS: If the task was conversational, DIAGNO sets up a linkage entry to the command analyzer and executor (via QLE), and then returns control to the task monitor, via the RETURN macro instruction. If the task was nonconversational, DIAGNO exits to ABEND.

OPERATION: The routine first fetches the VPSW, then uses the VPSW interruption code to find the corresponding diagnostic message. Next, it calls MAPSEARCH to get the name and base address of the CSECT in which the interruption occurred. This base address, together with the interruption address given in the VPSW, is used to develop the location of the interruption. Using MSGWR, the routine now issues to SYSOUT the diagnostic message, the old VPSW, the CSECT name, and the displacement of the interruption. The last two of these are supplied by DIAGNO in this form, provided the CSECT is found valid:

```
INTERRUPT OCCURRED IN CSECT xxx
WITH DISPLACEMENT OF yyy
FROM BEGINNING OF CSECT.
```

where xxx is the CSECT name and yyy is the displacement value.

DIAGNO may issue the following diagnostic messages:

Message Number	Message	Significance
E000	OPERATION CODE INTERRUPT. PSW = ___.	Attempt to execute an unassigned operation code. PSW contains location of interruption.

E001	PRIVILEGED OPERATION INTERRUPT. PSW = ___.	Attempt made to execute op code not permitted for user's privilege. PSW shows location of interruption and contains interruption code and user's privilege.
E002	EXECUTION INTERRUPT, PSW = _____.	When subject instruction of an execute instruction is another execute instruction. PSW shows location of interruption and contains interruption code and user's privilege.
E003	PROTECTION INTERRUPT. PSW = _____.	Tries to access privilege storage at location indicated in PSW.
E004	ADDRESSING INTERRUPT, PSW = _____.	User attempted to access address that did not exist. PSW contains location of interruption.
E005	SPECIFICATION INTERRUPT. PSW = _____.	User made syntactical error in an instruction. PSW contains location of interruption.
E006	DATA INTERRUPT. PSW = _____.	While accessing I/O device an error was encountered. PSW contains location of interruption.
E007	FIXED POINT OVERFLOW INTERRUPT. PSW = ___.	While doing fixed-point multiplication or division, register overflowed. PSW contains location of interruption.
E008	FIXED POINT DIVIDE INTERRUPT. PSW = ---.	Register overflowed while performing fixed-point division.
E009	DECIMAL OVERFLOW INTERRUPT. PSW = ___.	Destination field too small to contain the result field in a decimal operation. PSW contains location of interruption.
E010	NONPRIVILEGED PROGRAM ISSUED IOCAL, PGOUT. PSW = _____.	User tried to do privileged I/O paging operation. PSW contains location of interruption.
E011	IOPCB OR IORCB PAGE LIST TOO LONG. PSW = _____.	I/O control block too long for I/O program execution. PSW contains location of interruption.
E012	SPECIFIED ADDRESS IS NOT IN USERS VIRTUAL MEMORY (IOCAL PGOUT). PSW = _____.	Address attempted to access in PSW does not exist in user's virtual storage.

E013	PROGRAM HAS NO I/O DEVICES ASSIGNED TO IT (IOCAL PGOUT). PSW = _____.	Paging cannot be done because no device can be assigned for that purpose. PSW contains location of interruption.
E014	IORCB SIZE OF ZERO. PSW = _____.	I/O control block is nonexistent for I/O program. PSW contains location of program interruption.
E015	TSI SERVICE CALL INTERRUPT COUNTER OVERFLOW. PSW = _____.	Too many service call interruptions. PSW contains location of interruption.
E016	TSI EXTERNAL INTERRUPT COUNTER OVERFLOW. PSW = _____.	Too many external interruptions. PSW contains location of interruption.
E017	TSI ASYNCHRONOUS INTERRUPT COUNTER OVERFLOW. PSW = _____.	Too many asynchronous interruptions. PSW contains location of interruption.
E018	TSI INTERRUPT COUNTER OVERFLOW. PSW = _____.	Too many interruptions. PSW contains location of interruption.
E019	TSI INPUT/OUTPUT INTERRUPT COUNTER OVERFLOW. PSW = _____.	Too many I/O interruptions. PSW contains location of interruption.
E021	ADDRESS OF UNASSIGNED PAGE GIVEN TO CKCLS SVC PROCESSOR. PSW = _____.	A page that was once in user's virtual storage and is no longer, has been given to Check Class (CKCLS) SVC.
E022	ILLEGAL CODE GIVEN TO SETUP/XTRCT SVC PROCESSOR. PSW = _____.	Illegal form of SETUP/ XTRCT SVC macro user.
E023	AWAIT SVC NOT EXECUTED REMOTELY OR NOT LAST HALFWORD OF AN ECB. PSW = _____.	AWAIT called directly and not via an EXECUTE instruction.
E024	INVALID SHARED PAGE TABLE NUMBER TO ADSPG SVC PROCESSOR. PSW = _____.	An invalid shared page given to ADSPG.
E025	SOFTWARE HAS DETECTED A POSSIBLE HARDWARE MALFUNCTION. PSW = _____.	Simulated machine check.
E026	USER'S TASK NOT OF SUFFICIENT PRIORITY TO ISSUE SVC. PSW = _____.	User does not have correct privilege to issue SVC.
E027	SVC NOT ON WORD BOUNDARY. PSW = _____.	SVC must be on word boundary otherwise an invalid call.

E028	COUNT OF EXTERNAL ADDRESS IS ZERO. PSW = _____.	External address count illegal.
E029	ALL PARAMETERS ARE NOT IN ONE PAGE. PSW = _____.	All SVC calls are not within page boundary.
E030	NO ASYNCHRONOUS ERROR ROUTINE DEFINED FOR DEVICE WITH ERROR. PSW = _____.	Device malfunction and no error routine defined to handle malfunction.
E031	ASYNCHRONOUS INTERRUPT RECEIVED BUT NO DE AVAILABLE FOR DEVICE. PSW = _____.	No routine available to handle asynchronous interruption.
E032	PGOUT REQUEST FOR ZERO PAGES. PSW = _____.	Invalid request for number of output pages.
E033	SETTR NOT ACCEPTED BECAUSE SYSTEM LIMIT REACHED IN TABLE. PSW = _____.	No more main storage available for new entries.
E034	PROGRAM INTERRUPT RECEIVED WHILE IN TYPE III LINKAGE. PSW = _____.	Program interruption receive while going to or from privileged to nonprivileged status.
E035	SVC INTERRUPT RECEIVED WHILE IN TYPE III LINKAGE. PSW = _____.	No SVC permitted while in nonprivileged status.
E036	ATTEMPT TO ADD MORE THAN 256 SHARED PAGES TO A SEGMENT. PSW = _____.	Interruption occurs when a segment's maximum limit of pages is exceeded. PSW contains location of interruption.
E037	VSEND MESSAGE IS TOO LONG OR EXTENDS OVER PAGE BOUNDARY. PSW = _____.	MCB extends over 1920 or a page boundary.
E038	REQUEST TO DELETE PAGE FROM SEGMENT NOT PREVIOUSLY ASASSIGNED. PSW = _____.	Request to delete a page from an invalid segment.
E039	REQUEST TO DELETE PAGE NOT PREVIOUSLY ASSIGNED. PSW = _____.	Request deletion of a page that does not exist.
E040	MESSAGE CODE INTERRUPT NOT ASSIGNED PRESENTLY. PSW = _____.	Program interruption receive in PSW is undefined at present in this system.
E041	RELOCATION PAGE - IN ERROR (DEVICE DEFECTIVE) - MOVABLE VOLUME. PSW = _____.	Attempt to output a page on a defective device. Volume can be relocated.

E042	RELOCATION PAGE - IN ERROR (MEDIUM DEFECTIVE). PSW = ___.	Attempt to output a page but mechanism to output device is defective.
E043	IOCAL PAGE - IN ERROR (DEVICE DEFECTIVE) PERMANENT VOLUME. PSW = _____.	Attempt to output a page on a defective device. Volume is permanent and cannot be relocated.
E044	IOCAL PAGE - IN ERROR (DEVICE DEFECTIVE) - MOVABLE VOLUME. PSW = _____.	Attempt to output a page but device defective. The volume however can be relocated.
E045	IOCAL PAGE - IN ERROR (MEDIUM DEFECTIVE). PSW = _____.	Attempt to output a page but the mechanism is defective.
E046	OPERATION TASK HAS BEEN REINITIALIZED. PSW = _____.	Operator task has been reinitialized after previous interruption.
E047	UNAUTHORIZED USE OF TSSS SVC. PSW = _____.	An unauthorized SVC (codes 64-95) has been issued by a nonprivileged program. These codes are reserved for use by Time Sharing Support System (TSSS).
E01A	GQE TYPE CODE IS IN ERROR. PSW = _____.	Used wrong general queue entry code for Task Status Index.
E01B	IORCB SIZE EXCEEDS 1920 BYTES. PSW = _____.	I/O record control block exceeds maximum limit.
E01C	IORCB OR IOPCB CROSSES A PAGE BOUNDARY. PSW = ____.	I/O record control block or I/O page control block greater than page boundary.
E01D	DEVICE NOT ASSIGNED TO TASK (IOCAL PGOUT). PSW = _____.	Attempt to place pages on device not assigned to task.
E01E	IOCAL OR PGOUT SVC PAGE ADDRESS DOES NOT EXIST IN VIRTUAL MEMORY. PSW = _____.	Attempt to access a page that does not exist.
E01F	IOCAL OR PGOUT SVC PAGE IS NOT IN CORE. PSW = _____.	Page not in main storage.
E02A	COUNT EXCEEDS 991, BIT STRING FLAG SET FOR SETXP. PSW = S.	Invalid code used to call SETXP. PSW contains location of interruption.
E02B	PAGE UNASSIGNED. PSW = _____.	Attempted to access unassigned page.
E02C	COUNT EXCEEDS 1022, BIT STRING FLAG NOT SET FOR SETXP. PSW = _____.	Invalid code used to call SETXP. PSW contains location of interruption.

E02D	ENTER SVC ISSUED TO INTERRUPTABLE ROUTINE WHILE TYPE III LINKAGE IN EFFECT AND P1 FLAG ON. PSW = _____.	Attempt to start a recursive situation in task monitor. PSW contains location of interruption.
E02E	ENTER SVC ISSUED WITH INVALID ENTER CODE___ OVER 22K OR UNASSIGNED. PSW = _____.	Attempt to use ENTER SVC (to call a routine) with wrong code. PSW contains location of interruption.
E02F	SVC ISSUED IN NON-PRIVILEGED STATE AND NO INTERRUPT ROUTINE SPECIFIED. PSW = ____.	No routine exists for particular SVC executed at location contained in PSW.
E03A	INVALID SEGMENT NUMBER GIVEN TO ADSPG PROCESSOR. PSW = ___.	Illegal segment number given to ADSPG (address page) processor.
E03B	ILLEGAL CODE GIVEN TO SETSYS/XTRSYS SVC PROCESSOR. PSW = ___.	Invalid code received by SETSYS/XTRSYS SVC. PSW contains location of interruption.
E03C	ILLEGAL CODE GIVEN TO SETXTS/XTRXTS SVC PROCESSOR. PSW = _____.	Invalid code received by SETXTS/XTRXTS SVC. PSW shows location of interruption.
E03D	UNSUCCESSFUL DEQUEUE I/O REQUEST. PSW = _____.	Unsuccessful attempt to dequeue I/O. PSW shows location of interruption.
E03E	DRUM FLAG ILLEGALLY ON. PSW = _____.	Drum access flag on illegally. PSW shows location of interruption.
E03F	RELOCATION PAGE - IN ERROR (DEVICE DEFECTIVE) - PERMANENT VOLUME. PSW = _____.	Page cannot be relocated on device because device is defective. Volume is permanent and cannot be relocated. PSW shows location of interruption.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Task Dictionary Table (CHATDY)

Initial Attention Interruption Processor - IAIP - (CZAHB)

This routine processes the initial attention interruption from a conversational task by providing certain initialization functions in virtual storage for that task. (See Chart DG.)

ENTRY: CZAHB1 - normal entry

MODULES CALLED:

VMTI (CZAAF1) To define system data sets and indicate devices containing public volumes.

EXITS: The routine exits through NTCTMR via the RETURN macro instruction, except in the case of ABEND where a normal return is made.

OPERATION: A flag is set in task common to indicate that the new task is conversational, and Virtual Memory Task Initiation (VMTI) is called.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE:

Task Common (CHATCM)
Interruption Storage Area (CHAISA)

External Interruption Processor - (XIP/XIIS) - (CZAHC)

XIP receives intertask messages, checks their validity, copies them into virtual storage, and enqueues linkage to the specified subprocessor. It also processes the shutdown message.

XIIS handles task initiation for nonconversational, batch monitor and bulk I/O tasks. (See Chart DH.)

ENTRIES:

CZAHC1 - main entry point
CZAHC3 - entry to initiate nonconversational user's tasks
CZAHC4 - entry to initialize a bulk I/O task or the task monitor

MODULES CALLED:

QLE (CZCJQS)	Sets up linkage.
GETMAIN (CZCGA2)	Allocates space for MCB.
FINDJFCB (CZAEB1)	Finds a JFCB.
FREEMAIN (CZCGA3)	Frees storage used by MCB.
SETUP (CZAH2)	Places new SYSIN device address in TSI to indicate a new task has been logged on.
VMTI (CZAAF1)	Defines system data sets and indicates devices containing public volumes.

EXITS: The routine normally exits to the task monitor, via the RETURN macro instruction. If a message was erroneously sent to a particular task, the routine will enqueue linkage to the error subprocessor (XIES), and then exit to the task monitor via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: At entry CZAHC1, XIP tests if the message is a reply. If it is an awaited reply, XIP moves the message to the designated reply area, and returns control to the task monitor.

If the message is not a reply, XIP tests for a code indicating that data management has caused the interrupt to take the task out of the wait state. If such a code is found, control is returned to the Task Monitor with a return code of zero.

For all other messages, GETMAIN allocates space into which XIP can move the message control block from the IORCB field. The code in the message is used as an index to the required subprocessor's entry in a table and linkage to it is set up via QLE and control returned to the Task Monitor.

The following hexadecimal MCB message codes are recognized as valid codes:

Code	Indicates	Module
00	Message	CZAHD1
01	Batch Monitor to task being canceled	
02	Batch Monitor to third-level task initiation	CZABAB
03	Error (invalid code)	CZAHD2
04	MOCP (Main Operator Control Program)	CZACA1
06	Activate Batch Monitor	CZABA1
07	Initiate PRINT	CZABG1
09	Initiate WT	CZAB11
0A	Initiate PUNCH	CZABH1
0B	Initiate RT	CZABF1
12	AWAIT for device allocation	
16	Device allocation	CZABAA
50	EXECUTE to Batch Monitor	CZABA2
52	CANCEL to Batch Monitor	CZABA3
54	BULKIO to Batch Monitor	CZABA4
56	BACK to Batch Monitor	CZABA5
5A	LOGOFF to Batch Monitor	CZABA7
5B	BULKIO Init to Batch Monitor	CZASW1
5D	Shutdown to Batch Monitor	CZABA9
5E	ABEND	CZACR1
5F	LOGOFF (from FORCE)	CZAFN2
60	MOHR (Main Operator's Housekeeping)	CZACB3

If the message is the shutdown message, linkage is also enqueued to LOGOFF.

Entry CZAHC3 of the XIIS subprocessor is called to initiate nonconversational user tasks, while entry CZAHC4 is called for initialization of a bulk I/O task or the Batch Monitor.

For both entries, the message control block contains information needed by XIIS. FREEMAIN is called to clear the MCB area before returning to the Task Monitor.

For the Batch Monitor or a bulk I/O task, XIIS moves the parameters of the specified routine from the MCB into PARIN. The batch sequence number is put into task common. A linkage entry to the routine just loaded is created and FREEMAIN is called; control is then returned to the Task Monitor.

When the batch monitor initializes a nonconversational task, it sends a message to the task with a code for entry CZAHC3. FINDJFCB is called to find the JFCB of the SYSIN entered in the TSI. The flag in task common is set to nonconversational. The userid is placed in task common. FREEMAIN is called, and control is returned to the Task Monitor.

ERROR CONDITIONS: XIP will determine from the following conditions if a message was erroneously sent to a task:

- Sender's reply not the awaited one (either taskid or message number incorrect).
- User not awaiting reply when a reply is received.
- MEB not on word boundary.
- Shutdown message has incorrect taskid.

The routine will then call GETMAIN, move the message, and set up linkage to the error subprocessor (XIES) before returning to the task monitor.

The routine may issue this system error message:

SYSER Code	Severity	ABEND Message	Explanation
050702701	Minor	MCB LENGTH EXCEEDS MAXIMUM LIMIT	MCB length exceeds maximum limit

SYSTEM CONTROL BLOCK USAGE:

Interrupt Storage Area (CHAISA)
Task Data Definition Table (CHATDT)
Task Common (CHATCM)
Message Control Block (CHAMCB)
Message Event Block (CHAMEB)
Batch Work Queue (CHABWQ)

External Interruption Subprocessor - XIMS/XIES - (CZAHD)

This routine issues, to a task's SYSOUT, all messages sent to that task. The routine also handles any message that is erroneously sent to a task. (See Chart DI.)

ENTRIES:

CZAHD1 - entry point for the external interruption processor (XIP) to issue message contained in a specified message control block (MCB)
CZAHD2 - entry point for XIP to issue message that was erroneously sent to a particular task
CZAHD3 - entry point for task monitor to process a message that was unexpectedly sent

MODULES CALLED:

VSEND	Sends intertask messages.
GATWR (CZATC1)	Writes messages to task's SYSOUT.
FREEMAIN (CZCGA3)	Releases an area of virtual storage.
User Prompter (CZATJ1)	Issues diagnostic message to the user.

EXITS: The routine exits to the calling routine, via the RETURN macro instruction.

OPERATION: The routine is called (at CZAHD1) by the external interrupt processor (XIP) to issue a message contained in a specific message control block (MCB). The message text and sender's taskid are extracted from the MCB and placed in the parameter list for GATWR.

The message code field in the MCB is checked to determine if the message was valid (indicated by a code of 0). The code is then extracted from the MCB, converted to EBCDIC, and inserted in the message. If the message was invalid (any code other than 0), the code is still placed in the message, but the following is prefixed to the message:

THE FOLLOWING INVALID MESSAGE WAS RECEIVED BY YOUR TASK.

The total length of the message is calculated and placed in the GATWR parameter list GATWR then issues the message to the task's SYSOUT. After the message has been issued, FREEMAIN is called to release the MCB virtual storage area that was used during processing.

At CZAHD2, this routine is called by XIP to process a message that was erroneously sent to a particular task. The taskids of the sender and receiver of the message are extracted from the MCB and transposed so that the original sender will now receive the message, prefixed with the following:

THE MESSAGE YOU HAVE SENT HAS AN INVALID CODE.

Then a branch to CZAHD1 is taken to issue the erroneous message.

CZAHD3 is called by the task monitor. Processing at this entry is the same as at CZAHD2 except that the following message is sent:

THE MESSAGE YOU HAVE SENT HAS AN UNEXPECTED CODE.

ERROR CONDITIONS: None.

SYSTEM CONTROL BLOCK USAGE: Message Control Block (CHAMCB)

Virtual Memory Task Initiation - VMTI - (CZAAF)

This routine resides in initial virtual storage and performs certain functions required to initiate a task. These functions include creating job file control blocks (JFCBs) for system data sets, opening the system library, and adding to the task symbolic device list (TSDL) all those devices on which public volumes reside. (See Chart DJ.)

ENTRY: CZAAF1 - normal entry

MODULES CALLED:

VMTI-2 (CZATD)	Logs on task.
TIME (CZAVB)	Initializes task timer.
DDEF (CZAEA5)	Creates JFCBs for system data sets.
FINDJFCB (CZAEB1)	Finds SYSLIB JFCB.
ADDEV	Adds symbolic device address to TSDL.
LIB MAINTENANCE (CZCDHC)	Opens SYSLIB DCB.

OPEN (CZCLA) Opens SYSCAT DCBs.

EXITS: The routine normally returns to the calling routine, via the RETURN macro instruction. If a system error occurs, the routine exits to SYSER.

OPERATION: Initial virtual storage contains a task data definition table (TDT) with JFCBs for the SYSCAT (SYSSVCT and SYSUCAT), SYSIN, and SYSOUT data sets. The TDT header is set up so that the appropriate fields point to the last JFCB and the first free storage area; the pointer to the JFCB in the program library list points to zero; the temporary tabulation pointer also points to zero.

VMTI is called by either IAIP (for a conversational task) or XIP (for a nonconversational task) to initiate a new task.

The symbolic device allocation table (SDAT) is scanned to find all those devices that carry public volumes. Since all SDAT entries for such devices are chained together, this involves finding the beginning of the chain in the SDAT header and following it through the table. As each public device is found, its address is added to the TSDL using the ADDEV SVC. The system catalogs SYSSVCT and SYSUCAT are then opened for update.

Successive calls to DDEF are made to create and fill in the JFCBs for the user catalog (SYSUCAT), the system library, the user table, the system macro library and the macro library index. The timer for the task is initialized and control is passed to VMTI-2 to logon the task. On return, control is passed to the calling routine.

ERROR CONDITIONS: The routine may issue these system error messages:

SYSER Code	Severity	ABEND Message	Significance
050106102	Major	None	Error return code received from DDEF.
050106103	Major	None	No address returned by CZAER for SYSLIB JFCB.

SYSTEM CONTROL BLOCK USAGE:

Interrupt Storage Area (CHAISA)
Symbolic Device Allocation Table (CHASDA)
System Common (CHASCM)
Task Common (CHATLM)
Task Data Definition Table (CHATDT)

Virtual Memory Task Initiation II - VMTI II (CZATD)

This routine provides the task initialization interface between command system I and command system II. This includes opening the system message file (SYSMLF), opening the SYSIN data set for nonconversational updating of the user table, logging on the user, and in the case of operator task, calling the main operator housekeeping routine. (See Chart DK.)

ENTRIES:

CZATD1 - normal entry point for all tasks in the system
CZATD2 - special entry point for all bulk I/O tasks to call logon
CZATD4 - special entry point for Express Batch

MODULES CALLED:

RELEASE (CZAFJ)	Releases the SYSIN JFCB for nonconversational task.
RCR-OPEN	Performs OPEN for first express batch subtask for nonconversational tasks.
FINDJFCB (CZAEB)	Finds the SYSIN and SYSOUT JFCBs.
GATEOPEN (CZBTB4)	Opens the SYSIN DCB for nonconversational tasks.
GATE (CZATC)	Reads SYSIN.
LOGON (CZAFM)	Logs user on.
LOGON2 (CZCBTB)	Completes LOGON initialization functions.
DDEF (CZAEA)	Builds USERLIB JFCB for abended task.
MOHR (CZACB)	Initializes operator task.
QLE (CZCJQ)	Queues a linkage entry for the Command Analyzer and Executor.
WTO (CZABQ)	Informs operator that EXPRESS card not followed by LOGON (nonconversational).
OPEN (CZCLA)	Opens system message file (SYSMLF).

EXITS: The routine normally returns to the calling routine via the RETURN macro instruction. If a system error occurs, the routine exits to ABEND.

OPERATION: All tasks in the system enter at CZATD1 where the system message file (SYSMLF) is opened, and a FIND is issued to position a pointer at that particular VPAM member.

A check is then made to see if this task is the Batch Monitor or a bulk I/O task. In the case of the batch monitor or a bulk I/O task, a return is made to the calling routine.

In all other tasks, a check is made to see if the task is conversational. If it is, the conversational flag is turned on in New Task Common. If the task is nonconversational, the existing SYSIN JFCB is released, and a call is made to DDEF to construct a new SYSIN JFCB for the SYSIN data set; the conversational flag in New Task Common is set to zero. At this point Gate Open (CZBTB4) is called to open the SYSIN data set for a nonconversational task. The task ID is then extracted from the TSI and stored in Task Common. It is then examined to see if the task is the operator's task. If it is the operator task, the address of the user table DCB is picked up and the SYNAD and EODAD R-cons and V-cons are changed to point to the proper points in CZATD.

The user table data set is then opened for updating. A SETL is issued to place the pointer at the beginning of the data set. Each record is then retrieved via the VISAM GET macro instruction, and the activity, task count flag and RCR accounting fields are set to zero. Each record is then written back into the user table data set. When EODAD is reached, the data set is closed.

All tasks are again checked to see if they are nonconversational. If the task is nonconversational, a GATRD obtains the first record of the SYSIN data set. A check is made for the LOGON verb; if it is not in the record, a check is made for the 'EXPRESS' verb. If it is not an 'EXPRESS' record, the subtask is abnormally terminated.

If the LOGON verb is found, the TCMEXP1 (Express Batch mode) switch is examined; if it is off, processing continues as described in the next paragraph. At this point (CZATD4) all jobs in the Express Batch stream (with the exception of the first job which begins at CZATD1) begin their LOGON sequence. A call is made to the TIME command module (CZAVB4) to reinitialize the clock(s) as required. If the EXPRESS verb is found, the TCMEXP1 (Express Batch mode) switch is set on and a scan function is performed which establishes a pointer to the time parameter, bypassing the keyword, TIME=, when present. A call is then made to the #timE routine (CZCAVB4) to initialize System Clocks 13 and/or 14.

When return is made to VMTI-2, an additional GATRD is performed for the next record in SYSIN. These GATRDs are performed until a LOGON is encountered. If a LOGON is not found immediately after 'EXPRESS', a warning message is issued to the operator (in building an EXPRESS SYSIN data set, BULKIO ensures that an 'EXPRESS' card is followed by a 'LOGON' card); then additional GATRDs are performed. Once a LOGON is found, the clock(s) are reinitialized and processing continues.

In both conversational and nonconversational tasks, LOGON (CZAFM) is called. If task is conversational, information is extracted from the TCT slot for this user, and an SDAT entry is built for this device with pertinent information set. The device type found in the TCT is stored in GATE's PSECT. It is then translated to a model code which is used to determine the maximum device line length. This length is also stored in GATE. GATE uses this information to break up long output lines into segments that do not exceed the device line capacity and to determine the translation required.

An MCAST macro instruction allows a nonprivileged program to specify an input translation. In some application programs the terminal type must be known to perform the appropriate MCAST. The terminal type thus must be made available to nonprivileged programs. To do this, CZATD now moves the terminal type from the TCT (TCTDTY) to a field indicating terminal type in the ISA (ISADTY). ISADTY has the same code indicators as TCTDTY.

At this point a second entry point (CZATD2) is provided for ABEND and BULKIO to allow them to go to LOGON. A call to LOGON2 is then made for all tasks. When LOGON2 returns, a check is made for the first express batch subtask. If it is the first subtask, a call is made to RCR OPEN with the userid of SYSOPER0. Upon completion of LOGON2, another check is made for the operator task. If this is the operator task, the main operator housekeeping (MOHR) routine is called. When MOHR has finished, CZATD returns to the calling routine. Also, the completion of LOGON2 in bulk I/O tasks and abended tasks causes a return to the calling routine.

In all other tasks in the system, except Express Batch, a call is made to the Task Monitor to QLE to the Command Analyzer and Executor. The task initiation flag in the ISA is turned off; VMTI-2 returns to its calling routine.

For an Express Batch task, the source list and dictionary are reinitialized for the subtask via a call to LOGON2 (CZBTBB) and a return is made by way of the Task Monitor.

ERROR CONDITIONS: The following error conditions will result in an ABEND:

1. An attempt to FIND the system message file (SYSMLF) failed.

2. The absence of the LOGON verb on the first SYSIN record of a nonconversational task.

3. An attempt to DDEF a SYSIN data set for a nonconversational task failed.

4. An attempt to update a user table entry failed.

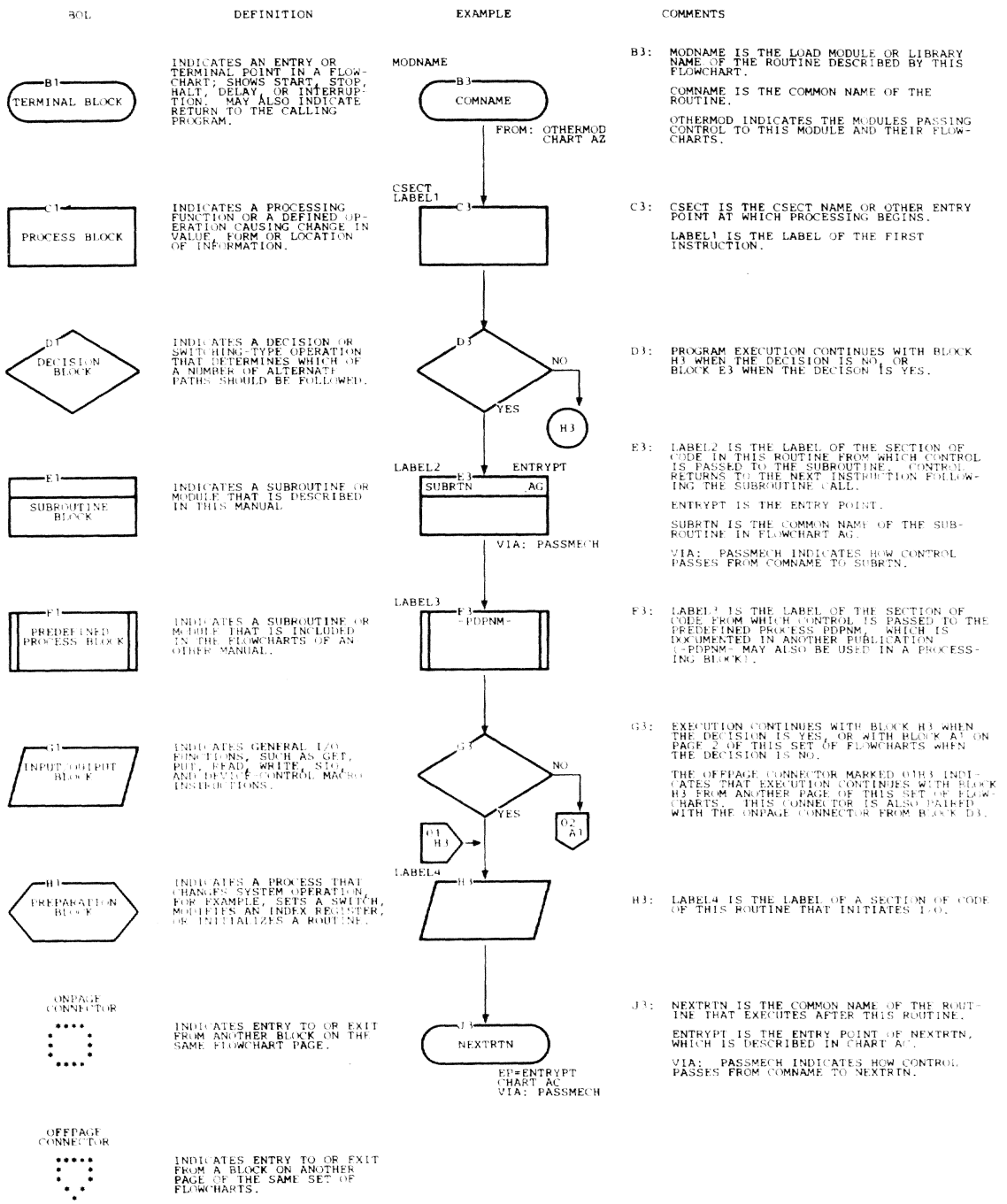
5. Model code in SDA is not 1-4.

6. RCR open for SYSOPER0 failed in Express Batch.

SYSTEM CONTROL BLOCK USAGE:

Data Control Block (CHADCB)
Combined Dictionary Header (CHADCT)
Interrupt Storage Area (CHAISA)
Message Control Block (CHAMCB)
New Task Common (CHANTC)
Symbolic Device Allocation Table (CHASDA)
Task Common (CHATCM)
Task Data Definition Table (CHATDT)
User Table (CHAUSE)
Terminal Control Table (CHATCT)

The flowcharts in this manual have been produced by the IBM System/360 Flowchart program, using ANSI symbols. These descriptions of the ANSI symbols and the System/360 Flowchart conventions will simplify interpretation of the flowcharts in this manual:



Program Logic Manual

GY28-2013-6

Command System

Flowcharts on pages 207-552 were not scanned.

This appendix can be used to find the command system modules that use a specific control block.

Control Block	Modules Using Control Block		
AAA	User Prompter	Attention Handler	User Control
ACT	ABEND		
AIR	ABEND	PAIR	RPS/CVV
AUL	ABEND RET	LOGOFF USAGE	LOGON
BPK	CA&E	User Control	
BWQ	EXECUTE	XIP/XIIS	
CCC	CATALOG EXECUTE RET UPDTUSER	DDEF RPS/CVV VAM Tape	EVV POD? RELEASE
CTT	GATE		
CVF	User Prompter	LOGON2	
DCB	GATE CONTEXT EXCERPT DATA LINE NUMBER STET BACK DATA LINE? MODIFY Procedure Expander UPDTUSER VSS VMTI-2	User Prompter CORRECT EXCISE LIST Profile Handler UPDATE CDD ERASE/DELETE LOGON PERMIT QUIT USAGE LPCMAIN PROCDEF	Text Editor Controller Edit Initialization INSERT LOCATE REGION ABEND CDS JOIN LOGON2 POD? RPS/CVV VAM Tape GETLINE User Control
DCT	Profile Handler PROCDEF	LOGON2	VMTI-2
DEB	ABEND	RELEASE	
DEC	GATE	ABEND	
DEN	CA&E PROCDEF	LOGON2	Procedure Expander
DHD	ABEND RELEASE	LOGOFF VAM Tape	POD?
DSC	DSS?/PC?		

Control Block	Modules Using Control Block		
DSE	CONVERT	DSS?/PC?	EVV
	RET	RPS/CVV	UPDTUSER
	VAM Tape		
DSF	CONVERT	RPS/CVV	
DSV	CONVERT		
ISA	CA&E Attention Handler	GATE	User Prompter
	CATALOG	ABEND	CDD
	ERASE/DELETE	DSS?/PC?	DDEF
	LOGON	JOIN	LOGOFF
	RPS/CVV	PERMIT	QUIT
	User Control	SHARE	UPDTUSER
	VMTI-2	VAM Tape	XIP/XIIS
		KEYWORD	VMTI
ITB	ABEND		
LIM	LOGON2		
MCB	ABEND	CANCEL	EXECUTE
	LOGON2	XIP/XIIS	XIMS/XIES
	VMTI-2		
MEB	ABEND	CANCEL	EXECUTE
	XIP/XIIS		
MSG	EXECUTE		
NTC	CA&E Attention Handler	GATE	User Prompter
	Profile Handler	Source List Handler	DATA LINE
	Procedure Expander	BACK	LOGON2
		User Control	LPCMAIN
		VMTI-2	CONTEXT
			Edit Initialization
			LIST
			LOCATE
			MATCH
			ABEND
			LOGON
			PROCDEF
PCT	CA&E Attention Handler	GATE	User Prompter
	LOGON2	DATA LINE	Profile Handler
	User Control	MCAST	Procedure Expander
PFL	User Profile	LOGON2	
POD	POD?		
POE	POD?		
POM	POD?		
PVT	DSS?/PC?	RET	RPS/CVV
	UPDTUSER		

Control Block	Modules Using Control Block		
RHD	ABEND	LOGOFF	POD?
	RELEASE	VAM Tape	LIST
RQU	RELEASE		
SAR	ABEND	FLOW	VMTI
SCM	ABEND	DDEF	JOIN
	LOGON	RET	RPS/CVV
	SECURE	TIME	UPDTUSER
SDA	CATALOG	DDEF	EVV
	LOGOFF	LOGON2	RPS/CVV
	RELEASE	VAM Tape	IAIP
	VMTI	VMTI-2	ABEND
SDM	LOGON	USAGE	ABEND
SDS	ABEND		
SLH	CA&E Procedure Expander	Source List Handler	IF String Comparison
		User Control	
SLM	CA&E Procedure Expander	Source List Handler	IF String Comparison
SLP	CA&E Procedure Expander	Source List Handler	IF String Comparison
		User Control	
STK	User Control		
TBC	DSS?/PC?		
TBD	DSS?/PC?	QUIT	ERASE/DELETE
	UPDTUSER		
TBS	DSS?/PC?		
TCM	GATE	SCAN	Source List Handler
	ABEND	BACK	CDD
	CANCEL	CATALOG	CDS
	DATA	DSS?/PC?	DDEF
	ERASE/DELETE	EXECUTE	FINDDS
	FINDJFCB	JOIN	LINE?
	LOGOFF	LOGON	MODIFY
	PERMIT	POD?	QUIT
	RET	RPS/CVV	RELEASE
	SECURE	SHARE	SYSXPAT
	TIME	UPDTUSER	USAGE
	VAM Tape	VSS	LPCMAIN
	GETLINE	DIAGNO	IAIP
	XIP/XIIS	VMTI-2	
TCT	GATE	CHGPASS	LIST
			CHGPASS
			LOGON2
			VMTI

Control Block	Modules Using Control Block		
TDH	LOGON		
TDT	Text Editor Controller	EXCERPT	REGION
	CDD	ABEND	BACK
	DATA	CATALOG	CDS
	FINDDS	ERASE/DELETE	EVV
	LOGON	FINDJFCB	LOGOFF
	POD?	LOGON2	MODIFY
	RPS/CVV	QUIT	RET
	VAM Tape	RELEASE	USAGE
	XIP/XIIS	LPCMAIN	IAIP
		VMTI-2	PROCDEF
			VMTI
TDY	POD?	DIAGNO	ABEND
TRN	Text Editor Controller	INSERT	REVISE
	STET	CONTEXT	DATA LINE
	TRIN	CORRECT	LIST
	Transaction Table Updater	Edit Initialization	LOCATE
	UPDATE	EXCERPT	NUMBER
	PROCDEF	EXCISE	REGION
		User Control	
TSI	BACK	LOGON	USAGE
	ABEND		
USE	GATE	ABEND	BACK
	JOIN	LOGOFF	LOGON
	QUIT	RET	UPDTUSER
	USAGE	VSS	VMTI-2
	CHGPASS		

```
ABEND    0000000 BUILTIN ABEND    ,CZASB66
ABENDREG0000000 BUILTIN ABENDREG,CZACP22
ASM      0000000 BUILTIN ASM      ,CFADA21
ASNBD    0000000 PROCDEF ASNBD
ASNBD    0000100 PARAM $1,$2,$3,$4,$5,$6,$7,$8,$9,$1X,$2X,$3X,$4X,$5X,$6X,$7X
ASNBD    0000200 SYSXPAT ASNBD
ASNBD    0000300 $1,$2,$3,$4,$5,$6,$7,$8,$9,$1X,$2X,$3X,$4X,$5X,$6X,$7X
AT       0000000 BUILTIN AT       ,CZAMF11
BACK     0000000 BUILTIN BACK     ,CZABC11
BCST     0000000 PROCDEF BCST
BCST     0000100 PARAM TEXT
BCST     0000200 SYSXPAT BCST
BCST     0000300 TEXT
BRANCH   0000000 BUILTIN BRANCH   ,CZAMB11
BUILTIN  0000000 BUILTIN BUILTIN  ,CZATP11
C        0000000 PROCDEF C
C        0000100 PARAM ALPHABET=$1
C        0000200 IF '$1'='2';DEFAULT ALPHABET=3
C        0000300 IF '$1'='1';DEFAULT ALPHABET=4
C        0000400 DEFAULT SYSIN=C
CA       0000000 PROCDEF CA
CA       0000100 DEFAULT SYSIN=C,ALPHABET=3
CALL     0000000 BUILTIN CALL     ,CZAMG11
CANCEL   0000000 PROCDEF CANCEL
CANCEL   0000100 PARAM BSN
CANCEL   0000200 SYSXPAT CANCEL
CANCEL   0000300 BSN
CATALOG  0000000 BUILTIN CATALOG  ,CZAEI11
CB       0000000 PROCDEF CB
CB       0000100 DEFAULT SYSIN=C,ALPHABET=4
CDD      0000000 PROCDEF CDD
CDD      0000100 PARAM DSNAME,$1
CDD      0000200 DEFAULT SYS$001=' ';IF '$1 '≠' ';DEFAULT SYS$001=',DDNAME-
CDD      0000300 =$1'
CDD      0000400 SYSCDD DSNAME
CDS      0000000 BUILTIN CDS      ,CZAFV11
CHGPASS  0000000 BUILTIN CHGPASS  ,CZATI11
CLOSE    0000000 BUILTIN CLOSE    ,CZCHB11
CONTEXT  0000000 BUILTIN CONTEXT  ,CZASM11
CORRECT  0000000 BUILTIN CORRECT  ,CZASQ11
CPS      0000000 BUILTIN CPS      ,CZAXX13
CVV      0000000 BUILTIN CVV      ,CZAXX12
DATA     0000000 PROCDEF DATA
DATA     0000100 PARAM DSNAME,RTYPE,DBASE,DINCR
DATA     0000200 SYSXPAT DATA
DATA     0000300 DSNAME,RTYPE,DBASE,DINCR
DDEF     0000000 BUILTIN DDEF     ,CZAEA11
DDNAME?  0000000 BUILTIN DDNAME?  ,CZAEK12
DEFAULT  0000000 BUILTIN DEFAULT  ,CZATR12
DELETE   0000000 BUILTIN DELETE   ,CZAEJ21
DIRECT   0000000 BUILTIN DIRECT   ,CZABA811
DISABLE  0000000 BUILTIN DISABLE  ,CZASW7
DISPLAY  0000000 BUILTIN DISPLAY  ,CZAMD11
DMPRST   0000000 BUILTIN DMPRST   ,CZUFAB
DROP     0000000 BUILTIN DROP     ,CZCM0411
DSS?     0000000 BUILTIN DSS?     ,CZAEL11
DUMP     0000000 BUILTIN DUMP,CZAMD21
EDIT     0000000 BUILTIN EDIT,CZATS2
ENABLE   0000000 BUILTIN ENABLE,CZASW6
END      0000000 BUILTIN END,CZASW22
ERASE    0000000 BUILTIN ERASE    ,CZAEJ11
EVV      0000000 BUILTIN EVV      ,CZCFB11
EXCERPT  0000000 BUILTIN EXCERPT,CZASK11
EXCISE   0000000 BUILTIN EXCISE,CZASL11
EXECUTE  0000000 BUILTIN EXECUTE  ,CZABB11
EXHIBIT  0000000 BUILTIN EXHIBIT  ,CZAYD31
EXIT     0000000 BUILTIN EXIT     ,CZAMZ88
EXPLAIN  0000000 BUILTIN EXPLAIN,CZATJX
FLOW     0000000 BUILTIN FLOW     ,CZAGD11
FORCE    0000000 PROCDEF FORCE
FORCE    0000100 PARAM USERID
FORCE    0000200 SYSXPAT FORCE
FORCE    0000300 USERID
FTN      0000000 BUILTIN FTN      ,CFADA11
GO       0000000 BUILTIN GO,CZAMC21
HOLD     0000000 BUILTIN HOLD     ,CZCM0711
IF       0000000 BUILTIN IF       ,CZBLT11
INSERT   0000000 BUILTIN INSERT,CZASJ11
JOBLIBS  0000000 BUILTIN JOBLIBS  ,CZAEK11
JOIN     0000000 BUILTIN JOIN     ,CZAFK11
JOINRJE  0000000 BUILTIN JOINRJE  ,CZABS111
K        0000000 PROCDEF K
```

```
K        0000100 PARAM ALPHABET=$1
K        0000200 IF '$1'='3';DEFAULT ALPHABET=2
K        0000300 IF '$1'='4';DEFAULT ALPHABET=1
K        0000400 DEFAULT SYSIN=K
KA       0000000 PROCDEF KA
KA       0000100 DEFAULT SYSIN=K,ALPHABET=2
KB       0000000 PROCDEF KB
KB       0000100 DEFAULT SYSIN=K,ALPHABET=1
KEYWORD  0000000 BUILTIN KEYWORD ,CZATH11
LABEL    0000000 BUILTIN LABEL   ,CZABXX
LINE?    0000000 BUILTIN LINE?   ,CZAEM11
LIST     0000000 BUILTIN LIST,CZASP11
LNK      0000000 BUILTIN LNK     ,CFADA31
LOAD     0000000 BUILTIN LOAD,CZBTG11
LOCATE   0000000 BUILTIN LOCATE,CZASN11
LOGOFF   0000000 BUILTIN LOGOFF  ,CZAFN11
LPDS     0000000 BUILTIN LPDS    ,CZAXX14
MCAST    0000000 BUILTIN MCAST   ,CZATU21
MCASTAB  0000000 BUILTIN MCASTAB ,CZATU31
MODIFY   0000000 BUILTIN MODIFY  ,CZAEG11
MSG      0000000 PROCDEF MSG
MSG      0000100 PARAM USERID=$1,TEXT
MSG      0000200 SYSXPAT MSG
MSG      0000300 USERID=$1,TEXT
MTT      0000000 BUILTIN MTT     ,CTCBPKD
NEWMSG   0000000 BUILTIN NEWMSG  ,CZATJD
NUMBER   0000000 BUILTIN NUMBER,CZASU11
PATCLEAR0000000 BUILTIN PATCLEAR,CZAFO3
PATFIX   0000000 BUILTIN PATFIX  ,CZUPFIX
PC?      0000000 BUILTIN PC?     ,CZAEL22
PERMIT   0000000 BUILTIN PERMIT  ,CZAFH11
PLI      0000000 BUILTIN PLI     ,CFBAA8
POD?     0000000 BUILTIN POD?    ,CZCOX11
POST     0000000 BUILTIN POST    ,CZASW8
PRINT    0000000 PROCDEF PRINT
PRINT    0000100 PARAM DSNAME,STARTNO,ENDNO,PRTSP,SZ01,SZ02,SZ03,SZ04,SZ05,SZ06,SZ07,SZ08,-
PRINT    0000200 HEADER,LINES,PAGE,ERASE,ERROPT,FORM,STATION,TAPOPT
PRINT    0000300 IF 'PRTSP'≠'EDIT';DEFAULT SY01='HEADER',SY02='LINES',SY03='PAGE',-
PRINT    0000400 SY04='ERASE',SY05='ERROROPT',SY06='FORM',SY07='STATION',SY08='TAPOPT';-
PRINT    0000500 PRINT2 DSNAME,STARTNO,ENDNO,PRTSP,SZ01,SZ03,SZ04,SZ05,SZ06,SZ07,SZ08
PRINT    0000600 IF 'PRTSP'='EDIT';DEFAULT SY01='ERASE',SY02=ERROROPT',SY03='FORM',-
PRINT    0000700 SY04='STATION',SY05='TAPOPT';PRINT3 DSNAME,STARTNO,ENDNO,PRTSP,SZ01,SZ02,SZ03,-
PRINT    0000800 SZ04,SZ05
PRINT2   0000000 PROCDEF PRINT2
PRINT2   0000100 PARAM DSNAME,STARTNO,ENDNO,PRTSP,SY01,SY02,SY03,SY04,SY05,SY06,SY07,SY08
PRINT2   0000200 SYSXPAT PRINT
PRINT2   0000300 DSNAME,STARTNO,ENDNO,PRTSP,SY01,SY02,SY03,SY04,SY05,SY06,SY07,SY08
PRINT3   0000000 PROCDEF PRINT3
PRINT3   0000100 PARAM DSNAME,STARTNO,ENDNO,PRTSP,SY01,SY02,SY03,SY04,SY05
PRINT3   0000200 SYSXPAT PRINT
PRINT3   0000300 DSNAME,STARTNO,ENDNO,PRTSP,SY01,SY02,SY03,SY04,SY05
PRMPT    0000000 BUILTIN PRMPT   ,CZBTC2
PROCDEF  0000000 BUILTIN PROCDEF ,CZATP12
PROFILE  0000000 BUILTIN PROFILE,CZASZ2
PUNCH    0000000 PROCDEF PUNCH
PUNCH    0000100 PARAM DSNAME,CBIN,STARTNO,ENDNO,STACK,ERASE,FORM
PUNCH    0000200 SYSXPAT PUNCH
PUNCH    0000300 DSNAME,CBIN,STARTNO,ENDNO,STACK,ERASE,FORM
PUSH     0000000 BUILTIN PUSH    ,CZAMZ111
QUALIFY  0000000 BUILTIN QUALIFY,CZAMR11
QUIT     0000000 PROCDEF QUIT
QUIT     0000100 PARAM USERID
QUIT     0000200 SYSXPAT QUIT
QUIT     0000300 USERID
QUITRJE  0000000 BUILTIN QUITRJE ,CZABS211
REGION   0000000 BUILTIN REGION,CZASF11
REJOIN   0000000 BUILTIN REJOIN  ,CZAFK12
RELEASE  0000000 BUILTIN RELEASE ,CZAFJ11
REMOVE   0000000 BUILTIN REMOVE,CZAMS11
REPLY    0000000 BUILTIN REPLY   ,CZACA31
RET      0000000 BUILTIN RET     ,CZAEN11
REVISE   0000000 BUILTIN REVISE,CZASH11
RPS      0000000 BUILTIN RPS     ,CZAXX11
RT       0000000 PROCDEF RT
RT       0000100 PARAM VOLUME,TATYPE,USERID,DSNAME1,DSNAME2,LINE,ERROROPT,-
RT       0000200 CTLG=$1
RT       0000300 IF '$1 '≠' ';SYSXPAT RT;CTLG,USERID,DSNAME1,DSNAME2,LINE,-
RT       0000400 ERROROPT
RT       0000500 IF '$1 '=' ';SYSXPAT RT;VOLUME,TATYPE,USERID,DSNAME1,-
RT       0000600 DSNAME2,LINE,ERROROPT
RTRN     0000000 BUILTIN RTRN    ,CZAMZ77
RUN      0000000 PROCDEF RUN
RUN      0000200 PARAM LOC
RUN      0000250 IF 'LOC '=' ';GO
RUN      0000300AIF 'LOC'='ASM';SYSASM
```

```
RUN      0000400AIF 'LOC'='FTN';SYSFTN
RUN      0000500AIF 'LOC'='LNK';SYSLNK
RUN      0000600AIF 'LOC'≠'ASM'&'LOC'≠'FTN'&'LOC'≠'LNK'&'LOC '≠' ';CALL LOC
SARD     0000000 BUILTIN SARD     ,CZAYEB
SECURE   0000000 PROCDEF SECURE
SECURE   0000100 PARAM $1,$2,$3
SECURE   0000200 DEFAULT SY01='',SY02=''
SECURE   0000300 IF '$2 '≠' ';DEFAULT SY01=',$2'
SECURE   0000400 IF '$3 '≠' ';DEFAULT SY02=',$3'
SECURE   0000500 SECURE1 $1
SECURE1  0000000 PROCDEF SECURE1
SECURE1  0000100 PARAM $1,SY01,SY02
SECURE1  0000200 SYSXPAT SECURE
SECURE1  0000300 $1SY01SY02
SET      0000000 BUILTIN SET,CZAMA11
SHARE    0000000 PROCDEF SHARE
SHARE    0000100 PARAM DSNAME,USERID,OWNERDS
SHARE    0000200 SYSXPAT SHARE
SHARE    0000300 DSNAME,USERID,OWNERDS
SHUTDOWN0000000 PROCDEF SHUTDOWN
SHUTDOWN0000200 SYSXPAT SHUTDOWN
STACK    0000000 BUILTIN STACK    ,CZAMZ101
STET     0000000 BUILTIN STET,CZASV11
STOP     0000000 BUILTIN STOP,CZAMC11
STRING   0000000 BUILTIN STRING   ,CZASB12
SYNCCAT  0000000 BUILTIN SYBCCAT  ,CZUFY11
SYNONYM  0000000 BUILTIN SYNONYM,CZATR11
SYSCDD   0000000 PROCDEF SYSCDD
SYSCDD   0000100 PARAM $1,'SYS$001'
SYSCDD   0000200 SYSXPAT CDD
SYSCDD   0000300 $1SYS$001
SYSPDEF  0000000 BUILTIN SYSPDEF,CZATP12
SYSXPAT  0000000 BUILTIN SYSXPAT,CZATE11
TIME     0000000 PROCDEF TIME
TIME     0000100 PARAM MINS
TIME     0000200 SYSXPAT TIME
TIME     0000300 MINS
TV       0000000 PROCDEF TV
TV       0000100 PARAM DSNAME1,DSNAME2
TV       0000200 SYSXPAT TV
TV       0000300 DSNAME1,DSNAME2
UNLOAD   0000000 BUILTIN UNLOAD,CZBTG21
UPDATE   0000000 BUILTIN UPDATE,CZASR11
UPDTUSER0000000 BUILTIN UPDTUSER,CZAGC11
USAGE    0000000 BUILTIN USAGE    ,CZAGB11
VMEREP   0000000 PROCDEF VMEREP
VMEREP   0000100 PARAM EMPTY
VMEREP   0000200 SYSXPAT VMEREP
VMEREP   0000300 EMPTY
VSS      0000000 PROCDEF VSS
VSS      0000100 PARAM USER
VSS      0000200 SYSXPAT VSS
VSS      0000300AUSER
VT       0000000 PROCDEF VT
VT       0000100 PARAM DSNAME1,DSNAME2
VT       0000200 SYSXPAT VT
VT       0000300 DSNAME1,DSNAME2
VV       0000000 PROCDEF VV
VV       0000100 PARAM DSNAME1,DSNAME2
VV       0000200 SYSXPAT VV
VV       0000300 DSNAME1,DSNAME2
WT       0000000 PROCDEF WT
WT       0000200 PARAM DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,-
WT       0000300 PRTSP,$1,$2,$3,ERASE,HEADER,LINES,PAGE
WT       0000350 DEFAULT SY01='',SY02='',SY03='',SY04=''
WT       0000400AIF 'PRTSP'='EDIT';DEFAULT SY01=',ERASE';IF '$1'≠'';DEFAULT SY01=',$1'
WT       0000500AIF 'PRTSP'='EDIT';WT2 DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,PRTSP
WT       0000600 IF 'PRTSP'≠'EDIT';DEFAULT SY01=',HEADER',-
WT       0000700 SY02=',LINES',SY03=',PAGE',SY04=',ERASE';-
WT       0000800 WT1 DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,PRTSP,$1,$2,$3
WT1      0000000 PROCDEF WT1
WT1      0000100 PARAM DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,-
WT1      0000200 PRTSP,$1,$2,$3
WT1      0000300 IF '$1'≠'';DEFAULT SY01=',$1'
WT1      0000400 IF '$2'≠'';DEFAULT SY02=',$2'
WT1      0000500 IF '$3'≠'';DEFAULT SY03=',$3'
WT1      0000600 WT2 DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,PRTSP
WT2      0000000 PROCDEF WT2
WT2      0000200 PARAM DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,-
WT2      0000300 PRTSP,SY01,SY02,SY03,SY04
WT2      0000400 SYSXPAT WT
WT2      0000500 DSNAME,DSNAME2,VOLUME,FACTOR,STARTNO,ENDNO,PRTSP-
WT2      0000600 SY01SY02SY03SY04
ZLOGON   0000000 PROCDEF ZLOGON
ZZZZZZ   0000000 PROCDEF ZZZZZZ
```

Where more than one page reference is given, the major reference is first.

GY28-2013-6

Command System Printed in U.S.A. GY28-2013-6

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

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

IBM Technical Newsletter

File Number S360-36

Base Publication No. GY28-2013-6

This Newsletter No. GN28-3214

Date February 1, 1972

Previous Newsletters None

IBM System/360 Time Sharing System
Command System

This Technical Newsletter provides replacement pages for the subject publication. Pages to be inserted and/or removed are:

135-136

A change to the text is indicated by a vertical line to the left of the change.

Summary of Amendments

A new return code has been added to the PAIR (CZACS) routine. This code indicates that an invalid parameter was passed to PAIR.

IBM Corporation, Dept. 643, Neighborhood Road, Kingston, N. Y. 12401

PRINTED IN U.S.A.