File No. S370-36
Order No. GC28-2004-6

Systems

IBM Time Sharing System Assembler User Macro Instructions

Provides the information necessary to code assembler language macro instructions in the IBM Time Sharing System (TSS). The intended audience is nonprivileged assembler language users.

The primary macro instruction services are program management and data management. These macro instructions facilitate TSS application programming.

The first section describes the macro instructions by functional groups, enabling the user to select macro instructions needed to manage programs (manage virtual storage, load and link modules, handle interruptions, transfer to command mode, use SYSIN/SYSOUT and the system log, communicate with the operator, maintain timers, and create commands) and to manage data (define, connect, access, manipulate, disconnect, or remove data sets). The second section lists the macro instructions alphabetically and provides the information needed to code the macro instructions. Appendixes describe exit lists, synchronous error exits, end-of-data processing, machine control characters, linkage conventions, DCB fields, the DDEF macro instruction, the generation of literals by macro instructions, interruption handling, the TSS Macro and Copy library, data set sharing, the OPEN/CLOSE generated parameter list, and the conditional assembly of macro instructions.

PREREQUISITE PUBLICATIONS

The reader should be familiar with the information presented in the publications:

IBM Time Sharing System:
    Concepts and Facilities, GC28-2003
    Assembler Language, GC28-2000

IBM

PREFACE

This publication contains a description of the IBM Time Sharing System (TSS) macro instructions available to the nonprivileged assembler language user.

The publication is divided as follows:

Part I: Macro Instruction Services - contains a summary of the macro instructions arranged into functional categories. Following a brief summary of the categories, Section 1 describes the data management services and Section 2 describes the program management services of TSS.

Part II: Macro Instructions - provides descriptions of the TSS macro instructions. Section 1 shows how the macro instructions are described and defines the terms and symbols used in the macro instruction descriptions. Section 2 includes the detailed descriptions of the macro instructions, arranged alphabetically; the TAMII macro instructions are described in Appendix N.

Appendixes -- explains the use of exit routines, control characters available wi certain data management facilities and i terruption handling routines, and the conditional assembly of macro instructions.

PREREQUISITE TSS PUBLICATIONS:

Concepts and Facilities, GC28-2003

Assembler Language, GC28-2000

Other recommended TSS publications are:

Command System User's Guide, GC28-2001

Assembler Programmer's Guide, GC28-2032

Seventh Edition (August 1976)

This is a revision of, and makes obsolete, GC28-2004-5 and Technical Newsletter GN28-3202.

This edition has been updated technically to reflect the addition of new macros for handling virtual storage segments. These macros are as follows:

RSVSEG -- Reserve Segment Group
DISCSEG -- Disconnect Segment Group
RELSEG -- Release Reserved Segment Group
DELSEG -- Delete Disconnected Segment Group
CONSEG -- Connect Disconnected Segment Group

Other technical changes are:

A new operand has been added to the GETMAIN macro.

New interruption codes for the SPEC macro have been added to indicate monitor call and program event recording.

A change in return codes for the GATRD macro.

A new parameter for the SAEC and SIEC macros.

This edition is current with Release 2.0 of the IBM Time Sharing System/370 (TSS/370), and remains in effect for all subsequent versions or modifications unless otherwise noted. Changes or additions to this publication will be provided in Technical Newsletters or, if changes are significant, in a new edition.

FIGURES

<u>Tables</u>

SECTION 1: OVERVIEW OF SERVICES

The TSS macro instructions provide two basic services: data set management and program management. These services are summarized below.

TSS provides data management macro instructions that:

- Define a data set -- by introducing it to a task, describing its characteristics or attributes, such as its name, organization, disposition (that is, OLD or NEW), and cataloging it by name. After the data set has been connected to the system, TSS refers to the attributes to determine the access method and control information.

- Connect a data set -- by making its attributes available to the system in the data control block (DCB). Appropriate access method routines are initialized, labels are processed, and the data set is positioned for user processing.

- Access a data set -- by using the macro instructions associated with the appropriate VAM or SAM access method, or by providing user-written, input/output device-management routines with the IOREQ macro instruction.

- Manipulate a data set -- by transferring it, rather than individual records within a data set, from one area of virtual storage to another, or to punched cards, printed listings, or magnetic tape.

- Disconnect a data set -- by telling the system that a user has finished processing the data set and, permanently or temporarily, disconnecting from the system the DCB containing the description of the data set attributes and access-method specifications.

- Remove a data set -- by physically erasing it and releasing the storage in which it was recorded.

TSS provides program management macro instructions that:

- Manage virtual storage -- by acquiring or releasing pages or multiples of eight bytes, or by transforming contiguous virtual storage bytes into an object module that is a single control section.

- Load and link a module -- by explicitly or implicitly loading object modules and establishing standard linkage between the calling and the called modules.

- Handle interruptions -- by assuming control of specific types of interruptions and executing user-written interruption-servicing routines, instead of system-supplied routines.

- Transfer to command mode -- by interrupting a program's execution, temporarily or permanently, and passing control to command mode for further processing.

- Communicate with SYSIN/SYSOUT -- by passing data, messages, and commands between a coded program and SYSIN/SYSOUT devices.

- Communicate with operator and log -- by passing messages, issued during a program's execution, to the system operator and by recording them in his log.

- Maintain timers -- by setting them to measure the time of a task's execution or the elapsed calendar time.
- Create commands -- by specifying user-written commands to be issued instead of, or in addition to, system-supplied commands.
- Use system-oriented macro instructions -- by employing those that are intended primarily for system programmers, but which are available to all users.

The macro instructions by which these services are requested are subsequently discussed. Section 2 provides a general description of each of the functional groups and a brief description of the function provided by each macro instruction. Part II of this publication contains the detailed description of each macro instruction.

SECTION 2: DATA SET MANAGEMENT

This section describes the TSS macro instructions that are available to the user for managing data sets; they are presented in functional groups that reflect their primary use in the system. Detailed explanations of these macro instructions are presented in Part II, Section 2, "Macro Instruction Descriptions."

DATA SET SPECIFICATION MACROS

Certain characteristics of a data set must be described to the TSS routines for data set task management before a user can use these facilities to process and manipulate his data sets. These data set attributes can be furnished to the system by from two-to-six sources, depending on whether a data set is new or one that has been previously defined to the system. The sources and their priorities are described in Appendix F. The two major sources, and the only mandatory sources, used to describe these data sets to the system are the DDEF and DCB macro instructions.

Attributes in a data set description are automatically cataloged when a public or private virtual access method (VAM) data set is opened. For sequential access method (SAM) data sets, however, the user must request that such attributes be recorded in the catalog by issuing a CAT macro instruction, which can also be used to rename VAM or SAM data sets and to alter catalog entries for SAM data sets. These entries can be deleted from a user catalog by the DEL macro instruction.

DDEF	invokes the DDEF command processor to provide the connection between a program and a data set.
CDD	calls the DDEF command processor with one or more DDEF commands obtained from a line data set.
FINDDS	locates the JFCB corresponding to a given data set name, and optionally creates a JFCB (invokes DDEF) if the data set name is in the catalog.
FINDJFCB	locates the JFCB corresponding to a given DDNAME, and optionally creates a JFCB (invokes DDEF).
CAT	invokes the CATALOG command processor to catalog data sets, rename data sets, or create generation data groups.

REL	invokes the RELEASE command processor to dispose of the specified JFCB, freeing the symbolic name of the corresponding DDEF statement for other use. Devices used for data sets on private volumes are optionally released for general use. RELEASE is used to free data sets from concatenation and to close and remove data sets from the job library chain. A RELEASE of the symbolic name of the DDEF statement associated with an open data set results in that data set being closed. Any programs loaded from a job library are unloaded by releasing a job library.
DEL	invokes the DELETE command processor to remove data set names from the catalog.
DCB	defines storage for a data control block.
DCBD	generates a dummy control section (DSECT) to describe the DCB with names having the appropriate attributes for DCB fields.

DATA CONTROL BLOCK PROCESSING MACROS

Before processing a data set, a user must connect it to the system. The OPEN macro instruction causes the system to interrogate for the data set attribute information specified by the DDEF and DCB macro instructions or any other available sources. The system determines whether an appropriate data set organization has been specified and whether all the necessary attributes for processing such a data set have been provided. If the user has indicated that he wants to alter the DCB contents during OPEN processing, by including the EXLST parameter (for BSAM and QSAM only) with his attribute specifications, the system immediately exits to the user modification routine. When all the required attributes have been provided, all new VAM data sets, public or private, are automatically created and cataloged (any new or uncataloged SAM data sets must be cataloged via the CAT macro instruction). For previously cataloged VAM data sets, the system uses attribute specifications recorded in the catalog. Any storage requirements indicated by a DDEF space parameter are then allocated accordingly. The system then makes available the access method that the user indicated he wants to employ (via attribute specification), thereby logically connecting him to the system.

At the time a user opens a data set, he can select or default a processing option that indicates to the system the type of processing he expects to perform on that data set. The processing option specified when the user issues the OPEN macro instruction determines whether all the macro facilities of an access method or only a portion of them can be used. If a user opens a data set for input only, he will only be allowed to use macro instructions that retrieve data; he will not be allowed to use those that store data into the data set he has opened.

Once the system knows the processing option and locates the device on which a data set is to reside, or currently resides, it proceeds to physically open that data set by processing labels and physically positioning the user at the data record he wants to process. The initial positioning directed by the system varies depending on the access method, the processing option, device type, and in some cases the status (that is, MOD) of the data set.

OPEN	collects the attributes of specified data sets from various sources, by priority, and merges the information in the respective DCBs. OPEN prepares a DCB and the data set associated with it for processing.

CLOSE	reverses the action of OPEN. CLOSE waits until I/O requests are complete before proceeding. When appropriate, output data set trailer labels are processed and access to volumes is positioned as specified. Control blocks, such as the DCB and JFCB, are restored to their original condition. CLOSE disconnects a data set from further processing and user access. For BSAM and VAM DCBs there is a CLOSE option that causes the same processing as the standard CLOSE macro except that fields of the DCB are not restored to their status before OPEN; the DCBs are in effect open and additional processing may be performed. With BSAM data sets, temporary close is useful for repositioning a volume for subsequent processing and serves the purpose of completing the data set (if it has just been written or extended). In the case of VAM data sets, temporary close causes the DSCBs to be written which captures the current status of the data set on external storage.

ACCESS METHOD MACROS

When a data set has been given a name, when its attributes have been described, and when it has been connected to the system, the user can employ the routines provided by the TSS data set management facilities for storing and retrieving data. These routines are called by using I/O macro instructions in the user's source program. The macro instructions are part of an access method and are dependent on the manner in which a user organizes and processes his data. There are two primary types of access method: virtual access method (VAM) and sequential access method (SAM).

VAM:

These are the access methods used in TSS. Data sets that must be interchanged with programs running under the OS or OS/VS Programming System, or data sets to be written on magnetic tape should be accessed using SAM.

Users create, read, and process VAM data sets on the basis of logical records. The system, however, blocks these records by pages (4096 bytes); the page is the unit of transfer between the direct access device and the user's virtual storage. The system also ensures that only those pages of a data set actually required are resident in virtual storage. Because VAM data sets can be organized as either sequential, index sequential, or partitioned, three distinct access methods are provided under VAM for data set processing:

Data Set Organization	Access Method
sequential	virtual sequential (VSAM)
index sequential	virtual index sequential (VISAM)
partitioned	virtual partitioned (VPAM)

SAM:

These are the access methods used for records that can be read and written with programs running under control of the OS or OS/VS Programming System, or when the data set is to be written on magnetic tape.

Users create, read, and process SAM data sets on the basis of physical records. The records within a physical record can be blocked or unblocked. Because of this, two access methods are provided under SAM for processing data sets:

<u>Data Set Organization</u>	<u>Access Method</u>
unblocked sequential	basic sequential (BSAM)
blocked sequential	queued sequential (QSAM)

Another special access facility, the input/output request facility (IOREQ) is provided for users who would rather program their own I/O device-control routines than employ any of the system-provided access methods.

Explanations of each of these access methods, and the macro instructions that may be used with them, follow.

<u>Virtual Sequential Access Method</u>

The virtual sequential access method (VSAM) enables a user to process virtual sequential data sets. These data sets can be stored on, or retrieved from, direct access devices only. The record format within each such data set may be fixed-length (blocked or unblocked), variable-length (blocked or unblocked), or undefined-length (unblocked only). Such attributes are unique for each data set; they must be defined to the system before a data set can be accessed by VSAM.

GET	reads logical records in sequential order.
PUT	writes logical records in sequential order.
PUTX	replaces a logical record, previously read by GET.
SETL	logically positions access to a data set at the beginning or end, at the previous record, or at any logical record within a sequential data set. Subsequent PUT or GET operations will proceed from this new position.

<u>Virtual Index Sequential Access Method</u>

The virtual index sequential access method (VISAM) enables a user to process index sequential data sets. These data sets may be stored on, or retrieved from, direct access devices only. The record format within each such data set may be fixed-length (blocked or unblocked) or variable-length (blocked or unblocked) format. Such attributes are unique for each data set; they must be defined to the system before a data set can be accessed by VISAM.

GET	reads logical records in sequential order, by key.
PUT	writes logical records in sequential order, by key.
READ	reads logical records in nonsequential or sequential order.
WRITE	writes logical records in nonsequential or sequential order.
DELREC	deletes a specified logical record from a data set.
SETL	logically positions access to a data set at its beginning at the previous record, or at any logical record. Subsequent PUT or GET operations will proceed from this new position.
ESETL	releases a read-lock set by other operations.
RELEX	releases a write-lock set by other operations.

Virtual Partitioned Access Method

The virtual partitioned access method (VPAM) enables a user to access partitioned data sets. Each partitioned segment (or member) is a complete VSAM or VISAM data set. The organizations of the records within members are the same as within VSAM or VISAM. VPAM may be used only to store or retrieve data set members on direct access devices.

When a partitioned data set has been defined and connected to the system, the user may employ the VPAM macro instructions (FIND and STOW) to locate its members. When the member is opened and located via a FIND macro instruction, the VSAM or VISAM macro instructions can be used to process the member. Although a member is defined by the same DDEF and DCB macro instructions that defined the partitioned data set, the member is not opened until a VPAM FIND macro instruction is executed.

FIND	opens an individual member within a VPAM data set for processing. After FIND, appropriate VISAM or VSAM macros can be used to process the records within the member.
STOW	causes a VISAM or VSAM member of a partitioned data set to be added to or deleted from the data set. It also adds, changes, deletes, or replaces member names or aliases, and provides for storing additional information in the partitioned organization directory (POD), as user data.

Basic Sequential Access Method

The basic sequential access method (BSAM) enables a user to access unblocked physical sequential data sets. Since BSAM does not provide a user with blocking/deblocking or buffering routines, it should be used primarily to process unblocked records. QSAM facilitates the processing of blocked records. A physical sequential data set can be stored on, or retrieved from, disk or tape. The record format within each such data set can be fixed-length (blocked or unblocked), variable-length (blocked or unblocked), or undefined-length (unblocked only). Such attributes are unique for each data set; they must be defined to the system before a data set can be accessed by BSAM.

READ	reads a physical record from an I/O device and specifies or defines a data event control block (DECB) to be used to indicate completion status for the operation. After READ, control is returned to the user program. The user program is responsible for deblocking logical records from physical records.
WRITE	is the same as READ except that data transfer is in the opposite direction.
CHECK	tests the queue of DECBs associated with READ or WRITE operations to determine if the operations are complete and if so, whether errors or exceptional conditions occurred.
DQDECB	removes all unchecked DECBs associated with READ and WRITE operations for a specified device. DQDECB is used when restarting I/O after user program action on error conditions.
NOTE	makes available to the program, for use with POINT, the relative position within a volume of the last block read or written.
POINT	repositions access to a data set at a specified block within the data set.

BSP	backspaces one physical record or block on the current tape or direct-access volume regardless of the direction in which data is being stored or retrieved on that device.
CNTRL	controls tape positioning and writing of tape marks. CNTRL can be used to obtain sense data from tape or direct-access devices.
FEOV	positions access to the data set at the next volume of a multivolume set.
GETPOOL	requests allocation of virtual storage for use as a buffer pool and assigns that area to a DCB.
GETBUF	obtains a buffer work area from a buffer pool previously assigned to a DCB either by a GETPOOL macro or as provided according to DCB buffer options.
FREEBUF	returns a buffer work area obtained by GETBUF to the related buffer pool.
FREEPOOL	releases areas previously assigned to specified DCBs as buffer pools either by a GETPOOL macro or as a result of buffer options specified in the DCB.

Queued Sequential Access Method

The queued sequential access method (QSAM) enables a user to access blocked or unblocked physical sequential data sets. QSAM, in contrast to BSAM, permits the programmer to store and retrieve records of a sequential data set without coding his own blocking/deblocking and buffering routines. A sequential data set can be stored on, or retrieved from, disk or tape. The record format within each such data set can be fixed-length (blocked or unblocked), variable-length (blocked or unblocked), or undefined-length (unblocked only). Such attributes are unique for each data set; they must be defined to the system before a data set can be accessed by QSAM.

GET	reads logical records in sequential order. The initial GET causes a physical record from the input device to be transferred to a system-maintained buffer area and makes the first logical record available to the user program. Each subsequent GET delivers logical records until all logical records within the physical record have been processed. Meanwhile, the next physical block is transferred.
RELSE	causes the remaining records of the current input buffer to be ignored and positions access to the data set at the first logical record of the next physical record. The next GET macro will retrieve the first logical record from the new input buffer.
PUT	is the same as GET except that data transfer is in the opposite direction.
PUTX	replaces a logical record, previously read by GET, or writes an updated or identical logical record directly from an input data set to an output data set.
TRUNC	causes the current output buffer to be regarded as if it were filled. The output buffer is written to the output device, leaving access to the data set positioned at the next buffer area. The next PUT issued is for the first record of the next block.

SETL	logically positions access to a data set at its beginning or end, at the previous record, or at any logical record. Subsequent PUT or GET operations will proceed from this new position.
CNTRL	controls tape positioning and writing of tape marks. CNTRL can be used to obtain sense data from tape or direct-access devices.

<u>Input/Output Request Access Method</u>

The input/output request facility (IOREQ) enables users to program their own I/O device-control routines, rather than use those from VAM or SAM. IOREQ provides a means to control I/O devices through user specification of channel command words (CCWs) that are normally created by the TSS-supplied access methods. Using IOREQ, the user can create a series of these channel instructions and execute them as he desires. The IOREQ, CHECK, and VCCW macro instructions enable users to create their own specialized access methods.

As with provided access methods, before the IOREQ facilities can be used to access a data set, the data set must be described and connected to the system and, when the user has finished using the data set, it must be disconnected from the system.

IOREQ	initiates a request for an I/O operation specified by a user-written channel program and specifies or defines a data event control block (DECB) to be used to indicate completion status for the operation. After IOREQ, control is returned to the user program.
CHECK	tests the queue of DECBs associated with IOREQ operations to determine if the operations are complete and if so, whether errors or exceptional conditions occurred.
DQDECB	removes all unchecked DECBs associated with IOREQ operations to a specified device. DQDECB is used when restarting I/O after user program action on error conditions.
VCCW	defines storage for a virtual channel command word (VCCW). A VCCW serves the same function as a CCW. The format is rearranged to allow for the 32-bit addressing mode of the 360/67. Chains of one or more VCCWs specify I/O operations to be performed.

COPY DATA SET & BULK OUTPUT MACROS

Entire data sets can be transferred from one storage device to another. A data set can be moved from one direct access device to another, or to a different area on the same direct access device. They can also be copied to punched cards, printer listings, or magnetic tape devices. Several macro instructions are provided with TSS to perform these operations.

A user may decide to include an existing data set in a partitioned data set, to renumber the lines of an existing line data set, or to store an existing data set on a different device, releasing the device on which the data set is stored. The COPYDS macro instruction lets a user accomplish these operations.

The bulk-output facilities allow a user to transfer entire data sets from virtual storage to punched cards, printer listings, or magnetic tape devices. These facilities provide a user with three macro instructions, print (PR), punch (PU), and write tape (WT), to accomplish these transfers. These three macro instructions are to be issued in a user program on SAM, VSAM and VISAM data sets only. Although VPAM data sets or members cannot employ these macro instructions, the members of the VPAM data set can first be copied with a COPYDS macro instruction (or command) into new VSAM or VISAM data sets and then be operated on by these macro instructions. Execution of these macro instructions causes requests for particular output operations to be set up as independent nonconversational tasks, places the requested task on a bulk output queue, and returns to the user's problem program. The user can then continue processing other data sets (or terminate his session) while the output task is being executed.

COPYDS	invokes the CDS command processor to create copies of existing data sets or members of partitioned data sets that have been previously defined to the system and reside on direct access or magnetic tape volumes. It also creates copies of line data sets with renumbered lines. The copies are placed in new data sets. The new data sets and the existing old data sets must be previously defined to the system. The old data sets do not, however, need to be opened by the user; they are opened by the CDS command processor.
PR	invokes the PRINT command processor to list a specified data set on a high-speed, on-line punch and, optionally, erases it from the user's catalog when the printing has been finished. Line spacing on the printed output can also be indicated by the user. The print operation takes place as an independent nonconversational task.
PU	invokes the PUNCH command processor to cause a data set to be punched on-line and, optionally, erases it from the user's catalog when the punching is finished. Stacker selection can also be indicated by the user. The punch operation takes place as an independent nonconversational task.
WT	invokes the WT command processor to cause a data set to be written on magnetic tape in proper format for subsequent off-line printing and, optionally, erases it from the user's catalog when the writing is finished. The write-tape operation takes place as an independent conversational task.

ERASE DATA SET MACRO

ERASE	invokes the ERASE command processor to uncatalog and free the space occupied by direct-access data sets.

SECTION 3: PROGRAM MANAGEMENT

This section describes TSS program management macro instructions. They are presented in functional groups that reflect their primary use in the system.

VIRTUAL STORAGE MANAGEMENT MACROS

GETMAIN	is used to acquire additional virtual storage.
FREEMAIN	releases virtual storage acquired with GETMAIN.
CKCLS	determines the most restrictive protection class assigned to a specified number of contiguous halfpages of virtual storage.
CSTORE	saves contiguous virtual storage areas in object module format.
RSVSEG	associates a name with a contiguous set of virtual storage segments.
DISCSEG	disconnects a segment group from a virtual address space and assigns a name to it.
CONSEG	connects a disconnected segment group to an unassigned portion of a virtual address space.
RELSEG	releases a reserved segment group, deleting the name, but leaving addressable the virtual address space of the group.
DELSEG	deletes a disconnected segment group. The name and any space on auxiliary storage are deleted.
EXCSEG	performs the CONSEG and DISCSEG macro instructions in one operation.
GETSEG	gets a page from a disconnected segment group and places it in a buffer specified by the user.
PUTSEG	puts a page from a virtual storage buffer into an existing disconnected segment group.

PROGRAM LINKAGE MACROS

A user has two ways of requesting that a module be loaded into virtual storage: an implied request or an explicit request. An implied request causes automatic loading of a program into a user's virtual storage, during program assembly, each time the source program refers to (via the CALL macro instruction) an undefined external symbol. An explicit request is satisfied during the actual execution of the program containing the request. When the explicit request (via a CALL or LOAD macro instruction) is executed, the module referred to is loaded into virtual storage assigned to the user's task.

Unlike the implicit call, the program loaded by an explicit call during program execution may be released by a DELETE macro instruction or an UNLOAD command. This releases the virtual storage area occupied by that program for other use.

When a user's program calls another program, either explicitly or implicitly, these programs establish linkage by using standard TSS linkage conventions. Thus, proper registers must be used in establishing linkage, and a save area must be set aside in the calling program. Two macro instructions (SAVE and RETURN) establish standard linkage.

LOAD	explicitly loads a program, if it is not already loaded, into virtual storage. The address at which the program has been loaded can be obtained from address constants previ-

	ously defined by an ADCON or ARM macro. The program remains in virtual storage until it is unloaded by a DELETE macro or an UNLOAD command.
CALL	explicitly or implicitly loads the called program into virtual storage and establishes conventional linkage between the calling and called program. The address at which the program has been loaded can be obtained from address constants previously defined by an ADCON or ARM macro. CALL causes control to be given to the called program.
ARM	initializes the address constant group defined by an ADCON macro with the name of the program, entry point, or control section that is to be loaded into virtual storage. The initialized address constant group can subsequently be used by a CALL or LOAD macro to explicitly load the program.
ADCON	generates a group of address constants for use by CALL, LOAD, or DELETE macro instructions.
ADCOND	generates a DSECT to describe the address constant group with names having the appropriate attributes. These names make it possible for an assembler language program to reference symbolically the resolved address constants and control flags placed in the group during execution of a LOAD or explicit CALL macro.
DELETE	unloads an explicitly loaded program that is no longer needed, freeing virtual storage. Any associated programs are also deleted.
SAVE	stores the contents of the general registers according to a standard convention. The SAVE macro is normally the first instruction in a called routine.
RETURN	restores the contents of the general registers according to a standard convention and returns control to the calling routine, optionally setting a return code for the calling routine.
MARKRTRN	indicates to the calling program that the called program has returned.

INTERRUPT HANDLING MACROS

TSS provides interruption-handling facilities that permit the user to control task interruptions. User-written routines can be invoked to service interruptions; these routines, which decide how to respond to each type of interruption, can ignore certain interruptions.

SIR	specifies a user interrupt routine (named via a SPEC, SAEC, SIEC, SEEC, STEC, or SSEC macro, according to the type of interrupt) to the task monitor. SIR specifies the processing priority for that routine. The user's routine replaces any system-supplied interruption servicing routines for this type of interruption, unless the user's routine is deactivated with the DIR macro. System-supplied routines are reinstated after the user routines are deleted.
DIR	deletes an interruption servicing routine, reversing the effect of the corresponding SIR macro.
SPEC	names a user-written program interruption servicing routine and defines an interrupt control block (ICB) in which data

pertaining to a program interruption can be recorded. The named routine will be used when it is defined to the task monitor as an interruption servicing routine by a SIR macro.

SSEC	names a user-written SVC interruption servicing routine and defines an ICB in which data pertaining to an SVC interruption can be recorded. The named routine will be used when it is defined to the task monitor as an interruption servicing routine by a SIR macro.
SEEC	names a user-written external interruption servicing routine and defines an ICB in which data pertaining to an external interruption can be recorded. The named routine will be used when it is defined to the task monitor as an interruption servicing routine by a SIR macro.
SAEC	names a user-written asynchronous interruption servicing routine and defines an ICB in which data pertaining to an asynchronous interruption can be recorded. The named routine will be used when it is defined to the task monitor as an interruption servicing routine by a SIR macro.
STEC	names a user-written timer interruption servicing routine and defines an ICB in which data pertaining to a timer interruption can be recorded. The named routine will be used when it is defined to the task monitor as an interruption servicing routine by a SIR macro.
SIEC	names a user-written I/O interruption servicing routine and defines an ICB in which data pertaining to an I/O interruption can be recorded. The named routine will be used when it is defined to the task monitor as an interruption servicing routine by a SIR macro.
INTINQ	inquires about the interruption information recorded in a specified ICB. Various options are available. Control can be relinquished until the ICB indicates an interrupt. If the interrupt has been queued, the routine in which the INTINQ is issued may regain control immediately. Also, the task can be made to wait until a corresponding interrupt has occurred. Interruptions queued on the ICB can be cleared by INTINQ. A specified branch can be taken if the interrupt information is present.
SAI	saves the task's current interruption servicing status indicator and inhibits further interrupts until a RAE macro is issued. Interruptions occurring while the inhibit indicator is on are saved and queued for later servicing.
RAE	restores the interruption servicing status previously saved by an SAI macro. Depending on the saved status (enabled or inhibited), processing continues. If interrupts were previously enabled, any interruptions that occurred while interruption servicing was inhibited are processed before processing continues.
PIREC	efficiently tests an address for validity. Program interrupt codes 4, 5, and 6 occurring when PIREC is being executed are not processed in the normal manner. Detection of an invalid address results in a branch to a specified location.
USATT	causes subsequent attention interruptions to be processed by a user-written routine that was previously established

as an interruption servicing routine by the SIR and SAEC macros.

CLATT	reverses the effect of a USATT macro. Control of attention interruptions obtained with a USATT macro is relinquished.
AETD	causes attention interruptions to be processed by any one of several user-written routines, depending on the number of times the attention key is pressed. The AETD macro is also used to relinquish control of attention interruptions acquired by the AETD macro.

COMMAND SYSTEM INTERFACE MACROS

TSS provides a user with several ways of interrupting a program's execution, either temporarily or permanently, and passing control to command mode for subsequent processing.

BPKDS	generates all necessary linkage information and parameter storage areas required for use during the execution of a command that was defined with the BUILTIN command. Also, information from the BPKDS expansion is used by the KEYWORD command.
GDV	gets the value for a default from the task's combined dictionary.
GETDV	gets the value for a specified name and type from the task's combined dictionary.
SETDV	sets the value for a specified name and type into the task's combined dictionary.
OBEY	temporarily passes control to the command system for execution of a specified command. The command specified by OBEY will be issued just as if the user had interrupted the program and issued the command. When the command or a program invoked as a result of the command returns control to the command system, execution of the program from which the OBEY was issued will be resumed.
PAUSE	(for conversational tasks only) writes a user-specified message on SYSOUT and causes the task to enter command mode. A GO command causes execution of the program to resume. The interruption of a program by PAUSE is very similar to that which results from an attention interrupt. If the user has control of attention interruptions before issuing a PAUSE, the system regains control of them until a GO command is issued. PAUSE is ignored in a nonconversational task.
COMMAND	is the same as PAUSE except that it is not ignored in nonconversational mode. The SYSIN data set is read for the next command. Execution of the interrupted program can be resumed with a GO command.
CLIC	is the same as the PAUSE macro except that no message is issued.
CLIP	is the same as the COMMAND macro except that no message is issued.
EXIT	is a simple way of terminating execution of a program and optionally causing a predefined system message and a user-

	specified message to be written on SYSOUT. Control is returned to the command system and the next commands are taken from SYSIN.
ABEND	indicates an abnormal end condition to the user and the operator. The ABEND macro provides for various types of system action based on the severity code specified. Codes are: (1) Terminate execution of the program, returning control to SYSIN for conversational tasks; for nonconversational tasks either delete the task from the system or switch SYSIN to a data set defined with a DDNAME of TSKABEND. (2) Terminate the task, creating a new task if the old task was conversational. (3) Terminate the task, do not create a new task. (4) Terminate the task without attempts to write to SYSOUT. (Used by privileged programs only.) A message may be specified with the ABEND macro, either as the actual message or as the identification code of a message in the system or user message file.

SYSIN/SYSOUT COMMUNICATION MACROS

The TSS communication facilities permit a user to pass data, messages, and commands, to and from a user's SYSIN and SYSOUT devices.

GATRD	reads a record from SYSIN and places it in a user-designated virtual storage area.
TGATRD	extended function form of GATRD macro.
SOLICIT	presents a continuously incremented number as a prompt to TGATRD operations.
GATWR	writes a record on SYSOUT.
TGATWR	extended function form of GATWR macro.
TGATWS	writes a record on the primary SYSOUT.
TREAD	reads (transparent) a device dependent record.
TWRITE	writes (transparent) a device dependent record.
TWRTLST	writes records from a list of virtual storage areas to SYSOUT.
GTWRC	writes a record on SYSOUT. The first byte of the record is used for carriage control when printing nonconversational SYSOUTs. Carriage control action is approximated for conversational tasks.
GTWAR	writes a record on SYSOUT and reads the next available record from SYSIN and places it in a user-designated virtual storage area.
TGTWAR	extended function form of GTWAR macro. If input buffering is in effect, the write operation is suppressed.
GTWSR	writes a record on SYSOUT and reads the response to that record, placing it in a user-designated virtual storage area. If issued in a nonconversational task, unless the user has indicated otherwise, the task will be terminated.
TGTWSR	extended function form of GTWSR macro. If input buffering is in effect, the write operation is performed immediately

	and the read operation in response to that write is performed immediately.
SYSIN	optionally writes a record on SYSOUT and reads a record from SYSIN into virtual storage. If the record is recognized as a command, it is placed in the source list for subsequent processing by the command analyzer. User programs can detect the incidence of commands and take action accordingly. Otherwise, the user program is interrupted and the command is processed.
TCNTRL	specifies miscellaneous control operations.
CHCKT	checks the status of a DECB related to SYSIN/SYSOUT operations.
TRCBUF	reads a record from the conversational buffer for the terminal and places it in a user-designated virtual storage area.
TDCMD	issues device control commands from user programs to control the terminal environment.
TCLEAR	purges any pending or active request buffers on SYSIN/SYSOUT.
TFREE	disconnects a secondary SYSIN/SYSOUT from the task.
MCAST	temporarily substitutes a user-specified character translation table and function control table. The character translation table specifies substitution of character codes for transfer of data between user programs and SYSIN and SYSOUT. The function control table identifies characters which are to have special effects, for example backspace to mean overstrike, not character correction.
ATTNSAV	saves current conditions (buffers and terminal environment) in a pushdown stack.
ATTNRST	restores previously saved conditions and buffers, disposing of the current conditions.
ATTNDST	disposes of saved conditions and buffers no longer needed.
PRMPT	invokes a system facility which prompts the user with messages from the system message file, if not from the user message file. The prompter analyzes responses to messages whose coding indicates that a response is required.

OPERATOR & SYSTEM LOG COMMUNICATION MACROS

The TSS communication facilities provide macro instructions for user-communication with the main operator's terminal and with the system log (a generation data group, in which each VSAM data set contains a record of system-to-operator and operator-to-system communications, from startup-to-shutdown). These routines should normally be used only for programs having specialized I/O routines that require operator intervention.

WTO	writes a user-specified message on the operator's console.
WTOA	writes a user-specified action message on the operator's console. Action messages differ from those sent by WTO in that they are prefixed by characters intended to catch the

	operator's eye. They should only be used when action is required by the operator, otherwise the operator may disregard the action message format.
WTOR	writes a user-specified message on the operator's console. The user task waits for the operator to respond to the message. The operator is periodically reminded of unanswered messages. The reply from the operator is made available to the program. If the operator fails to reply within a reasonable time, the user can use the attention key to regain control and decide on some other course of action.
WTL	causes a user-specified message to be written in the system log data set. If the operator wishes to have WTL messages appear on the console, a default can be set in the combined dictionary of the operator USERID.

TIMER MAINTENANCE MACROS

TSS requires maintenance of elapsed-time and resource-usage statistics. The user needs the facility to set a timer that will measure his task's execution time or the elapsed calendar time. Each task has eight interval timers associated exclusively with that task that are accessible to the user.

STIMER	sets a software interval timer, measuring either task execution time or real time, and indicates what action should be taken when that specified time interval has elapsed.
TTIMER	tests an interval timer previously set by the STIMER macro and indicates the time remaining in that interval. It can also be used to cancel a previously specified timer setting.
REDTIM	provides time as a double precision, fixed-point number in microseconds. In TSS the epoch is March 1, 1900.
EBCDTIME	converts system-maintained time into specified EBCDIC formats. The time is expressed in some combination of years, months, days, hours, minutes, seconds, tenths of seconds, and hundredths of seconds.

SYSTEM ORIENTED MACROS

In addition to those system-oriented user macro instructions already indicated within the various functional groupings, several other such macro instructions are available:

AWAIT	tests for completion of an event and returns control to the task if the event is completed, or places the task in a delay state from which it will be removed when any task interruption occurs.
VSEND	sends a message from one task to another. The message is queued on the recipient task status index (TSI) as an external interrupt.
USAGE	causes resource statistics for a task to be made available for processing by a user program.

XTRTM	extracts and examines the total accumulated CPU time of the issuing task from the extended task status index (XTSI) of the task.
HASH	provides a hash value for a name.
LPCEDIT	invokes the editor, which can be used by a language processor controller for input of source statements.
LPCINIT	identifies the program which issues it as a language processor controller and initializes the editor for later use.
LIBESRCH	determines if a specified object module is to be found in any of the job libraries and if so, which library.
CHDERMAC	generates messages pertaining to errors encountered during macro expansion.
CHDVAL	determines the type code of a parameter during macro expansion.
CHDPSECT	changes the name of the current control section of an assembly to the name of the first PSECT for that assembly. If no PSECT exists, a branch to a specified location is generated. either as the actual message or as the identification code of a message in the system or user message file.
ENQ	requests exclusive/shared read only access to a resource.
DEQ	releases a previously issued resource access request.

PART 2: MACRO INSTRUCTIONS

This part describes the macro instructions supplied with TSS that are available to all users. Section 1 tells how they are described; section 2 describes the macro instructions, arranged alphabetically.

| The macro definitions for the macro instructions in this book can be
| primarily found in the system data set TSS*****.SYSMAC; a few are in
| TSS*****.ASMMAC.

SECTION 1: HOW MACRO INSTRUCTIONS ARE DESCRIBED

First, for a basic understanding of how macro instructions in this book are described, look at Figure 2. This figure may also serve as a quick reference when reading the description of a macro instruction. You may wish to tab it.

The information that follows supplements Figure 2.

MACRO INSTRUCTION FORMAT

Macro instructions, like assembler instructions, are written in this format:

Name	Operation	Operand	Comment

Name Field

The name field of the macro instruction may contain a symbol or may be blank. Normally, the symbol is the name associated with the first executable instruction of the macro expansion.

Operation Field

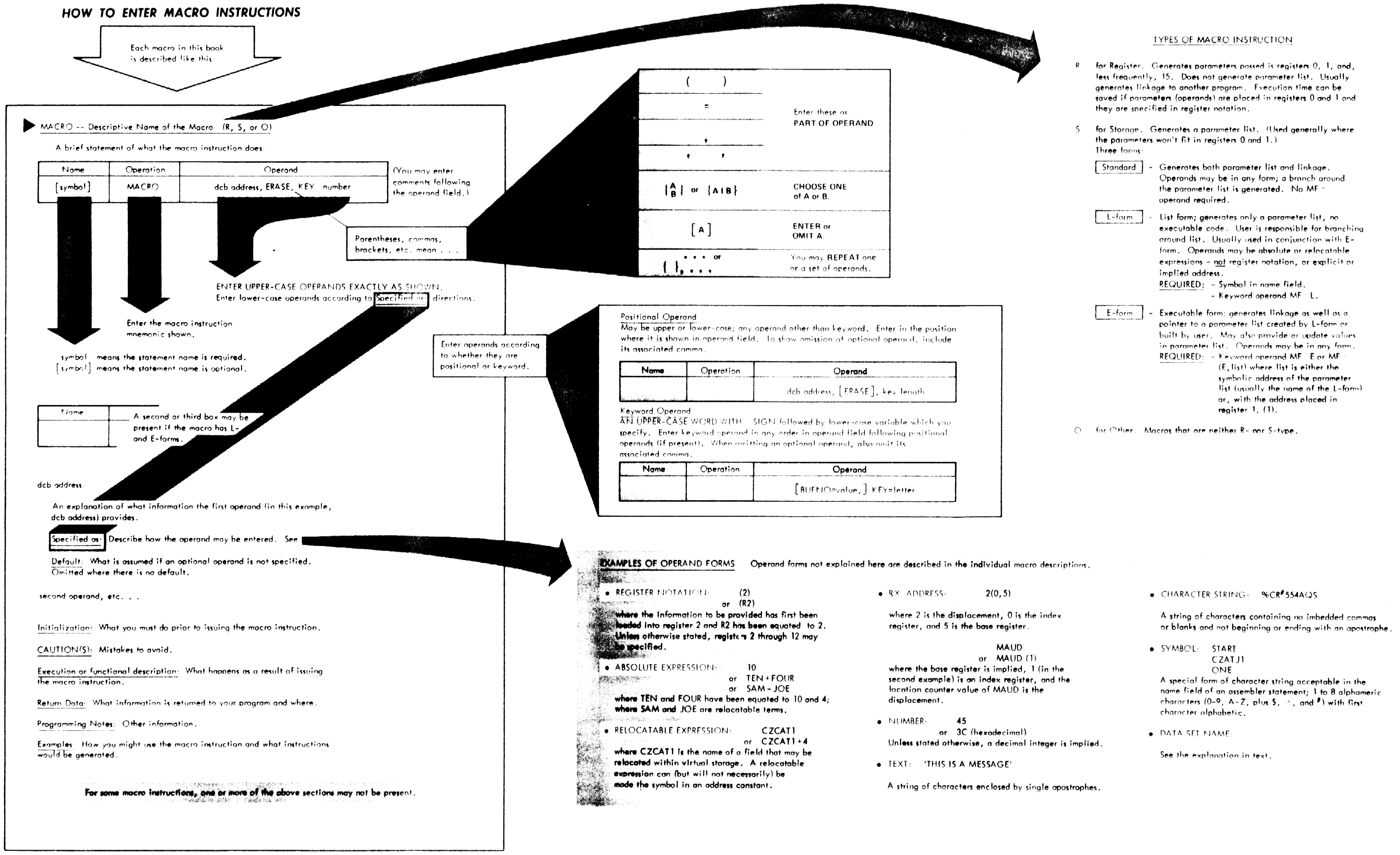
This field contains the mnemonic operation code of the macro instruction. The code may be a string of not more than eight alphameric characters, the first of which is alphabetic.

Operand Field

This field may contain no operands, or one or more operands separated by commas; the two types of operands are positional and keyword. Blanks may not be imbedded between operands.

Comment Field

This field is separated from the operand field by at least one blank. Comments may contain all valid characters in the character set, including blanks. In statements where an optional operand entry is omitted, or in statements which allow no operand but in which a comment entry is to be used, the absence of the operand entry is indicated by a comma preceded and followed by one or more blanks. In this publication, the comments field is not shown in the macro instruction formats.



HOW TO ENTER MACRO INSTRUCTIONS

Each macro in this book is described like this:

MACRO -- Descriptive Name of the Macro (R, S, or O)

A brief statement of what the macro instruction does.

Name	Operation	Operand
[symbol]	MACRO	dcb address, ERASE, KEY = number

(You may enter comments following the operand field.)

Parentheses, commas, brackets, etc. mean . . .

( )	Enter these as PART OF OPERAND
=	
,	
' '	
{A B} or {A | B}	CHOOSE ONE of A or B.
[A]	ENTER or OMIT A.
. . . or { }, . . .	You may REPEAT one or a set of operands.

ENTER UPPER-CASE OPERANDS EXACTLY AS SHOWN.
Enter lower-case operands according to Specified as: directions.

Enter the macro instruction mnemonic shown.

symbol means the statement name is required.
[symbol] means the statement name is optional.

Enter operands according to whether they are positional or keyword.

Positional Operand
May be upper or lower-case; any operand other than keyword. Enter in the position where it is shown in operand field. To show omission of optional operand, include its associated comma.

Name	Operation	Operand
		dcb address, [ERASE], key length

Keyword Operand
AN UPPER-CASE WORD WITH = SIGN followed by lower-case variable which you specify. Enter keyword operand in any order in operand field following positional operands (if present). When omitting an optional operand, also omit its associated comma.

Name	Operation	Operand
		[BUFNO=value,] KEY=letter

Name	

A second or third box may be present if the macro has L- and E-forms.

dcb address

An explanation of what information the first operand (in this example, dcb address) provides.

Specified as: Describe how the operand may be entered. See

Default: What is assumed if an optional operand is not specified. Omitted where there is no default.

second operand, etc. . .

Initialization: What you must do prior to issuing the macro instruction.

CAUTION(S): Mistakes to avoid.

Execution or functional description: What happens as a result of issuing the macro instruction.

Return Data: What information is returned to your program and where.

Programming Notes: Other information.

Examples: How you might use the macro instruction and what instructions would be generated.

For some macro instructions, one or more of the above sections may not be present.

TYPES OF MACRO INSTRUCTION

R for Register. Generates parameters passed is registers 0, 1, and, less frequently, 15. Does not generate parameter list. Usually generates linkage to another program. Execution time can be saved if parameters (operands) are placed in registers 0 and 1 and they are specified in register notation.

S for Storage. Generates a parameter list. (Used generally where the parameters won't fit in registers 0 and 1.)
Three forms:

Standard	- Generates both parameter list and linkage. Operands may be in any form; a branch around the parameter list is generated. No MF = operand required.
L-form	- List form; generates only a parameter list, no executable code. User is responsible for branching around list. Usually used in conjunction with E-form. Operands may be absolute or relocatable expressions - not register notation, or explicit or implied address. REQUIRED: - Symbol in name field. - Keyword operand MF = L.
E-form	- Executable form; generates linkage as well as a pointer to a parameter list created by L-form or built by user. May also provide or update values in parameter list. Operands may be in any form. REQUIRED: - Keyword operand MF = E or MF = (E, list) where list is either the symbolic address of the parameter list (usually the name of the L-form) or, with the address placed in register 1, (1).

O for Other. Macros that are neither R- nor S-type.

EXAMPLES OF OPERAND FORMS Operand forms not explained here are described in the individual macro descriptions.

- REGISTER NOTATION: (2)
  or (R2)
  where the information to be provided has first been loaded into register 2 and R2 has been equated to 2. Unless otherwise stated, registers 2 through 12 may be specified.
- ABSOLUTE EXPRESSION: 10
  or TEN + FOUR
  or SAM - JOE
  where TEN and FOUR have been equated to 10 and 4; where SAM and JOE are relocatable terms.
- RELOCATABLE EXPRESSION: CZCAT1
  or CZCAT1+4
  where CZCAT1 is the name of a field that may be relocated within virtual storage. A relocatable expression can (but will not necessarily) be made the symbol in an address constant.
- RX ADDRESS: 2(0,5)
  where 2 is the displacement, 0 is the index register, and 5 is the base register.
  MAUD
  or MAUD (1)
  where the base register is implied, 1 (in the second example) is an index register, and the location counter value of MAUD is the displacement.
- NUMBER: 45
  or 3C (hexadecimal)
  Unless stated otherwise, a decimal integer is implied.
- TEXT: 'THIS IS A MESSAGE'
  A string of characters enclosed by single apostrophes.
- CHARACTER STRING: %CR#554AQ5
  A string of characters containing no imbedded commas or blanks and not beginning or ending with an apostrophe.
- SYMBOL: START
  CZATJ1
  ONE
  A special form of character string acceptable in the name field of an assembler statement; 1 to 8 alphameric characters (0-9, A-Z, plus $, @, and #) with first character alphabetic.
- DATA SET NAME:
  See the explanation in text.

Figure 2. How to enter macro instructions

POSITIONAL OPERANDS: Positional operands are those that must be written in a specific position within the operand field. Assembly processing of positional operands is determined by the position they are assigned in the operand field. The positions of the operands are maintained by the separating commas; if an operand is omitted, the comma must nevertheless be supplied to maintain the position of succeeding operands. For example:

```
oper1,oper2,oper3
```

The three operands are processed in order from left to right. If the second operand is omitted, the operands are written:

```
oper1,,oper3
```

If the last positional operand or operands are omitted, delimiting commas need not be written. For example, if operands oper2 and oper3 are omitted, the operand field may be written:

```
oper1
```

KEYWORD OPERANDS: A keyword operand consists of a keyword, immediately followed by an equal sign and the value of the keyword. The keyword uniquely identifies the operand to the assembler; these operands may therefore be written in any sequence. Either

```
      BUFNO=20,BUFL=132

or    BUFL=132,BUFNO=20
```

may be written.

If keyword operands are omitted, their separating commas may also be omitted.

MIXED OPERANDS: An operand field may contain both positional and keyword operands; in this case, all positional operands must precede any keyword operands.

```
132,20,NA,KEY=A,CODE=NT
```

OMITTING OPERANDS FROM MIXED OPERAND FIELDS: The rules for omitting positional or keyword operands apply to mixed fields. In the example immediately above, if operands 20, NA, and KEY are omitted:

```
132,CODE=NT
```

If operands 132 and CODE are omitted:

```
,20,NA,KEY=A
```

OPERAND SUBLISTS: An operand sublist consists of one or more positional operands, separated by commas; the total list must be enclosed in parentheses. The entire string is considered as one operand in that it occupies a single position in the operand field or is associated with a single keyword. The contents of the string are processed in the same way as positional operands. These are operand sublists:

```
(A,B,C)
(A)
```

Note that sublist (A) consists of only one operand. When a macro instruction description shows that even one operand is written as an operand string, the enclosing parentheses must be written.

OPERAND STRINGS: In a number of macro instruction descriptions in this publication, the operand field, except perhaps for the operand specifying the macro form (MF), consists of a list of keyword and/or positional operands that are written as fields of a character string. The character string itself, enclosed in apostrophes, or the address of the string in storage, may be written in the operand field. The manner in which the address may be specified may depend upon the macro form.

If the operand is presented as a character string, the macro expansion places it in the assembled program followed by an end-of-message code, and loads a pointer to the string in register 1. If the operand field specifies the address of the character string, the expansion places that address in register 1. In the latter case, the user must define the operands elsewhere in the program and provide an end-of-message code.

If the user wishes to refer to and manipulate the operands in an operand string in coding, the address option of the operand is used, permitting the operand character string to be set up as a series of adjacent fields, each with its own label.

The string must end with X'27', which serves as an end-of-message code. Any unused space in each of the adjacent fields in the string must be filled with blanks to the maximum size of that field. Unlike other operand forms, all commas in an oplist operand must be coded, even if parameters are defaulted. EX1 and EX2 show how each of the string and address options for the operand field are used; the two examples have the same effect.

```
EX1       MACRO    'first operand,second operand'
EX2       MACRO    STRING
          .
          .
          .
STRING    DC       C'first operand'
OP2       DC       C'second operand'
          DC       X'27'
```

Macro Description Notational Symbols: Notational symbols in the operand field of macro instruction descriptions assist the user in showing how, when, and where an operand should be written; these symbols are themselves never written in the operand field. The notational symbols are: vertical stroke, |; hyphen, -; braces, { }; brackets, [ ]; ellipsis, ...; and underscore, __.

1. Vertical stroke means "exclusive or." For example, A|B means that either the character A or the character B, but not both, may be written. Alternatives are also indicated by operands being aligned vertically, as shown in the next paragraph.

2. Braces denote grouping. They are used most often to group alternative operands or alternative operand forms. For instance, the following two operand descriptions are equivalent:

   {INPUT|OUTPUT}

   $\left\{\begin{array}{l}\text{INPUT}\\\text{OUTPUT}\end{array}\right\}$

3. Brackets denote options. Information enclosed in brackets may either be omitted or written in the macro instruction, depending on the service to be performed. In the following case, the operand of the EXAMPLE macro instruction is optional and need not be supplied.

Name	Operation	Operand
[symbol]	EXAMPLE	[mode]

4. An ellipsis denotes that the preceding syntactical unit can be repeated one or more times in succession. If the syntactical unit consists of one term, it is followed by a comma and an ellipsis; for example,

   dcb address,...

   indicates that the term dcb address can be repeated, with a comma separating each term from the succeeding term, but with no comma after the last term.

   If the syntactical unit to be repeated consists of more than one term, it is enclosed in braces to indicate the terms that may be repeated, and the comma and elipsis are placed outside the braces; for example:

   {dcb address,option},...

   indicates that dcb address,option can be repeated with commas separating each term. No comma is placed after the last term.

5. Uppercase (capital) letters indicate the portion of the operand that must be written exactly as shown.

   DISP=NEW

6. Lowercase letters indicate the portion of the operand that is to be replaced by a permissible value. The macro description will specify the permissible values. For example:

   spacing

   Specified as: 1 or 2

   length

   Specified as: A relocatable expression, or register notation (2 through 12).

   In the first example, either 1 or 2 may be coded as the complete operand. In the second operand, 'length' could be replaced by MSGLEN (a relocatable expression), or by (3) as the complete operand.

7. Commas and parentheses must be written as shown in an operand field. They are delimiters, not notational symbols.

Macro Descriptions

The macro descriptions specify the form in which each operand may be written.

For the macro instruction descriptions in this publication, each positional operand is specified by a meaningful name or phrase, as illustrated:

Name	Operation	Operand
[symbol]	EXAMPLE	dcb address

Each keyword operand is specified by the keyword, an equal sign, and a meaningful name or phrase, as illustrated:

Name	Operation	Operand
[symbol]	EXAMPLE	DSNAME=name of data set

In describing the manner in which an operand may be specified, the following terms may be used.

<u>Relocatable Expression</u>: A relocatable expression represents the name assigned to a field that may be relocated within virtual storage during program execution. (In contrast, an absolute expression represents a field that may not be so relocated.) The value of a relocatable expression changes by n if the program in which it appears is relocated n bytes from its originally assigned storage area. All relocatable expressions must have positive values. A relocatable expression may be a single relocatable term. Also, a relocatable expression may contain multiple relocatable terms, or a combination of relocatable terms with absolute terms, under these conditions:

1. The expression must contain an <u>odd number</u> of relocatable terms.

2. All relocatable terms except one must be paired; pairing is described later in "Absolute Expression."

3. The unpaired term must not be directly preceded by a minus sign.

4. A relocatable term must not enter into a multiply or divide operation.

A relocatable expression reduces to a single relocatable value, which is the value of the odd relocatable term adjusted by the values represented by the absolute terms or paired relocatable terms associated with it. The relocatability attribute is that of the odd relocatable term. Complex relocatable expressions are permitted (refer to <u>Assembler Language</u>).

In the following examples of relocatable expressions, SAM, JOE, and FRANK are in the same control section and are relocatable; PT is absolute.

```
SAM
SAM-JOE+FRANK
JOE-PT*5
SAM+3
```

Note that SAM-JOE is not relocatable, because the difference between two relocatable addresses is constant.

<u>Register Notation</u>: This is written as an absolute expression enclosed in parentheses. The absolute expression, when evaluated, must be some value 2 through 12, indicating the corresponding general purpose register. In these examples of register notation, SAM and JOE are relocatable and have the same relocatability attribute and PAL is absolute:

```
(5)         -indicates register 5
(PAL)
(PAL+3)
(SAM-JOE)   -invalid
```

The absolute expression (SAM-JOE) is invalid because it contains paired relocatable terms. See Absolute Expression below.

When register notation is used for an operand, the indicated register must be loaded with the desired value before execution of the macro instruction. No register other than those stated as being permissible should be specified; the integrity of other registers cannot be relied upon. See "Using Parameter Registers" under R-Type Macro Instructions below.

RX Address:
(1) may be explicit, written in the same form as an assembler language operand:

```
a(b,c)
■ ■ ■
| | |
| | base register
| |
| index register
|
displacement
```

Examples are:

```
2(0,5)
0(2,4)
```

(2) may be implied (indexed), written as a symbol, optionally indexed by an index register. For example:

```
INITIAL
ALPMAY(4)
```

Note that ALPMAY is indexed by register 4.

Symbol: This may be a symbolic address (that is, a single relocatable term), such as the symbolic name of an instruction in an assembler language program, or it may be a character string used for identification, not location (such as the ddname parameter of a DCB macro instruction).

In TSS, the alphabetic characters are the letters A-Z, plus $, @, #: the alphameric characters are the alphabetic characters plus the digits 0-9.

The symbol is written as a string of as many as eight alphameric characters, the first of which is alphabetic. Embedded commas and blanks are not permitted. Symbols beginning with the characters CHD and SYS may not be used, since symbols beginning with those characters are reserved for system use. Examples of symbols are:

```
DDNAME1
ROGER
LOOP12
START
#1
```

Number: Unless stated otherwise, this will imply a decimal integer.

Absolute Expression: This may be an absolute term or any arithmetic combination of absolute terms. An absolute term may be an absolute symbol or any self-defining term. All arithmetic operations are permitted between absolute terms. An absolute expression may contain relocatable terms alone or in combination with absolute terms, provided that these conditions are met:

1. The expression must contain an even number of relocatable terms.

2. The relocatable terms must be paired.

3. Each pair of terms must have the same relocatability attribute; that is, they must appear in the same control section of an assembly.

4. Each pair must consist of terms with opposite signs. The paired terms do not need to be adjacent; for example, RT+AT-ST, where RT and ST are relocatable with the same relocatability attribute and AT is absolute.

5. A relocatable term must not enter into a multiply or divide operation.

Pairing of relocatable terms (with opposite signs and the same relocatability attribute) cancels the effect of relocation. The value represented by the paired terms remains constant, regardless of program relocation. It should be noted that absolute expressions composed of paired relocatable terms should not be used as macro operands since the attempt by a macro definition to use them in an AIF, SETA and SETB will result in an error. The assembler does not assign location counter values to relocatable terms until all macro expansions are completed.

Example: In the absolute expression A-Y+X, the term A is absolute, and the terms X and Y are relocatable with the same relocatability attribute. If A equals 50, Y equals 25, and X equals 10, the value of the expression becomes 35. If X and Y are relocated by a factor of 100, their values become 125 and 110. However, the expression still evaluates as 35 (50-125+110=35).

An absolute expression reduces to a single absolute value.

In these examples of absolute expressions, JOE and SAM are relocatable and defined in the same control section; BERNY and DAVE are absolute:

```
331
DAVE
BERNY+DAVE-83
JOE-SAM
DAVE*4+BERNY
```

Data Set Name: This is the name of one data set or a group of data sets. The rules for writing data set names are presented below; the types of names that may be written for each macro instruction are identified under the description of each macro instruction.

Fully qualified name uniquely identifies one data set.

1. A stand-alone data set name identifies a data set that is not a member of a partitioned data set nor a generation of a generation data group. The name of a stand-alone data set is written as a series of symbols separated by periods. For example:

```
DATASET.TRIAL.TEST1
TERI.ROGER.LAURIE
A.B.C
```

The rightmost symbol is the data set's simple name (TEST1, LAURIE, and C above); the other symbols are qualifiers. In TSS, for cataloging purposes, the maximum number of characters in a data set name, including periods, is 35. The maximum number of one-character qualifiers for a one-character name is 17.

<u>Note</u>: Data set names created under the IBM OS, or OS/VS System can contain a maximum of 44 characters; if data sets with names greater than 35 characters are to be cataloged in TSS, the user should employ the renaming facility of the CAT macro instruction or CATALOG command to define a suitable TSS name.

2. <u>Partitioned Data Set and Member Name</u> identifies a data set that combines individual data sets, called members, into a single data set. The partitioned organization allows the user to refer to either the entire data set or to an individual member of the data set.

   - The rules for writing the name of a partitioned data set are the same as for writing the name of a stand-alone data set. The parentheses and member name are merely considered as an appendage to that name.

   - The rules for writing a member name vary with each macro instruction that can manipulate members. Sometimes (as in LOAD and DELETE) only the simple member name (a symbol) is written. The full name is not required because the user has indirectly defined the partitioned data set (library) in which the module resides by assuring that the library is on the program library list prior to issuing those commands. The user could write

     ```
     LOAD SORTR
     ```

     if he has previously entered SORTR in a library currently on the program library list.

     In other macro instructions (for example, COPYDS), the user must give the fully qualified member name. This consists of the name of the partitioned data set suffixed by the simple member name in parentheses. For example:

     ```
     HQW(ONETRY)
     G.H.AB(H)
     ```

     Here HQW and G.H.AB are partitioned data sets with members ONETRY and H, respectively.

3. <u>Generation Names</u> identify data sets which are part of a generation data group. These data sets can be referred to on an absolute or relative basis:

   a. <u>Absolute Generation Names</u> are written as the name of the generation data group followed by a period and the characters GxxxxVyy, where xxxx is a four-digit decimal generation number, and yy is a two-digit decimal version number. For example:

      ```
      HURST.LINER.TT.G0001V00
      HJ.LA4.WW.G0003V01
      HARQ.G0147V03
      ```

      The characters GxxxxVyy are considered a fixed part of the overall name. The name of the generation data group (for example, HURST.LINER.TT) is a partially qualified name applicable to all generations in the group.

If the generation is a partitioned data set, a member (for example, JOE) within that data set is referred to as follows:

```
A.B.C.GxxxxVyy(JOE)
```

b. Relative Generation Names are written as the name of the generation data group followed by the appropriate relative generation number enclosed in parentheses, such as:

```
G.D.G(0)
```

The relative generation number of the most recent generation is (0); the generation just prior to that is (-1); the one before that is (-2), etc.; and a new generation to be added is (+1). For example:

```
GOST.UU.L19P(+1)
GOST.UU.L19P(-3)
MRQ.T.L5.SWIM(0)
```

If the generation is a partitioned data set, a member within that data set is referred to as follows:

```
SEAT(-3)(JOE)
```

where JOE is the member in question.

Partially Qualified Names refer to all data sets having a given partially qualified name as their common higher-order qualifier.

1. Generation Data Group Name is the name that is common to each generation in the group. Generation data group names are restricted to a maximum of 26 characters including periods.

2. Other Partially Qualified Names can also be used to refer to two or more data sets. For example, the partially qualified name GO.AB14 can be used to refer to both of the following data sets: GO.AB14.A and GO.AB14.B. If these were the only two of a user's data sets with the same higher-order qualifier, GO.AB14, and he wished to erase them both, he could do so merely by specifying GO.AB14 in the ERASE macro instruction.

Alphameric Characters: An alphameric-character operand is written as a string of alphameric characters, the first of which need not be alphabetic. For example:

```
A00764
10E0D4
```

The limit on the number of characters is given in the description of each macro instruction in which it is used.

TYPES OF MACRO INSTRUCTIONS

Most system macro instructions are either R-type (register) or S-type (storage). In this publication, the letter (R) or (S) follows the name of each macro instruction description to identify its type; macro instructions that are neither R- nor S-type, referred to as "other" type macro instructions, are identified by (O).

Some macro instructions generate literals in their expansions. Consequently, the rules for literal pool coverage must be followed. Refer to Assembler Language, Section 2, "Terms and Expressions." Parameters can be contained in the two parameter registers, 0 and 1.

An R-type macro instruction does not generate a parameter list; the parameters are placed in the parameter registers by instructions in the macro expansions. Execution time can be saved if the user places the data in the parameter registers before executing an R-type macro instruction, and uses register notation to specify the operands in the macro instruction.

Address operands in R-type macro instructions are always specified as | an RX address, or in register notation. This arrangement allows the user to employ indexing, although the addresses passed in R-type macro instructions must be properly covered; that is, the base register used for the passed address must contain the proper value to ensure that the address refers to the desired location in virtual storage. For example, assume an R-type macro instruction, RTYPE, which will contain an address "area" in register 1 and the "length" of that area in register 0. Its external macro description would be:

Name	Operation	Operand
[symbol]	RTYPE	area,length

area
specifies an address.

Specified as: Register notation (1 through 12), in which case the
| address must be placed in the register before execution of the
| macro instruction; an RX address

length
specifies a length.

Specified as: Register notation (0 or 2 through 12), in which case the length must be placed in the register before execution of the macro instruction; or an absolute expression.

Using Parameter Registers: The user's problem program might be written so that one or both of the parameters are already in the proper parameter register when the macro instruction is issued. In this case, (1) or (0) is written as the operand. Registers 1 and 0 cannot be used in a macro instruction unless their use is mentioned in the "Specified as" paragraph for the operand.

S-Type Macro Instructions

An S-type (storage) macro instruction is used when the number of parameters to be passed to the called routine cannot be contained in the two parameter registers. These parameters are placed in a parameter list whose address is passed to the called routine in register 1.

There are three forms of the S-type macro instruction:

1. The standard form (in which the MF= operand is defaulted)

2. The L-form (parameter list only - specified as MF=L)

3. The E-form (executable code only - specified as MF=E)

S-TYPE, STANDARD FORM: This form of macro instruction generates both the parameter list required by the called routine and the linkage to that routine. If the S-type macro instruction is coded in a module that has a PSECT, the parameter list is generated in the PSECT. In this case, the PSECT must be properly covered by a base register. If an S-type macro instruction is coded in a PSECT, or if it is coded in the

CSECT of a module that has no PSECT, the parameter list is generated in-line and coding is generated to branch around it.

Address operands in S-type standard form macro instructions are always specified as register notation or a relocatable expression. Hence, they may not be indexed, and the user's problem program does not need to provide cover registers. As an example, assume an S-type macro instruction, STYPE, that expects the addresses of two storage areas, "input" and "output," and the "length" of those areas. Its external macro description might be:

Name	Operation	Operand
[symbol]	STYPE	input,output,length

input
: specifies the input area.

  Specified as: Register notation or a relocatable expression.

output
: specifies the output area.

  Specified as: Register notation or a relocatable expression.

length
: specifies the length in bytes of the input and output areas.

  Specified as: Register notation or an absolute expression.

S-TYPE, L-FORM: This form of macro instruction creates a parameter list. E-form macro instructions then link to the service routine and point to the parameter list that is generated by the L-form macro instruction. The assembler recognizes an L-form macro instruction by the keyword operand MF=L in its operand field.

Because the L-form macro instruction generates only a parameter list, operand forms that require executable code, such as register notation, are prohibited. The external description of the L-form S-type macro instruction becomes (compare with the standard form):

Name	Operation	Operand
symbol	SLTYPE	[input,][output,][length,]MF=L

input
: specifies the address of the input area.

  Specified as: A relocatable expression.

output
: specifies the address of the output area.

  Specified as: A relocatable expression.

length
: specifies the length in bytes of the input and output areas.

  Specified as: An absolute expression.

The name field is required in the L-form because it usually becomes the label of the generated parameter list and is referred to by the E-form.

All operands of an L-form macro instruction (except MF=L) are optional. Operands that are omitted in the L-form are assumed to be supplied in the E-form macro instruction.

The L-form macro instruction generates the parameter list at the place the macro instruction is encountered. Because the L-form expansions contain no executable instructions, they should be placed in the program so that they do not receive control; for example, among the DS and DC instructions. An L-form macro instruction should never be written in a read-only control section.

S-TYPE, E-FORM: A parameter list created by an L-form macro instruction, or by any other means, may be referred to by an E-form macro instruction. The user can update a parameter list by supplying operands in the E-form macro instruction. The assembler recognizes an E-form macro instruction by the presence of the keyword operand in its operand field:

```
    MF=(E,list)
```

Here, list should specify the location of the parameter list to be used by the E-form macro instruction. If register notation is specified, the register should be loaded (before execution of the macro instruction) with the address of the L-form parameter list. The symbol in the name field of an L-form macro instruction becomes the name of the parameter list.

The E-form allows the user to index addresses; however, proper cover registers must be provided. The external description of the E-form S-type macro instruction becomes (compare with the standard and L-forms):

Name	Operation	Operand
[symbol]	SETYPE	[input,][output,][length,]MF=(E,list)

input
    specifies the input area.

| Specified as: Register notation (2 through 12); an RX address

output
    specifies the output area.

| Specified as: Register notation (2 through 12); an RX address

length
    specifies the length in bytes of the input and output areas.

    Specified as: Register notation (2 through 12); or an absolute expression.

Each operand except the last is optional. The position of positional operands supplied in the E-form macro instruction causes the generation of values that replace the corresponding parameters in the parameter list of the L-form macro instruction.

<u>Other Macro Instructions (O-Type)</u>

The system macro instructions that cannot be classified as either R-type or S-type are referred to as "other", and identified by (O) in the macro instruction descriptions. For example, the SAVE macro instruction does not produce parameters that pass to a called program. Its expansion results in instructions in the user's program that completely perform the requested service.

CONDITIONAL ASSEMBLY OF MACRO INSTRUCTIONS

Some macro instructions can be assembled with non-privileged code only, some with privileged code only, and some can be assembled with either kind of code. However, the DCLASS macro instruction may be used to assemble macro instructions as desired. For instance, a nonprivileged U-authority programmer can assemble privileged code by first issuing a DCLASS PRIVILEGED macro instruction and following the privileged code with a DCLASS USER macro instruction (if subsequent macro instructions that can only be assembled in privileged code are to be assembled correctly, the DCLASS USER macro instruction is issued after the last of these).

Although a privileged macro instruction may be assembled in a privileged or nonprivileged control section by first issuing a DCLASS PRIVILEGED macro instruction, the code thus assembled can only be executed in a privileged module. Those macro instructions that are restricted to one kind of code or the other return error messages when an attempt is made to assemble with the inappropriate DCLASS setting. The descriptions of each macro instruction in Section 2 include cautions (as appropriate) about the need to issue DCLASS macro instructions, and also about the need to provide save areas, as discussed in Appendix E. A summary of those macro instructions requiring the use of appropriate DCLASS macro instructions is given in Appendix M, together with a further discussion of this topic.

MACRO INSTRUCTION GENERATION OF LITERALS

The TSS assembler places literals in a module's first declared PSECT, if one has been declared. If no PSECT has been declared, address literals are treated as any other literals: i.e., placed in whatever literal pool is proper.

If a literal is generated, the user must be sure that the location containing the literal is covered by a base register at the time the macro instruction is executed.

SECTION 2: MACRO INSTRUCTION DESCRIPTIONS

Figure 2 shows how the macro instructions in this section are described. The complete descriptions for all user macros are given alphabetically in this section, except for the TAMII macros which appear in Appendix N.

Macro instructions intended for a special class of users are described in System Programmer's Guide, Manager's and Administrator's Guide, and Multiterminal Task Programming and Operation.

ABEND -- Abnormal Task End (S)

The ABEND macro instruction serves as an error exit for an assembled program, either to terminate execution of the program or to eliminate the user's current task from the system, and then return control to the user in command mode.

Standard and E-form:

Name	Operation	Operand
[symbol]	ABEND	exit type,{address of message|'message'},
		{message id address|'message id'}
		{[,(parameter address,...)]|'parameter'}
		[,MF=(E,list)]

L-form:

Name	Operation	Operand
symbol	ABEND	[exit type],[{address of memessage|'message'},]
		[{message id address|'message id'}]
		[{(parameter address,...)|'parameter'},MF=L]

Note: The name field is required with MF=L. Any required operand that is omitted from the L-form must be supplied in the E-form. The operands specified in the E-form will overlay those specified in the L-form. If the MF operand is omitted, the standard form is assumed.

exit type
    specifies the return type or completion code.

    Specified as: (0), 1, 2, 3, or 4. Exit type 1 causes return either to the conversational user at his terminal or, in nonconversational mode, to a data set with the data definition name TSKABEND, to retrieve commands for execution. In either case, the task is returned to command mode. Exit type 2 terminates the user's task. If the task is conversational, a new task is created for the user and turned over to him as though he had just logged on. Exit type 3 is similar to exit type 2 in that it terminates the user's task, but it does not create a new task or return control to the user. The user's terminal is deactivated. Exit type 4, in privileged programs only, is similar to exit type 2, except that it is not possible to send a message to the user, since the terminal is being held, and the transmission line is physically disconnected and disabled.

    If (0) is specified, the exit type must be loaded into register 0 before execution of this macro instruction.

address of message
    specifies the location containing the message to be issued (see below). The address points to a one-byte length field that pre-

cedes a field containing the message text (the length is in hexadecimal bytes, not to exceed 257).

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, in register notation (1 through 12) or an RX address.

message
specifies the message text to be issued to SYSOUT when the ABEND macro instruction is executed. If this operand is specified, neither of the last two operands can be specified.

Specified as: The text of the message, written as a character string enclosed in apostrophes (embedded blanks and special characters are permitted).

message id address
specifies the identification of the message to be issued to SYSOUT. The identification is an 8-byte code that identifies a message in the system (in SYSMLF). If this operand is specified, the message operand cannot be specified. The address must point to an 8-byte field containing the identification, left-aligned and padded with blanks.

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, in register notation (2 through 12) or an RX address.

message id
specifies the identification of the message to be issued to SYSOUT.

Specified as: The text of the message identification, enclosed in apostrophes.

parameter address
specifies the location of a parameter (see below) that modifies the message being displayed. This operand is only used if the message identification is specified. See the Note below.

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, in register notation (2 through 12), or an RX address.

parameter
specifies information that is to be used to complete or alter the message being displayed at the terminal. This operand is only specified if the message identification is specified.

Specified as: A character string enclosed in apostrophes.

Note: The number of parameters or parameter addresses can be intermingled but cannot exceed 20.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: ABEND with exit type 1 returns the task to command mode and removes any previously invoked user control of attention interruptions.

Following an abnormal termination, the VPSW and the general registers are displayed on SYSOUT, together with the message text specified in the

| ABEND operand. If different messages are specified for different exits, the error exit taken can easily be identified.

The TSKABEND data set might contain a sequence of PCS commands that obtains a selective dump of the program before terminating the task with a LOGOFF command.

If an error occurs during the processing of an exit type 1 condition, the ABEND procedure is reinvoked, and the error is processed as an exit type 2 condition.

Examples: The user wishes to provide an error exit if his program encounters trouble on one path. He includes in that path the ABEND macro instruction:

```
ERROR    ABEND     1,'ABEND BECAUSE TROUBLE IN PATH N'
```

The user wants to provide messages for different error conditions. For his first error condition, he provides the message:

```
ERR1     DC        AL1(L'TEXT1)
TEXT1    DC        C'ABEND FOR INCOMPLETE PATH'
```

In the coding path that discovers this first error condition, he includes:

```
         LA        1,ERR1
         B         EREX
EREX     ABEND     1,(1)    COMMON ERROR EXIT
```

ADCON -- Generate an Adcon Group (O)

The ADCON macro instruction generates a group of address constants - an adcon group - (see the Programming Notes) for use by a CALL, LOAD, or DELETE macro instruction.

Name	Operation	Operand
symbol	ADCON	type of adcon group [,EP=entry point] [,LDERR={CODE|ERR}] [,DELOPT={SMO|SDM}] [,HSHTAB={XPOS|NOPM}]

Note: A symbol is required in the name field.

type of adcon group
specifies the type of adcon group to be generated.

Specified as: One of the following codes.

Code	Meaning
CALL	An explicit adcon group is generated for use by the CALL macro instruction.
LOAD	An explicit adcon group is generated for use by the LOAD macro instruction.
DELETE	An adcon group is generated for use by the DELETE macro instruction.
IMPLICIT	An implicit adcon group is generated using the externally defined symbol specified in the EP operand.

INTERNAL An implicit adcon group is generated using the internally defined symbol specified in the EP operand.

EP=

specifies the entry point of the module to which the adcon group refers. For an INTERNAL adcon group, the R-value indicates the origin of the control section containing the ADCON macro instruction. An ADCON macro instruction that specifies INTERNAL as the type must consequently not be written in an unnamed control section. Refer to the ARM and ADCOND macro instruction descriptions.

Specified as: A symbol (one to eight alphameric characters, the first of which is alphabetic).

Default: If EP is omitted, eight blank characters are used as the entry point name for CALL, LOAD, and DELETE adcon groups, whereas zero is used as the entry point address and R-value for IMPLICIT and INTERNAL adcon groups.

LDERR=

specifies whether the dynamic loader is to take an error exit or to present a return code if the specified module cannot be loaded. The LDERR operand may be used only if the type of adcon group is LOAD or CALL.

Specified as: CODE or ERR. If CODE is specified, ADCON sets bit ADCC2CB7 of the ADCC2C control byte to 1; the dynamic loader will then store a return code of X'07' in the ADCC2C control byte if an error is encountered while attempting to load the module. If ERR is specified or if the LDERR operand is omitted, ADCON sets bit ADCC2CB7 to 0; the dynamic loader will initiate "load error procedure" when the specified module cannot be loaded.

Default: ERR

DELOPT=

specifies the DELETE option desired. The DELOPT operand may be used only with a type operand that specifies DELETE.

Specified as: SMO or SDM. If SMO is specified, ADCON sets bit ADCC3DB7 of the ADCC3D control byte to 1; the dynamic loader will then attempt to delete only the specified module. If SDM is specified or if the DELOPT operand is omitted, ADCON sets bit ADCC3DB7 to 0; the dynamic loader will then attempt to delete all modules on which the specified module depends, as well as the specified module itself.

Default: SDM

HSHTAB=

specifies whether the system or user search chains are to be used to locate external names.

Specified as: XPOS or NORM. Normally (NORM), the system search chain is used when ADCON is issued in a privileged program and the user chain is used when ADCON is issued in a nonprivileged program. The opposite chain is used if XPOS is specified.

Default: NORM

Cautions: Although the EP operand may be omitted from the ADCON macro instruction, the entry point must eventually be supplied to the appropriate fields of the adcon group before the adcon group is actually used. ADCON cannot be specified within the first twelve bytes of a control section.

<u>Programming Notes</u>: An explicit adcon group is altered the first time a CALL or LOAD macro instruction refers to it. In this altered state, the adcon group is said to be disarmed; before being altered, it is said to be armed. The ADCON macro instruction may be used to generate a fully armed explicit adcon group having all control bytes and the entry point name generated with the desired values. The user may, however, want to complete arming by supplying the entry point name or control byte settings after the adcon group is generated. In any case, an explicit adcon group must be fully armed the first time it is used by a LOAD or load type-E CALL macro instruction.

Once an adcon group has been disarmed during loading or calling of a program, it may subsequently be used in that state only for one purpose and under certain conditions. If the program that was loaded or called has not been deleted, and if the adcon group used in its loading or calling has not been modified either by the ARM macro instruction or by the user's own code, the same adcon group may be used in subsequent calls to the same program. A disarmed adcon group may be made available for the following purposes only if it is rearmed by means of the ARM macro instruction:

- Calling or loading the same program again after it has been deleted.
- Calling the same program again after the adcon group referred to has been rearmed for a different program.
- Calling or loading a different program.

Note that an explicit adcon group generated for use by the LOAD macro instruction must not be used by the CALL macro instruction and vice versa, except in the following situation. An explicit adcon group that is used to load a program may be used in subsequent calls to the loaded program, if the explicit adcon group is not subsequently modified either by ARM or by the user's own code, and if the loaded program is not subsequently deleted.

If the user issues ADCON macro instructions, the V-con and R-con pair are located at a displacement of 12 from the label used for the ADCON macro instruction.

The user may refer directly to certain fields of adcon groups of any type. These fields are described below; no other fields can ever be altered directly by the user. The name for each field or bit position is the name provided by the ADCOND macro instruction. All references to adcon group fields and bit positions must use these names.

<u>EXPLICIT ADCON GROUPS FOR USE WITH LOAD OR CALL MACRO INSTRUCTIONS</u>:

Field Name		
For LOAD	For CALL	Meaning
ADCC1L	ADCC1C	Control byte 1
ADCC1LB7	ADCC1CB7	Bit of control byte 1; specifies the type of explicit adcon group; bit is 0 for LOAD; 1 for CALL adcon groups
ADCC2L	ADCC2C	Control byte 2
ADCC2LB7	ADCC2CB7	Bit of control byte 2: corresponds to the LDREF operand
ADCPNAM	ADCPNAM	Eight-byte field containing as a character constant the name of the program to be loaded or called

<u>DELETE ADCON GROUP</u>

Field Name	Meaning
ADCC3D	Control byte 3
ADCC3DB7	Bit of control byte 3; corresponds to DELOPT operand
ADCC4D	Control byte 4 in which the dynamic loader places the return code indicating results of a DELETE request; a return code of X'00' indicates successful deletion; a code X'04' indicates no deletion took place because the module defining the specified EP symbol was not present in the user's virtual storage when the request for deletion was given; a code X'08' indicates no deletion took place because of other outstanding references to the specified program
ADCPNAMD	Eight-byte field containing as a character constant the name of program to be deleted

IMPLICIT ADCON GROUPS

Field Name	Meaning
ADCEP	A four-byte adcon, aligned on a fullword boundary, containing the entry point of the specified program (V-value)
ADCRV	A four-byte adcon, aligned on a fullword boundary, containing the R-value of the specified program

CAUTION: Because adcon groups must be capable of being changed, they must not be generated in read-only control sections.

Examples:

1. This coding sequence generates an implicit adcon group for calling EXNAM, an externally defined entry point name:

```
          LA       15,LEXNAM
          CALL     (15),,,E
          .
          .
          .
LEXNAM    ADCON    IMPLICIT,EP=EXNAM
```

2. This coding sequence generates a DELETE adcon group for deleting only EXNAM, the specified module. EXNAM is assumed to have been previously loaded.

```
          .
          .
          .
          DELETE   EPLOC=LEXNAM
          .
          .
          .
LEXNAM    ADCON    DELETE,EP=EXNAM,DELOPT=SMO
```

ADCOND -- Provide Symbolic Names for an Explicit Adcon Group (O)

The ADCOND macro instruction generates a dummy control section (DSECT) that provides symbolic names for the fields in an explicit adcon group. The name of the generated DSECT is CHAADC.

This DSECT permits symbolic access to the resolved V-type and R-type address constants that are placed in the explicit adcon group upon execution of a LOAD or explicit CALL macro instruction. The control byte C2, which directs the loader to a course of action, may also be accessed. For an explanation of the control byte, refer to the LOAD macro instruction in this section.

Name	Operation	Operand
[symbol]	ADCOND	

Note: A symbol present in the name field will not be generated. There are no operands.

CAUTION: The ADCOND macro instruction may be used only once in an assembly.

Programming Note: The C1 control byte is addressable by the following names: ADCC1C (for CALL) or ADCC1L (for LOAD). The C2 control byte is addressable as ADCC2C (for CALL) or ADCC2L (for LOAD). The C3 control byte for DELETE is addressable by the symbolic name ADCC3D; the C4 control byte for DELETE has the symbolic name ADCC4D. The symbolic name ADCVCON addresses the resolved V-type address constant. The symbolic name ADCRCON addresses the resolved R-type address constant. When ADCC1C is set to X'00', a LOAD explicit adcon group is implied, and when set to X'01', an explicit CALL adcon group is implied.

The macro instruction may appear at any point in a control section. However, if it is written at any location other than at the end of a control section, the original control section must be resumed.

Example: The following example illustrates how a program accesses a field in an explicit adcon group. The program alters the C2 byte so that the loader will return codes that indicate the action of the loader. Refer to the description of LOAD macro instruction.

The ADCON macro instruction generates an explicit adcon group for a LOAD. ARM readies the adcon group for use by a LOAD. The LA instruction places the address of the adcon group into register 5. A USING statement establishes a base register for CHAADC. The MVI instruction sets the C2 control byte to 1; this setting requests the loader to return codes when the adcon group is used by a LOAD.

```
                .
                .
                .
      RALPH   ADCON     LOAD
                .
                .
                .
              ARM       RALPH,SQROUT
              LA        5,RALPH
              USING     CHAADC,5
              MVI       ADCC2L,X'01'
                .
                .
                .
      SQROUT  DC        CL8'SQROUT'
              ADCOND
```

AETD -- Create an Attention Entry Table (S)

The AETD macro instruction enables the user to bypass the system attention interruption handler; by pressing the attention key during processing, he can enter a predefined user-coded routine to process the interruption. A number of routines may be provided to process interruptions and their entry points specified in the operands of the AETD macro instruction: the desired routine is then selected by pressing the attention key the number of times that corresponds to the position of the routine in the list of routines specified.

Standard, L- and E-forms:

Name	Operation	Operand
[symbol]	AETD	[({entry point name,save area name} ,...)] [,MF={L|(E,list)}]

Note: The name field is required with MF=L. If the MF operand is omitted, the standard form is assumed. The parameters specified in the E-form will overlay those specified in the L-form. The E-form may not specify more operands than are specified in the corresponding L-form.

For example:

```
          SUE       AETD  (ENTRYPTA,SAVEA),MF=L
                    AETD  (,SAVEB),MF=(E,SUE)
```

When the E-form of this macro instruction is executed, the save area specified in the L-form (SAVEA) will be replaced in the parameter list by the save area specified in the E-form (SAVEB).

entry point name
: specifies the symbolic entry point name of a routine to be entered upon pressing the attention key at the terminal.

  Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

save area name
: specifies the symbolic name of a 21-word save area that is to be associated with the routine whose entry point is specified by the first operand. The 21-word save area is provided in addition to the standard 19-word save area (which must be provided in order to conform to standard linkage conventions). The two additional words in the 21-word save area are for saving the VPSW.

  Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

Note: If AETD is issued with neither of the above operands, any previously-defined attention entry tables are disconnected and the system resumes handling the attention interruptions. The system attention interruption handler can be invoked as an option by including a blank entry in the AET (see below).

| Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The AETD macro instruction generates a table containing the addresses of routines that are to be given control when a user presses the attention key a specified number of times. Thus, a user may specify, by the number of times the attention key is pressed, which routine is to be entered. The first time the attention key is pressed, the user's program execution is interrupted and procedure 1 in the table will be initiated; if he presses the attention key a second time <u>before procedure 1 has been completed</u>, he will enter procedure 2, and if he presses the attention key a third time before <u>procedure 2 has been completed</u>, he will enter procedure 3, and so on for as many predefined procedures as desired. Procedures specified in this manner are generally used to communicate with the user's terminal, thus allowing program modification at execution time. In order for the user to proceed from the first level to the second level, each routine executed by the entered level must delay, for example by communicating with the user's terminal, because the system does not provide a delay in level processing to allow the user to press the attention key more than once. If the delay is not provided by the user's coded routine, then on most central processors the user will not be able to press the attention key fast enough to proceed beyond level 1.

The user might employ the AETD macro instruction to pass control to any user-provided control systems, or to provide partial backup in a current task so that an error situation does not have to cause the task to be reconstructed from scratch. It can be used to predefine simple automatic debugging procedures by using PCS commands in the AETD attention handling routines.

When the attention key is pressed, registers 0-15 are stored in the specified save area. Registers 2-12 are passed to the user routine invoked.

The table that is generated (Attention Entry Table, AET) consists of three words containing V-type and R-type address constants and the save area addresses for each attention handling routine that a user has specified. Any null operand pair causes three words containing binary zeros to be created in the table. Entries are generated in the same order as given in the operands.

If AETD is issued with no operand, the current table (AET), if one was previously defined, is disconnected from the system and the system attention handling routines are invoked for subsequent processing of attention interruptions.

If the AETD macro instruction is issued with operands, but the parentheses around the operands are missing, the diagnostic message "PARENTHESES ENCLOSING OPERANDS ARE REQUIRED" will be issued with a severity code of 2.

Error recovery during execution of an attention interruption servicing routine can be accomplished by pressing the ATTN key a number of times corresponding to a blank entry in the AET table (see "Blank AET Entries" below). This causes control to be passed to the Command System, which prompts the user for additional input. The user can then:

- Enter commands in an effort to recover from the error,
- Press the ATTN key to continue with his next sequential AET entry, or
- AETD may be specified in a user program that is invoked to handle an attention interruption in another program, as defined by an AETD in that program, without causing the first AETD to be ignored. See Appendix I for a discussion of this facility.

If the save area or entry point is externally defined, it must be used as an argument of an EXTRN statement in the user's program. If the entry point is not externally defined, it must be used as the argument of an ENTRY statement.

A-type and R-type address constants are normally generated for each entry point name; in this case, the R-value is the origin of the first declared PSECT in the assembly module containing AETD. If an entry point is externally defined, AETD generates a pair of V-type and R-type address constants for that entry point operand. An A-type address constant is also normally generated for each save area address.

If a user is using AETD to handle attentions, the attention handling routine must include a TCNTRL TYPE=RESTART (see Appendix N) macro, or the user's default for the implicit operand ATTNMODE must be OLD; if not, then a terminal lockout will occur (see Appendix I).

<u>Blank AET Entries</u> - Blank entries may be placed in the AET by skipping an AETD operand; that is, by entering three commas (,,,) or by the system when fewer than five AET entries are provided by the user. Thus, if a user codes three attention handling routines but codes them as the first, second, and fourth operands of the AETD macro instruction, pressing the ATTN key three times will pass control to the command system. If he codes them as consecutive operands, pressing the ATTN key four times will pass control to the command system.

<u>Example</u>: In the following example, the user has provided two attention-handling routines having the entry points EPMODA and EPMODB respectively. If the user presses the attention key at the terminal once following execution of the first AETD macro instruction, control will be passed to the user-coded routine at EPMODA. When the user presses the attention key a second time before the routine at EPMODA has completed execution, control will be immediately passed to the routine at EPMODB. Execution of a second AETD macro instruction, having no operand, will return control of attention interruptions to the appropriate system routines.

```
                 .
                 .
             AETD  (EPMODA,SAVA,EPMODB,SAVB)
                 .
                 .
             AETD
                 .
```

<u>ARM -- Initialize an Explicit Adcon Group (O)</u>

The ARM macro instruction initializes (arms) an explicit adcon group (see the description of the ADCON macro instruction), so that it may be used by a load type-E CALL macro instruction or a LOAD macro instruction.

Explicit adcon groups must be initialized if:

1. They have already been used to refer to one program and the same adcon group is to be used to refer to a different program.

2. The adcon group has been used at least once and the associated program has been deleted by the DELETE macro instruction.

3. They were generated by an ADCON macro instruction without the EP operand.

Name	Operation	Operand
[symbol]	ARM	adcon group address,external name address

adcon group address
    specifies the address of adcon group to be initialized.

Specified as: An RX address, or register notation (2 through 12).

external name address
specifies the address of an eight-byte field that contains the external name that is to be placed in the explicit adcon group: the name of the module, entry point, or control section to be loaded or called.

Specified as: An RX address, or register notation (2 through 12).

Return Data: After execution of the ARM macro instruction, register 15 contains the address of the armed adcon group.

AWAIT -- Wait for an Interruption (R)

The AWAIT macro instruction enables you to check for the completion of an event and to enter your task into the delay state to await completion.

Name	Operation	Operand
[symbol]	AWAIT	

Note: There are no operands.

Execution: The AWAIT routine checks whether the SVC (1) was the subject of an execute (ILC=2) and (2) is positioned on the second halfword of a fullword (implying an event control block). If both of these conditions exist, the event control block complete bit (bit 1 of the first byte) is checked. If this bit is on (or if any enabled interruptions are pending on the task's TSI), the event is complete, no waiting is required, and control is returned to the issuing program. If this bit is off, a wait is required; the task is put into the delay state.

Programming note: AWAIT resets ISALCK (if previously set) and allows any interruptions enabled in the VPSW.

Example: Suppose you want to place your task in the delay state (inactive TSI list) until an I/O operation is completed. You might write:

```
WAIT    EX      0,ECB+2
        B       SOMEPLACE
ECB     DS      0F
        DC      H'0'            SECOND BIT IS COMPLETE BIT
        AWAIT                   AWAIT MUST BE SUBJECT OF EXECUTE
```

BPKDS -- BUILTIN Procedure Keyword Dictionary (O)

The BPKDS macro instruction, in conjunction with the BUILTIN facility of the command system, provides for specification of prototype command parameter lists that may include simple keywords, self-defining keywords, repeating keywords, list keywords and repeating list keywords (which may be unnamed).

Standard form:

Name	Operation	Operand
extname	BPKDS	entry [,parameter,...][,MF=I]

L-form:

Name	Operation	Operand
label	BPKDS	[,parameter,...],MF=L

E-form:

Name	Operation	Operand
extname	BPKDS	entry,MF=(E,label,count)

extname
    specifies the symbolic name of the keyword dictionary, to be used as the external name (EXTNAME) operand of the BUILTIN command.

    Specified as: a symbol, one to eight alphameric characters, the first of which must be alphabetic.

entry
    specifies the symbolic name of the starting point of the routine that is to execute the command.

    Specified as: a symbol, one to eight alphameric characters, the first of which must be alphabetic.

    Note: entry may be external to the module in which the BPKDS occurs; if entry is in the same module as BPKDS, it need not be an ENTRY point.

label
    specifies the symbolic name of the L-form BPKDS to be used with the E-form.

    Specified as: a symbol, one to eight alphameric characters, the first of which must be alphabetic.

count
    specifies the number of first-level keywords in the L-form used by the E-form.

    Specified as: a decimal integer, 1 to 255 inclusive.

parameter
    may be:

keyword	simple keyword
*keyword	self-defining keyword
keyword..	repeating keyword
(keyword,sub-parameter[,...])	list keyword
(keyword..,sub-parameter[,...])	repeating list keyword
(..,sub-parameter[,...])	unnamed repeating list

    Note: in the above expressions, sub-parameter may take any of the forms allowed for parameter; keyword is a symbol to be recognized as a keyword in the user's command operand list.

    Specified as: one to eight alphameric characters, the first of which must be alphabetic.

<u>Note</u>: The I-form and E-form must be in a private read-write control section. The L-form should be in a read-only control section. Any number of E-forms may reference one L-form.

A simple keyword is used in the command as "symbol=string" and is resolved as a parameter list entry which points to the string.

A self-defining keyword is used as "symbol", "NOsymbol", or "symbol=string", and is resolved as a parameter list entry which points to a character string. Specification of "symbol" points to a string 'Y'; "NOsymbol" points to string 'N'; and "symbol=string", equivalent to a simple keyword, points to the string.

A repeating keyword is used as "keyword=string1 [string2] [,...]" and is resolved as a parameter list entry which points to a parameter list. The entries of the second list point to the strings.

A list keyword is used as "keyword=(sub-parameter1 [,sub-parameter2 [,...])" or as "keyword=string", and is resolved as a parameter list entry which points to a second parameter list in which element is one of the sub-parameters. The second list points to character strings and-or further parameter lists, as required by the BPKDS specification of the sub-parameters. Using "keyword=string" is equivalent to using "keyword=(sub-parameter1=string)".

<u>Macro expansion</u>: BPKDS I-form generates a chained keyword dictionary, containing no alterable data. The physical structure of the dictionary is unpredictable. The logical structure is a relatively simple map of the prototype parameter list. The dictionary may be used to drive specialized parameter-retrieval routines, such as retrieval-by-keyword.

The E-form (one is generated by the I-form) generates the anchor or root, of the dictionary and also provides command entry point information for lthe BUILTIN processor. It contains alterable data.

The current level of DSECT CHABPK properly describes the entire structure.

<u>Command processing</u>: The CZATE command analysis routine, which is invoked to process a command, provides complete analysis of BPKDS-defined command parameter lists. The analysis function includes full DEFAULT searching in addition to construction of chained, ordered, and counted parameter lists mapped by the prototype parameter list.

A special entry point to CZATE may be called by privileged programs to perform BPKDS-controlled analysis of internally generated parameter lists. A macro or direct call entry to a command routine can have the same parameter list capability as a command call entry by using the same parameter retrieval code.

<u>Parameter analysis</u>: Upon entry to the command routine specified by BPKDS, general register (1) points to the first level parameter list which is normally built in the area provided by the E-form or the macro. The entered program must regard the entire parameter structure as being read-only. The logical organization of the analyzed parameter list corresponds to the BPKDS prototype list; the physical structure is unpredictable.

<u>Macro call processing</u>: The following call makes the full parameter string analysis facility available to a command routine called directly by another program:

```
    CALL CZATE6,(string,bpkds,option)
```

where: string is the address of the command parameter string to be analyzed (must be terminated by the character X'27'); bpkds is the address of the E-form or I-form BPKDS macro to be used; and option is the address of a one-byte switch that must contain X'01' if default and synonym searching is to be done: otherwise, it must contain X'00'.

Return codes: upon return, the low-order byte of register 15 will contain one of the following return codes:

Code	Meaning
X'00'	successful
X'04'	bpkds (the address) did not point to a valid BPKDS
X'08'	string was longer than 256 bytes
X'10'	option was not X'01' or X'00'

The parameter tree that would normally be pointed to by register 1 will be found at location BPKSPAR of DSECT CHABP~~AK~~.

BSP -- Backspace a Block (R)

The BSP macro instruction (for BSAM) backspaces a block on the current magnetic tape or direct access volume. Backspacing is always toward the load point (or beginning-of-file on direct access) regardless of the OPEN macro instruction's parameters or the direction of reading.

This macro instruction is applicable only to magnetic tape or a direct access device and becomes a NOP for other devices.

Name	Operation	Operand
[symbol]	BSP	dcb address

dcb address
    specifies the address of the data control block opened for the data set to be backspaced.

    Specified as: An RX address, or register notation (2 through 12).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Abnormal termination occurs if:

1. The data control block specified by the user is not validly opened.

2. The track overflow option is specified.

3. All read and write operations have not been checked for completion.

CAUTION: Two BSP macro instructions should not be issued on a direct-access data set without an intervening I/O operation (such as read or write). If two consecutive backspace operations are attempted, the second BSP macro instruction will not cause a backspace.

Return Data: Following execution of the BSP macro instruction, register 15 contains a return code of X'00' if the operation is completed normally. It also contains a return code of X'00' if the operation encountered a permanent positioning error, in which case the next CHECK of a READ or WRITE passes control to the SYNAD routine.

If two BSP macro instructions are issued without at least one intervening I/O operation, a return code of X'04' is placed in register 15, indicating that the second BSP macro instruction was not executed.

If a tape mark is encountered on a backspace, the tape is repositioned to its position before BSP was issued; a return code of X'04' is placed in register 15.

If the user attempts to backspace into a header or trailer label track on a direct access volume, backspacing does not occur and a return code of X'04' is placed in register 15.

<u>Programming Notes</u>: All read or write operations must be checked for completion before the BSP macro instruction is executed.

Rather than issue more than one BSP without an intervening READ or WRITE, NOTE, POINT, or CNTRL macro instructions should be used.

CALL -- Call a Module (S)

The CALL macro instruction passes control from one module to another module or from one point in a module to another point within the same module.

The module issuing the CALL macro instruction is referred to as the calling module; the module receiving control is referred to as the called module.

Standard form:

Name	Operation	Operand
[symbol]	CALL	entry point name,[(parameter address,...)],[VL] [,{E|I}][,ID=identifier]

L-form:

Name	Operation	Operand
symbol	CALL	,[(parameter address),...],[VL],MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	CALL	entry point name,[(parameter address,...)], [VL][,{E|I}][,ID=identifier],MF=(E,list)

entry point name
   specifies the symbolic name of an entry point to which control is to be passed.

   Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic); or register notation (15 only). If the module is not reenterable, the symbol can be:

   - the name of a control section,
   - the name in the operand field of an assembler language ENTRY statement, or
   - a module name.

   If the module is reenterable, control section name must not be used. If register 15 is specified, and the load type is I, the address of an implicit adcon group must be loaded into register 15 before execution of this macro instruction. If register 15 is specified and the load type is E, the address of an explicit adcon group must be loaded into register 15 before execution of this macro instruction (see the Programming Notes).

parameter address
   specifies the address of a parameter to be passed to the called program. The parameters must be written as a sublist enclosed in parentheses. If one or more parameter address operands are written, a parameter list is generated; it consists of a fullword for each operand. Each fullword is aligned on a fullword boundary and contains the address to be passed. The addresses appear in the parameter list in the same order as in the macro instruction.

   When the called program is entered, register 1 contains the address of the parameter list. If the E-form macro instruction is used, the parameter addresses overlay the corresponding L-form parameter addresses.

   | Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also in register notation (2 through 12); in the E-form only, also as an RX address.

VL
   specifies that the first word preceding the parameter list contains a binary number equal to the number of parameters (including null parameters) supplied by the parameter address operand.

   The operand parameter list is fixed-length if it contains a known number of parameters every time the called program is given control. The list is variable length if it contains a varying number of parameters. In the latter case, the VL operand must be included so that the called program will take action to determine the length of the parameter list being passed.

If VL is specified on the E-form, it must have been specified on the L-form; if VL is not specified on the E-form, it must not have been specified on the L-form.

Specified as: VL

{E|I}
specifies whether the call is explicit or implicit.

Specified as:
E - an explicit call is requested.
I - an implicit call is requested.

Default: I

ID=
specifies a binary calling sequence identifier for the CALL macro instruction. This parameter may be used to identify the CALL macro instruction uniquely. This parameter generates a NOP (see the Programming Notes below).

Specified as: An absolute expression, maximum value 4095.

Programming Notes: The explicit CALL macro instruction causes the named module to be loaded (if necessary) during execution; it may then be deleted through use of the DELETE macro instruction (refer to the DELETE macro instruction in this section). If an implicit CALL macro instruction is issued, the called object module is already in virtual storage and may not be deleted by the calling object module through use of the DELETE macro instruction.

An implicit adcon group consists of two contiguous fullwords: the V-type and R-type address constants of the entry point. These address constants must be coded as a V-type followed by an R-type. See Example 2 below.

An explicit adcon group may be generated through the ADCON macro instruction. The ARM macro instruction can be used to reinitialize the adcon group.

```
L       ADCON     CALL,EP=entry point name
```

Refer to the ADCON and ARM macro instructions in this section.

If (15) is written for the entry point name operand of an explicit CALL macro instruction, the explicit adcon group should be armed if necessary and then reused for any subsequent calls to the desired program. ADCON is capable of generating an armed adcon group(refer to the ARM macro instruction in this section). However, the explicit adcon group is altered by the execution of the first CALL macro instruction and cannot be reused if the module has been deleted (refer to the DELETE macro instruction). If an object module has not been loaded or has been loaded and then deleted and it is desired to call it using a previously used explicit adcon group, it is necessary to issue or reissue the ARM macro instruction. The ARM macro instruction adjusts the explicit adcon group so that it may be reused. Refer to Examples 4 and 5, the ADCON macro instruction, and the ARM macro instruction.

If the entry point name operand specifies an internal symbol, it must appear as the operand of an assembler language ENTRY statement. The reason for this rule is that the called name must be in the program module dictionary (PMD) if the CALL macro instruction is to execute properly.

Upon entry to the called program, the ID value can be determined by examining the location whose address is contained in register 14; the address is that of a fullword, the low-order two bytes of which contain the ID. When CALL is specified as:

```
CALL (15),(3),,,ID=16
```

the expansion contains the following code:

```
BASR     14,15      LINK
DC       X'4700'    A NOP
DC       AL2(16)    ID INTO OPERAND FIELD OF NOP
```

Return Data: Register 14 contains a valid return address when control is passed to the called module. Therefore, by issuing a RETURN macro instruction or branching to the address in register 14, control is transferred to the instruction after the CALL macro instruction in the calling module. The CALL macro instruction is advantageous because it eliminates the need for writing linkage to the called module.

L- and E-Form Use: E-form parameter list entries overlay the corresponding L-form parameter list entries.

This example shows L- and E-form use:

```
ALPHA    CALL  ,(A,,C),MF=L
BETA     CALL  RTNA,(,B,),ID=36,MF=(E,ALPHA)
```

Examples: The following are typical examples of implicit and explicit use of CALL.

EXAMPLE 1 - Implicit CALL:

```
EX1      CALL  ENT
```

When the CALL macro instruction in the calling program is executed, control is passed to ENT.

EXAMPLE 2 - Implicit adcon group for an implicit CALL:

```
EX2      CALL  (15),(ABC,DEF),VL
```

Calling program contains an implicit adcon group:

```
SAMNAM   ADCON IMPLICIT,EP=CLDRTN
```

Before the CALL macro instruction is executed, register 15 must be loaded with the address of the adcon group; for example, LA 15,SAMNAM.

When the called program is entered, register 1 points to a two-word parameter list. The first word contains the address of ABC; the second word contains the address of DEF. The word preceding the parameter list contains a 2, indicating that two words containing the addresses of parameters follow.

EXAMPLE 3 - Explicit CALL:

```
     EX3      CALL   ATOL,(BAT,CAT),,E
```

At execution time, the program whose entry point name is ATOL is loaded into virtual storage (if necessary) and control is transferred to ATOL. When the called program is entered, register 1 points to a two-word parameter list that contains the addresses of BAT and CAT. Register 14 contains the return address.

EXAMPLE 4 - Repetitive explicit CALLs, reusing an explicit adcon group:

```
                .
                .
                .
               ARM      MAX,JOE
              CALL      (15),,,E
                .
                .
                .
              CALL      (15),,,E
                .
                .
                .
              CALL      (15),,,E
                .
                .
                .
     MAX      ADCON     CALL
     JOE       DC       CL8'CALLEE'
```

EXAMPLE 5 - Repetitive explicit CALLs with an intervening DELETE, reusing an explicit adcon group.

```
                .
                .
              ARM       MAX,JOE
              CALL      (15),,,E
                .
                .
                .
              DELETE    EP=CALLEE
                .
                .
                .
              ARM       MAX,JOE
              CALL      (15),,,E
                .
                .
                .
     MAX      ADCON     CALL
     JOE       DC       CL8'CALLEE'
```

CAT -- Create or Change Catalog Entry (S)

The CAT macro instruction creates a catalog index for a generation data group, or renames a data set. For physical sequential data sets, CAT creates or alters a catalog entry.

The CAT macro instruction can be coded with either of two sets of operands, depending on the objective. To rename a VAM or physical sequential data set, to change a version number of a generation data group, or to create or alter a catalog entry for a physical sequential data set, use:

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	CAT	{address of operand string 'data set name 1,{N|U},[{R|U}][,data set name 2]'}

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	CAT	'data set name 1,{N|U},[{R|U}][,data set name 2]' ,MF=L

Note: A symbol is required in the name field.

E-form:

Name	Operation	Operand
[symbol]	CAT	MF=(E,list)

To create a generation data group for VAM or physical sequential data sets, use:

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	CAT	{address of operand string 'GDG=gdg name,number,[{A|O}][,ERASE={Y|N}]'}

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	CAT	'GDG=gdg name,number,[{A|O}][,ERASE={Y|N}]',MF=L

Note: A symbol is required in the name field.

E-form:

Name	Operation	Operand
[symbol]	CAT	MF=(E,list)

address of operand string
    specifies the address of the first operand in the operand string.

    Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified as a character string enclosed in apostrophes, as shown.

data set name 1

specifies the name of an uncataloged SAM data set defined in a DDEF macro instruction or command, or specifies any cataloged VAM or SAM data set name. The name may be that of a VAM data set only if a generation index is to be created or the name cf the data set is to be changed. The data set must reside on a direct access or magnetic tape volume.

Specified as:

- The fully qualified name of a partitioned or nonpartitioned data set or a partitioned or nonpartitioned generation data group (identified by absolute generation name or relative generaticn number).
- The partially qualified name of any data set other than a generation data group.

{N|U}

specifies the updating of an existing VAM or SAM catalog entry (U), or the creation of a new SAM catalog entry (N).

Specified as:
N (SAM only) or U

{R|U} (N/A for VAM)

specifies the owner access qualification fcr SAM data sets:

Specified as:
R - read-only access
U - unlimited access

If R is specified, the data set owner may erase but not write into his data set.

Default: U
This default is valid only if a new catalog entry is being made; otherwise, no change is made to the access qualification.

data set name 2

specifies the new name for the data set. This operand is necessary only if the currently defined name of the data set is to be changed. The data set name may have a relative generation number appended.

Specified as:

- The fully qualified name of a partitioned or nonpartitioned data set or a partitioned or nonpartitioned generation data group (identified by absolute generation name or relative generaticn number).
- The partially qualified name of any data set other than a generation data group.

GDG=gdg name

specifies the name of a new generation data grcup.

Specified as: GDG=gdg name, where gdg name is a data set name as defined in Part II, Section 1.

number

specifies the number of generations to be maintained in the generation data group.

Specified as: An absolute expression.

{A|O}

specifies the action to be taken when the next generation (beyond the number specified in the previous operand) is being cataloged in the generation data group.

Specified as:
A - all previous generations are to be removed from the catalog.
O - only the oldest generation is to be removed.

Default: O

ERASE=

specifies the disposition of old generations deleted from the catalog. This applies to private volumes only; data sets on public volumes are always erased when uncataloged.

Specified as:
Y - erase external storage belonging to old generation data group members.
N - save old generation data group members.

Default: N

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVIIEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The system automatically recatalogs multivolume data sets that expand or contract.

When data set name 1 is given, a new entry is made in the catalog if the N option was specified. When the U option is given, the catalog entry is updated with the requested changes to the data set name (VAM and SAM data sets) and/or access qualifier (SAM only). In addition, when data set name 2 is supplied, a change is made to the name in the data set labels (DSCBs) on the volumes containing the data set. This step is omitted if the volumes are on tape.

If the GDG keyword is specified, an index is created for a new generation data group using the parameters supplied. If the generation data group is already cataloged, no updating is possible.

If the data set name is specified with a member name, only the data set name itself is used; the member name is removed.

If the user wants to change the definition information for a cataloged SAM data set, he may do so merely by issuing a CAT macro instruction with "update" indicated (U).

For private data sets only, the owner of a generation data group is allowed to catalog generations of that group. Sharers, regardless of their level of access, are not permitted to do this.

Generations of a generation data group that reside on private storage can be saved by the user even after they are uncataloged.

Return Data: At completion of execution of the CAT macro instruction, the low-order byte in register 15 contains one of the following codes:

Code (Hexadecimal)	Significance
00	Cataloging accomplished as requested
04	Name cannot be changed since new data set name not unique, no cataloging
08	Invalid element in input string
0C	No cataloging for other reasons
10	Data set name not unique, already in catalog
14	No volume of data set mounted; cannot catalog
20	VAM data set not GDG or rename option
24	Open DCB

Examples: In EX1, the operands are presented as a character string. In EX2, an address designates the location of the operands.

```
     EX1     CAT       'DATASET,U,U'
     EX2     CAT       OPLISTC
```

CDD -- Retrieve and Execute DDEF Commands (S)

The CDD macro instruction retrieves one or more DDEF commands from a line data set containing prestored DDEF commands (line data sets are discussed in Command System User's Guide). The macro instruction processes the retrieved commands as though they had just been entered by the user. The user can thus create a line data set of commonly used DDEF commands for reference through the CDD macro instruction, eliminating the need for direct DDEF macro instruction or command entries for each run of a program.

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	CDD	{address of operand string {'data set name[,DDNAME=data definition name,...]'}

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	CDD	'data set name[,DDNAME=name,...]',MF=L

Note: A symbol is required in the name field.

E-form:

name	Operation	Operand
[symbol]	CDD	MF=(E,list)

address of operand string
  specifies the address of the first operand in the operand string (see "Operand String" in Part II, Section 1).

  Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified in the operand as a character string enclosed in apostrophes, as shown.

data set name
specifies the name of the line data set containing the prestored DDEF commands. See "Data Set Name" in Part II, Section 1.

Specified as: The fully qualified name of a nonpartitioned data set, or of a nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number).

DDNAME
specifies the name of a particular DDEF command in the data set.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

Default: All DDEF commands in the data set are retrieved and executed.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: The user must make sure that none of the DDEF commands or macro instructions for his task has the same name as a DDEF command retrieved through this macro instruction.

Return Data: At completion of execution of the CDD macro instruction, the low-order byte of register 15 contains one of the following hexadecimal codes:

Code	Significance
00	Successful completion
04	Invalid data set name
08	Invalid data definition name
0C	Data definition name not in data set
10	Error return from DDEF routine
14	Not a line data set

CHDERMAC -- Generate Error Message (O)

This inner macro instruction is used to generate error messages pertaining to errors encountered in macro expansions.

Name	Operation	Operand
[symbol]	CHDERMAC	message number,[operand name], [outer operand 1],[outer operand 2], [outer operand 3][,S=severity code]

message number
specifies the message to be generated.

Specified as: A number. Message numbers and the messages they identify are shown in Table 1.

operand name (opnm in Table 1)
specifies the name of an outer macro instruction operand, or other information defined by the programmer.

<u>Specified as</u>: A character string (a character string contains no embedded blanks or commas and is not enclosed in apostrophes).

outer operand 1, 2, and 3 (opva, opvb, and opvc in Table 1)
specify operands of the outer macro instruction. A maximum of three operands can be specified in any one error message. These operands may also be used for other purposes, which the programmer defines.

<u>Specified as</u>: A character string (a character string contains no embedded blanks or commas and is not enclosed in apostrophes).

S= (sc in Table 1)
specifies the severity code associated with the error.

<u>Specified as</u>: 0, 1, or 2. The meanings of these severity codes are as follows:

0 - message is not included in the error count.

1 - the error count is incremented by one, and a W appears on the message.

2 - the error count is incremented by one, and an F appears on the message.

If specified greater than 2, it is automatically set to 2.

<u>Default</u>: A severity code associated with the specified message number is used (shown in Table 1).

Message number	sc	mmm	Message text
1	2	004	REQUIRED OPERAND(S) NOT SPECIFIED
2	2	001	FIRST OPERAND REQ'D-NOT SPECIFIED
3	2	001	SECOND OPERAND REQ'D-NOT SPECIFIED
4	2	001	THIRD OPERAND REQ'D-NOT SPECIFIED
5	2	001	FOURTH OPERAND REQ'D-NOT SPECIFIED
6	2	001	DCB OPERAND REQ'D-NOT SPECIFIED
7	2	001	DECB OPERAND REQ'D-NOT SPECIFIED
8	2	001	KEY OPERAND REQ'D-NOT SPECIFIED
9	2	001	FIFTH OPERAND REQ'D-NOT SPECIFIED
10	2	001	LOW. LIM. OPERAND REQ'D-NOT SPECIFIED
13	2	001	AREA OPERAND REQ'D-NOT SPECIFIED
14	2	001	LENGTH OPERAND REQ'D-NOT SPECIFIED
15	2	001	VALUE OPERAND REQ'D-NOT SPECIFIED
17	2	001	MODE OPERAND REQ'D-NOT SPECIFIED
18	2	001	REGISTER OPERAND REQ'D-NOT SPECIFIED
19	2	001	MESSAGE OPERAND REQ'D-NOT SPECIFIED
21	2	001	NAME OF DCB REQ'D-NOT SPECIFIED
22	2	001	NAME OF ADCON REQ'D-NOT SPECIFIED
23	2	001	NAME OF CSECT REQ'D-NOT SPECIFIED
24	2	001	NAME OF L FORM REQ'D-NOT SPECIFIED
25	2	001	TYPE OF OPERAND REQ'D-NOT SPECIFIED
28	2	001	CODE OPERANDAND REQ'D-NOT SPECIFIED
31	2	001	EP OR EPLOC OPERAND REQ'D-NOT SPECIFIED
35	2	002	INVALID MF OPERAND SPECIFIED-opva
36	2	002	INVALID FIRST OPERAND SPECIFIED-opva
37	2	002	INVALID SECOND OPERAND SPECIFIED-opva
38	2	002	INVALID THIRD OPERAND SPECIFIED-opva
39	2	002	INVALID FOURTH OPERAND SPECIFIED-opva
40	2	002	INVALID FIFTH OPERAND SPECIFIED-opva
42	2	002	INVALID EP OR EPLOC OPERAND SPECIFIED-opva
44	2	002	INVALID LENGTH OPERAND SPECIFIED-opva
45	2	002	INVALID MODE OPERAND SPECIFIED-opva
46	2	002	INVALID REG(S) OPERAND SPECIFIED-opva
47	2	002	INVALID AREA OPERAND SPECIFIED-opva
48	2	002	INVALID TYPE OPERAND SPECIFIED-opva
49	2	002	INVALID OPTION OPERAND SPECIFIED-opva
50	2	002	INVALID OPTION 1 OPERAND SPECIFIED-opva
51	2	002	INVALID OPTION 2 OPERAND SPECIFIED-opva
54	2	002	INVALID KEYWORD OPERAND SPECIFIED-opva
55	2	002	INVALID REGISTER NOTATION SPECIFIED-opva
56	1	025	PACK OPERAND NOT ALLOWED W/MODE=R
57	2	002	INVALID PR OPERAND SPECIFIED-opva
58	2	002	INVALID PACK OPERAND SPECIFIED-opva
59	2	002	LV OPERAND REQ'D-NOT SPECIFIED
62	1	067	ADCOND MACRO PREVIOUSLY SPECIFIED
63	2	002	INVALID TAM CHARACTER CODE OPERAND SPECIFIED-opva
69	2	006	REGISTER NOTATION INVALID W/MF=L
73	0	024	CSECT NAME BLANK. MACRO NAME OMITTED.
85	1	013	MESSAGE OPERAND NOT ALLOWED W/MF=E
86	1	013	OPLIST OPERAND NOT ALLOWED W/MF=E
87	2	014	DECB NOT SPECIFIED AS SYMBOL
88	1	015	MORE THAN ONE OF EP OR EPLOC PRESENT
89	0	050	opnm OPERAND INCONSISTENT WITH TYPE=opvaopvb
90	2	050	opnm INCONSISTENT W/opva OPERAND
147	0	050	opnm OPERAND INCONSISTENT-IGNORED
157	1	051	INVALID CODE FOR opnm-IGNORED-opva
159	1	053	INVALID CODE FOR DSORG-IGNORED-opva
162	1	056	MACRF INVALID WITH SPECIFIED DSORG-IGNORED-opva
163	1	056	EXLST INVALID WITH SPECIFIED DSORG-IGNORED-opva

Table 1. Error messages issued by CHDERMAC (part 1 of 2)

Message number	sc	mmm	Message text
166	1	060	INVALID CODE FOR DEVD WITH SPECIFIED DSORG-IGNORED-opva
167	1	065	MACRF INVALID-IGNORED-opva
169	1	067	DCBD MACRO PREVIOUSLY USED
173	1	062	DDNAME LONG-TRUNCATED TO 8 CHAR
174	1	070	devd=opvb IGNORES opnm=opva
175	1	071	INVALID opnm OPERAND SPECIFIED-IGNORED-opva
176	1	072	MULTIPLE DEVICE-DEP. PARAM. 1 SPECIFIED-IGNORED-opva=opvb
177	1	073	MULTIPLE DEVICE-DEP. PARAM. 2 SPECIFIED-IGNORED TRTCH=opva
178	0	074	PAD OPERAND GT 50-SPECIFIED VALUE USED-opva
179	0	101	CSECT ORIGIN USED FOR opnm RCON
180	1	003	opnm OPERAND INVALID OR NOT SPECIFIED-SET TO opva
181	1	076	BPY CNTR INDICATES WRAP AROUND TO TOP OF CRT
182	1	077	BLC GREATER THAN OR EQUAL TO BLIM
183	1	002	opnm INVALID-SET TO opva
184	*	078	* CURRENT BUFFER opnm=opva
185	*	079	* CURRENT BEAM POSITION COUNTER IS X=opnm, Y=opva
186	1	080	opnm COUNTER EXCEEDS CRT LIMITS
187	1	081	LOAD VARIABLE SPACE ORDER MAY NOT HAVE BEEN SPECIFIED PRIOR TO ENTERING STROKE MODE
188	2	103	opnm MACRO NOT ALLOWED FOR PRIVILEGED USER
189	2	103	opnm MACR NOT ALLOWED FOR NONPRIVILEGED USER (SETTDE)
200	1	101	ZERO USED FOR opnm RCON
201	1	075	VAR OPERAND NOT ALLOWED W/MODE=R
210	2	001	opnm OPERAND REQ'D-NOT SPECIFIED
211	2	002	INVALID opnm OPERAND SPECIFIED-opva
212	2	001	NAME OF BPKD REQ'D-NOT SPECIFIED
213	2	001	TOO MANY OPERANDS SPECIFIED

Key to abbreviations:

opnm = operand name	opva = outer operand 1
sc = severity code	opvb = outer operand 2
mmm = error message number	opvc = outer operand 3

Table 2. Error messages issued by CHDERMAC (part 2 of 2)

Execution: For any specified message number, an MNOTE instruction is generated to produce an error message of this form:

```
        nnnnnn (B*sc+S)***CHDmmm text
```

where:

(B*sc+S)	is the severity code.
B	is set equal to zero if the S= operand is present or to 1 if it is not present.
sc	is the defaulted severity code shown in Table 1.
S	is the severity code operand; if it is not present, zero is used.
nnnnnn	is the six-digit line number of the macro instruction for which the MNOTE is generated.
mmm	is the error message number shown in Table 1.

In general, you should attempt to continue processing a macro expansion after detecting an error and generating a message. However, although it is difficult to generalize, some errors should cause termination of processing. An example is an invalid MF operand in an S-type macro instruction, which makes further processing impossible. Another instance is the occurrence of an error that propagates other errors.

CHDPSECT -- Reserve Storage for Parameter List (O)

The CHDPSECT inner macro instruction establishes the next available location in the user's PSECT as the location at which the parameter list will be located. If no PSECT exists when the macro instruction is being assembled, the next available location in the current control section is used, and CHDPSECT generates a branch around the list.

Name	Operation	Operand
[symbol]	CHDPSECT	[location],[alignment][,string]

symbol
: is the symbolic location of the first byte to be assigned to the parameter list.

Specified as: One to eight alphameric characters, the first of which must be alphabetic.

location
: specifies the location to which the branch instruction is to transfer control.

Specified as: A relocatable expression or register notation.

Default: If this operand is omitted, CHDPSECT establishes its own branch address based on the length of the string specified in the string operand. Symbols generated by CHDPSECT will be of the form CHDx&SYSNDX where x identifies a unique symbol.

Note: If the location operand is omitted, the string operand must be present.

alignment
: specifies the alignment for the beginning of the parameter list.

Specified as:

0F - alignment on a fullword.
0H - alignment on a halfword.

Default: No special alignment is performed.

string
: specifies a character string, originally specified as an operand in the outer macro instruction, which is to be placed, as is, in the parameter list.

Specified as: The character string itself. (CHDPSECT generates and stores X'27' to indicate the end of the string.)

Note: When this operand is not specified, the branch address must be specified in the location operand.

Programming Notes: The use of a CNOP to force alignment is generally ineffective, since the parameter list may be generated in another control section (the PSECT). Placing the CNOP instruction before CHDPSECT has no effect other than to align the macro instruction.

CHDVAL -- Determine Type Code

CHDVAL tests a parameter 'P' and sets the GLOBAL SETC symbol CHDTYP to the type code of 'P', and returns in a register the value of 'P', or the address of 'P'.

Name	Operation	Operand
[symbol]	CHDVAL	P[,R]

P

the symbolic parameter whose type is to be tested by CHDVAL

Specified as: any alphameric string (beginning with an alpha character), or Register 0 through 15.

R

the register that is to contain the value associated with P.

Specified as: Register 0 - 15.

Initialization: The macro using CHDVAL must have defined CHDTYP as a global SETC symbol.

Return Data: The GLOBAL SETC symbol CHDTYP will, on return, be assigned one of the following values:

Code	Value of T'P	Meaning
C	B,X,C	P is a character, hexadecimal, or binary type symbol. R contains the address of P
U	U, N	P is undefined or a self-defining term. R contains the address of P.
R	register notation used	Register notation. R is not changed.
H	H, Y	P is defined as one of the halfword aligned types. R contains the contents of the field defined by P.
F	A, F, V, R, Q, D	P is defined as one of the fullword aligned types. R contains the contents of the field defined by P.
null	none of the above	Type code was not one of the ones checked and expected.

Examples: Assuming user has coded GBLC & CHDTYP:

(1) symbolic parameter

```
    CHDVAL WORD,15
  +    ST 15,4(,1)            On return CHDTYP='F' and
          .                   Register 15 will contain
          .                   an '8'.
          .
       WORD DC F'8'
```

(2) Register notation for P

```
     CHDVAL (5),15
  +    AIF ('CHDTYP'EQ'R')    On return CHDTYP='R'
```

```
|                .               and Register 15 has
|                .               not been changed.
                 .
|        ST 5,4(,1)
```

CHECK -- Wait for and Test Completion of an I/O Operation (R)

The CHECK macro instruction (for BSAM and IOREQ) waits for completion of an I/O operation requested by a READ or WRITE macro instruction for BSAM or by an IOREQ macro instruction for IOREQ, and detects any errors and exceptional conditions that may occur. If the I/O operation is completed successfully, the program resumes execution at the instruction after the CHECK macro instruction.

The user must issue a CHECK to test the I/O operation associated with a data event control block (DECB) before modifying or reusing it.

Name	Operation	Operand
[symbol]	CHECK	decb name

decb name
specifies the address of the data event control block (DECB) created as part of the expansion of a READ or WRITE macro instruction or furnished in the IOREQ macro instruction that is being checked.

| <u>Specified as</u>: Register notation (1 through 12), or an RX address.

<u>Initialization</u>: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

<u>Programming Notes</u>: The CHECK macro instruction must be used to test for completion of every READ, WRITE, or IOREQ. For each data set, the CHECK macro instruction must be issued in the same order in which the READ, WRITE, or IOREQ was issued. A BSAM CHECK must be issued before the number of outstanding READ or WRITE macro instructions exceeds the DCBNCP count (specified in the DCB macro instruction) in the data control block for the data set.

<u>For BSAM</u>: As required, the CHECK macro instruction passes control to appropriate exits that are specified by the user in the data control block for error analysis (SYNAD) and end-of-data set (EODAD). The CHECK automatically initiates volume switching for input data sets. Additional space for output data sets is automatically obtained when current space is filled and another WRITE is issued.

If the CHECK macro instruction tests a DECB that has not been posted as complete, the user's task waits until the event is completed.

If the CHECK macro instruction tests a READ operation that attempted to gain access to a block after the last block of a data set had been read, control is passed to the end-of-data set exit (EODAD) whose
| address is provided in the EODAD field of the data control block.
| The task is abnormally terminated if an EODAD address is not sup-
| plied. Refer to Appendix C for contents of registers when the EODAD routine is entered.

If the CHECK macro instruction determines that the READ or WRITE operation was not completed correctly because of an I/O error, control is given to the user's synchronous error exit (SYNAD) routine. Refer to Appendix B.

The RETURN macro instruction may be used to return to the calling program from the SYNAD routine. The program may then proceed, if desired, as if an error had not occurred. For input, processing may be continued; for output, the data control block should be closed.

The task is terminated if an error is detected by the CHECK macro instruction and the user has not provided a SYNAD routine.

If the CHECK macro instruction detects an end-of-volume condition when processing a multivolume data set, processing continues with the next volume. If there are no additional volumes, the user's EODAD routine is entered.

A hardware-detected incorrect-length block is not interpreted as an error by the CHECK macro instruction if format-U records or truncated blocks of format-F records are being read. To determine the length of the block actually read, the user can examine the channel status word (part of the status indicators pointed to in the DECB) after issuing the CHECK macro instruction. The first byte of a format-U record read backwards from magnetic tape may be located by the same method.

Figure 3 lists the results of incorrect-length error in which the length of the record read is different from the DCBBLK for format-F and format-U, or the LL field for format-V.

Record Format (RECFM)	Control passed to SYNAD
Fixed (F)	Yes
Fixed blocked (FB)	If block is short by a nonmultiple of LRECL
Fixed standard (FS)	Yes
Fixed block standard (FBS)	If block is short by a multiple of LRECL *
Variable (V)	Yes
Variable blocked (VB)	Yes
Undefined (U)	No
*If the block is short by a multiple of the record length (LRECL), the next record causes an end-of-volume condition. If the current volume is the last of the data set, control is passed to EODAD. If the current volume is not the last, processing continues on the next volume.	

Figure 3. How incorrect record length is handled

For IOREQ: If an IOREQ results in a unit check or unit exception, the CHECK of the DECB associated with this IOREQ causes control to be given to the user's SYNAD routine specified in his data control block. If a linkage to SYNAD is executed by CHECK, all outstanding IOREQs are purged from the system. In the user-provided SYNAD routine, the user may reference the DCBEC field to facilitate reissuing any purged IOREQ. A RETURN may be issued in a SYNAD routine that causes control to be returned to the instruction following the CHECK that invoked the SYNAD routine.

Upon entry to the SYNAD routine, register 1 contains the address of the DECB associated with the IOREQ involved.

When a subsequent IOREQ is executed after the SYNAD routine is invoked, the contents of the area pointed to by DCBDEC in the data control block may be changed.

If the DCBDEVD field is zero or defaulted, any unit check or unit exception causes the CHECK of the appropriate DECB to invoke SYNAD.

Example: The CHECK macro instruction tests for completion of I/O operations in the order in which they are requested. The operand field contains the name of the data event control block specified in the read, write, or I/O request.

```
                .
                .
                .
      EX1       READ       INDECB,SF,INVEN,WORK,100
                .
                .
                .
                CHECK      INDECB
                .
                .
                .
      EX2       WRITE      OUTDECB,SF,MNTHRPRT,WORK,100
                .
                .
                .
                CHECK      OUTDECB
                .
                .
                .
      EX3       IOREQ      IODECB,N,DCBAD,VCCWAD,10,3
                .
                .
                .
                CHECK      IODECB
                .
                .
                .
```

CKCLS -- Check Protection Class (R)

The CKCLS macro instruction enables you to check the most restrictive protection class assigned to a group of halfpages.

Name	Operation	Operand
[symbol]	CKCLS	[starting address][,number of halfpages]

starting address
    specifies the virtual storage address of the first halfpage you want to check.

Specified as: An RX address, or register notation (1 through 12). If register notation is used, the address must be loaded into the specified register before issuing the macro instruction.

Default: It is assumed that the issuer has placed the starting address in register 1.

number of halfpages
specifies the number of consecutive halfpages you want to check.

Specified as: An absolute expression or register notation (0 and 2 through 12). The maximum number of halfpages that may be specified is 8192.

Default: It is assumed the issuer has placed the information in register 0.

Execution: Consecutive halfpages starting at the address contained in register 1 and equal to the halfpage count contained in register 0 are checked.

Return Data: A code indicating the most restrictive protection class of the pages checked is returned in the low-order byte of register 0. One of these codes is returned:

Code	Protection Class
0	Page unassigned
1	User read/write (least restrictive)
3	User read only
7	User cannot read or write (most restrictive)

Example: Suppose you want to check the protection class of the five halfpages beginning at RJG. You write:

```
CKCLS       RJG,5
```

CLATT -- Give System Control of Attention Interruptions (O)

The CLATT macro instruction allows the user to relinquish control of attention interruptions; the system then processes attention interruptions.

Name	Operation	Operand
[symbol]	CLATT	

Note: There are no operands.

Initialization: This macro instruction cannot be assembled in a privileged module unless the most recently issued DCLASS macro instruction in the assembly specified USER (see Appendix M) or if the DCLASS option is USER by default.

Programming Notes: This macro instruction is used in conjunction with the USATT macro instruction, discussed in this section.

CLIC -- Read Command From SYSIN (Conditional) (O)

For conversational tasks only, control is passed to the command system and the user is given an opportunity to enter a command at his SYSIN terminal.

Name	Operation	Operand
[symbol]	CLIC	

Note: There are no operands.

Initialization: A DCLASS macro instruction with the USER option must be coded (or defaulted) in a CSECT prior to coding CLIC. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding CLIC must be issued with the USER option.

Execution: the CLIC macro instruction switches the task from program mode to command mode to allow the user to enter commands. This macro instruction causes an unconditional pause in conversational mode and is disregarded in non-conversational mode. Any commands may be issued. The task can be switched back to running the program by issuing the RUN command.

Example: at any point in the program you want to pause, write:

```
            CLIC
```

CLIP -- Read Command From SYSIN (unconditional) (O)

For nonconversational or conversational tasks, control is passed to the command system and the next command is read from the user's SYSIN device.

Name	Operation	Operand
[symbol]	CLIP	

Note: There are no operands.

Initialization: A DCLASS macro instruction with the USER option must be coded (or defaulted) in a CSECT prior to coding CLIP. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding CLIP must be issued with the USER option.

Execution: The CLIP macro instruction switches the task from program mode to command mode to allow the user to enter commands. This macro instruction causes an unconditional pause, and executes whether the task is conversational or nonconversational. Any commands may be issued from the terminal during conversational program stoppage. If the stopped program is nonconversational, the SYSIN data set will be interrogated for the next commands. The task can be switched back to program mode by issuing a RUN command.

Example: the point in the program you want to pause, write:

```
            CLIP
```

Note: The CLIP macro instruction reads from the SYSIN data set and does not require a terminal; CLIC reads only from a terminal and must, therefore, only be used in a conversational task.

CLOSE -- Disconnect Data Set From User's Problem Program (S)

The CLOSE macro instruction disconnects one or more data sets from the user's problem program.

During the execution of CLOSE, the user's trailer label routine, if supplied, will be given control (BSAM and QSAM only). (Refer to Appendix A.)

Standard form:

Name	Operation	Operand
[symbol]	CLOSE	({dcb address,[{REREAD|LEAVE|RUN}]} ,...)[,TYPE=T]

L-form:

Name	Operation	Operand
[symbol]	CLOSE	({dcb address,[{REREAD|LEAVE|RUN}]} ,...)[,TYPE=T] ,MF=L

Note: A symbol is required in the name field.

E-form:

Name	Operation	Operand
[symbol]	CLOSE	[({dcb address,[{REREAD|LEAVE|RUN}]} ,...)] ,MF=(E,list)

Note: E-form operands will overlay those specified in the L-form. With MF=E, no more dcb addresses may be specified than were specified with the L-form CLOSE.

dcb address (all access methods)
    specifies the address of the data control block opened for the data set that is to be permanently or temporarily disconnected (closed) from the system. If more than one data control block is specified, two commas must be placed between each address to indicate the omission of the repositioning option (see below), even though it is applicable to BSAM and QSAM only.

    Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address.

REREAD|LEAVE|RUN (BSAM and QSAM)
    specifies the volume repositioning that is to be performed as a result of closing. This operand is applicable to volume disposition of magnetic tape devices only; it is ignored for other devices.

Specified as:
REREAD - the current volume is positioned to process the data set again.
LEAVE - the current volume is positioned to the logical end of data on the volume.
| RUN - the current volume is to be rewound and unloaded.

Default: LEAVE

TYPE=T (BSAM and VAM only)
indicates that labels are created and volumes are positioned, but the fields of the data control block are not altered. The data set can be processed without issuing another OPEN macro instruction. If TYPE=T is designated, it applies to all of the associated data control blocks.

After this macro instruction has been executed, the user's program can issue other macro instructions directed toward processing the data set because the data control block remains in OPEN status.

Specified as: TYPE=T

Default: If this operand is omitted, the close is permanent (and the data set cannot be processed without issuing another OPEN macro instruction).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Errors shown in Figure 4 cause the results indicated.

Errors	Result
Permanently or temporarily closing a data control block that is not open	No action
Temporarily closing (TYPE=T) a data control block that has not been opened for BSAM or VAM	No action
Permanently closing when the dcb address operand does not specify the address of a data control block	Task terminated
Temporarily closing (TYPE=T) when the dcb address operand does not specify the address of a data control block	Unpredictable
Permanently closing a data control block containing an invalid DSORG specification	Task terminated

Figure 4. Results of errors in closing a data set

Programming Notes: Any number of data control block addresses and associated options (BSAM and QSAM) may be specified in the CLOSE macro instruction. This makes it possible to close data control blocks and their associated data sets in parallel, which is more efficient than to close them individually.

VAM only

The CLOSE macro instruction releases any sharing interlocks set for the data set. Rules for sharing VAM data sets are given in Appendix K.

If more than one data control block is specified in a CLOSE macro instruction for VAM data sets, two commas must be placed between each to indicate the omission of the repositioning operand, which is applicable to BSAM only.

When a CLOSE (TYPE=T) is specified for a VAM data set, data pages, directory pages, and data set control blocks, where required, are written to external storage, ensuring that data set control information on external storage reflects the contents of the data pages on external storage. The data set remains in OPEN status. Nonpartitioned data sets are positioned (via SETL) according to the original OPEN option and data set organization prior to the completion of the CLOSE (TYPE=T). When partitioned data sets are processed, members for which a FIND has been issued are stowed (STOW type R) as during a normal CLOSE. A FIND macro instruction must be issued by the user if the member is subsequently to be reprocessed.

BSAM and QSAM only

The CLOSE (TYPE=T) macro instruction may be used to disconnect temporarily, from the problem program, one or more data sets if they reside on magnetic tape. An OPEN macro instruction must have been executed for each data control block specified in this form of the CLOSE macro instruction.

When the data sets are temporarily disconnected, labels are processed and user label exits are taken, if necessary. Magnetic tape volumes are repositioned as specified in this macro instruction.

Magnetic tape positioning varies depending on the options chosen in OPEN and CLOSE (TYPE=T) macro instructions and on whether the data set uses labels. Figure 5 defines a final position number for labeled and unlabeled tapes and Figure 6 relates the options chosen in OPEN and CLOSE macro instructions to positioning of tape volumes.

User trailer-label exits are taken for a data set processed for INOUT or OUTIN if the last operation was a WRITE. No user trailer label exits are taken if the last operation was a READ.

Position	Labeled Tape	Unlabeled Tape
1	Preceding data set header label group on current volume	Preceding first data block of portion of data set resident on current volume
2	Following tape mark that terminates trailer label group of data set on current volume	Following tape mark that terminates last data block of portion of data set resident on current volume

Figure 5. Final magnetic tape positions

Option of OPEN specified as	Other Factors Influencing Positioning	Direction of Last Input Operation	Volume repositioning as Specified in CLOSE	
			LEAVE	REREAD
OUTPUT	--	Not applicable		
OUTIN (BSAM only)	--	Not determining factor		
INOUT (BSAM only)	At least one WRITE operation in this data set	Not determining factor	Position 2	Position 1
INPUT	--	Forward		
INOUT (BSAM only)	No WRITE operation executed in this data set	Forward		
RDBACK	--	Forward		
INPUT	--	Backward		
INOUT (BSAM only)	No WRITE operation executed on this data set	Backward	Position 1	Position 2
RDBACK		Backward		

Figure 6. Factors determining magnetic tape positioning in BSAM and QSAM

If the data set resides on a magnetic tape, the following concerns the writing of trailer labels:

1. If the data set was opened for OUTIN or INOUT and the last I/O operation was a WRITE, then CLOSE or CLOSE (TYPE=T) both cause trailer labels to be written. If CLOSE (TYPE=T) is issued, additional READ or WRITE macro instructions are accepted without issuing a new OPEN macro instruction.

2. If the data set was opened for OUTIN or INPUT and the last I/O operation was a READ, and then a CLOSE or CLOSE (TYPE=T) was issued, additional READ and WRITE macro instructions are accepted without a new OPEN macro instruction being given.

3. If the data set was opened for OUTPUT, a CLOSE or CLOSE (TYPE=T) each cause trailer labels to be written. If a CLOSE (TYPE=T) is issued, additional WRITE macro instructions are accepted without a new OPEN macro instruction being given.

4. If the data set was opened for INPUT or RDBACK, a CLOSE or CLOSE (TYPE=T) does not cause trailer labels to be written. If CLOSE (TYPE=T) is issued, additional READ macro instructions are accepted without a new OPEN macro instruction being given.

<u>L- and E-Form Use</u>: The format of the parameter list generated by the CLOSE macro instruction is described in Appendix L.

For example:

```
        JOE     CLOSE     (ADCB,,BDCB,,),MF=L
        TERI    CLOSE     (,,PRODCB,,AXDCB),MF=(E,JOE)
```

When the E-form macro instruction is executed, the data control block PRODCB replaces the data control block BDCB in the parameter list, and the data control block AXDCB is added to the parameter list in the position reserved by the two commas following BDCB in the L-form. Thus, data control blocks with symbolic addresses ADCB, PRODCB, and AXDCB are closed.

Examples:

For BSAM or QSAM:

EX1 closes the data set associated with data control block INVEN with no repositioning. EX2 closes the two data sets associated with data control blocks INVEN and REPORT with different options. EX3 closes data sets associated with two data control blocks. Since the volume repositioning option is omitted in EX3, volume disposition is defaulted to LEAVE. EX4 generates a parameter list for closing INVEN, and EX5 closes INVEN.

```
        EX1     CLOSE     (INVEN,LEAVE)
        EX2     CLOSE     (INVEN,LEAVE,REPORT,REREAD)
        EX3     CLOSE     (INVEN,,MASTER)
        EX4     CLOSE     (INVEN,LEAVE)
        EX5     CLOSE     MF=(E,EX4)
```

For VAM:

EX1 closes data sets associated with two data control blocks. EX2 generates a parameter list for closing INVEN, and EX3 closes INVEN.

```
        EX1     CLOSE     (INVEN,,MASTER)
        EX2     CLOSE     (INVEN),MF=L
        EX3     CLOSE     MF=(E,EX2)
```

CNTRL -- Control On-Line Tape Drives (R)

The CNTRL macro instruction (for BSAM) performs repositioning operations on magnetic tape drives.

Name	Operation	Operand
[symbol]	CNTRL	dcb address,action[,number]

dcb address
    specifies the address of the data control block opened for the data set being processed.

| Specified as: Register notation (1 through 12), or an RX address

action
| specifies the positioning to be performed.

    Specified as: One of the three-character codes in Figure 7, or (0), in which case one of the abbreviated two-character codes must be placed in the two high-order bytes of register 0 before this macro instruction is executed.

Code	Abbreviated Code	Effect
BSF	BF	Moves backward past a tapemark. A number operand of 1 is assumed.
BSM	BM	Moves backward past a tapemark and forward spaces over the tapemark. A number operand of 1 is assumed.
BSR	BR	Backspaces over a number of blocks on magnetic tape, the number of blocks being specified by the number operand. One block is assumed if the number operand is omitted.
ERG	ER	Executes an erase gap for magnetic tape.
FSF	FF	Moves forward past a tapemark. A number operand of 1 is assumed.
FSM	FM	Moves forward past a tapemark and backspaces over the tapemark. A number operand of 1 is assumed.
FSR	FR	Forward spaces over a number of blocks on magnetic tape, the number of blocks being specified by the number operand. One block is assumed if the number operand is omitted.
REW	RW	Rewinds magnetic tape.
WTM	WM	Writes a tapemark on magnetic tape. A number operand of 1 is assumed.
RUN	RU	Rewinds and unloads magnetic tape.
SNS	SN	Obtains sense information on the tape or direct access device for the data set being processed (a pointer is returned in register 1 to a DECB containing the sense information in DECASB). Up to 3 bytes of sense data will be read according to device type currently in use. Unused sense bytes will be returned as binary zero.

Figure 7. Tape control options

number
    specifies the number of blocks to forward space or backspace on magnetic tape (as specified by the action operand).

    Specified as: A positive decimal integer, maximum 32,767. Alternatively, (0) may be specified for the action operand and the number may be placed in the two low-order bytes of register 0 before execution of the macro instruction. The parameter register is used only if the action code is specified in the high-order bytes of the register.

    Default: If BSR or FSR is specified for the action operand, the default for the number operand is 1. If the action code specifies BSF, BSM, FSF, FSM, WTM, or SNS, the system uses a number value of 1 no matter what is specified by the user.

CAUTIONS: If magnetic tape positioning is performed, an uncorrectable tape-spacing error results in linkage to the user's SYNAD routine; this does not apply to action codes REW or RUN. Refer to Appendix B for a discussion of SYNAD.

Abnormal termination occurs if:

1. The action code is undefined or not applicable.

2. An operation is attempted for an invalid device type.

3. A SYNAD-type error occurs and the user has not provided a SYNAD address.

4. The data control block specified by the user has not been validly opened.

5. A repositioning operation is attempted after a permanent error.

6. The requested operation did not succeed.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Return Data: If the repositioning operation is successful, register 15 contains binary zeros; otherwise, it contains, in its two low-order bytes, a count of the remaining number of forward spaces or backspaces that were not completed.

Programming Notes: READ and WRITE operations must be checked for completion before the CNTRL macro instruction is issued.

Control is returned to the user if a tape mark or a load point is encountered during an attempt to forward space or backspace blocks; control is not passed to the SYNAD routine.

COMMAND -- Enter Command Mode (P)

The COMMAND macro instruction switches the task from program mode to command mode to allow the user to enter commands. This macro instruction causes an unconditional pause, and executes whether the task is conversational or nonconversational. Any commands may be issued from the terminal during conversational program stoppage. If the stopped program is nonconversational, the SYSIN data set will be interrogated for the next commands. The task can be switched back to program mode by issuing a RUN command.

The word COMMAND followed by the optional message specified is written on SYSOUT.

Name	Operation	Operand
[symbol]	COMMAND	{address of message|'message'}

address of message
specifies the location of the message to be written (see below).

| Specified as: Register notation (1 through 12) or an RX address of the location that contains the message as a character string. The first byte of the message must contain the length of the message (in bytes).

message
specifies the message to be issued.

Specified as: The message itself, enclosed in apostrophes.

Initialization: This macro instruction cannot be assembled in a privileged module unless the most recently issued DCLASS macro instruction in the assembly specified USER (see Appendix M) or if the DCLASS option is USER by default.

Programming Notes: If the user has control of interruptions before issuing a COMMAND macro instruction, the system regains control until a RUN command is issued.

Examples: In EX1 the message is supplied as text. In EX2 the message is given at location BEMEL.

```
EX1   COMMAND    'PROG IN SUBRTN SQROOT'
EX2   COMMAND    BEMEL
```

CONSEG -- Connect Disconnected Segment Group (0)

The CONSEG macro instruction connects to an unassigned portion of a user's address space, a disconnected group. The disconnected name is deleted.

L-form:

Name	Operation	Operand
symbol	CONSEG	[DSNAME=,RNAME=,[MF=L

E-form:

Name	Operation	Operand
[symbol]	CONSEG	[DNAME=,RNAME=,ADDRESS=,] MF=(,list)

Standard-form:

Name	Operation	Operand
[symbol]	CONSEG	DNAME=[,RNAME=,ADDRESS=]

Note: All operands are keyword.

DNAME=

specifies the eight character EBCDIC name of a disconnected segment group to be connected.

Specified as: name enclosed within apostrophies; in E or standard form only, as the address of DNAME expressed as a relocatable expression, Rx address, or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the disconnected segment group name.

Default: None.

CAUTION: DNAME must be specified in standard form. DNAME must be specified in L-form and/or E-form macros which form an executable group.

Any user specified DNAME beginning with SYS will be rejected by the system.

RNAME=
specifies the eight character EBCDIC name to which the disconnected segment group is to be attached.

Specified as: name enclosed within apostrophies; in E or standard form only, as the address of RNAME expressed as a relocatable expression, Rx address, or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the reserved segment group name.

Default: If this operand is omitted, the system will connect the group at the disconnected address if at disconnect time BOUND=Y was specified; otherwise, the group will be connected at any unassigned contiguous space available in the user's address space.

CAUTION: If the group was disconnected with BOUND=Y, operands RNAME and ADDRESS are ignored.

Any user specified RNAME beginning with SYS will be rejected by the system.

ADDRESS=
specifies the segment aligned relative address to which the disconnected segment group is to be attached. If RNAME is specified, address is the relative address offset from the beginning of RNAME. If RNAME is not specified, address is the relative address offset from zero (i.e., an absolute address).

Specified as: In the E or standard form only the address of ADDRESS, expressed as a relocatable expression, RX address, or register notation.

Default: Relative zero.

CAUTION: If BOUND=Y at disconnect time, ADDRESS and RNAME will be ignored at connect time.

Return Data: On return from execution of CONSEG all defaulted operands will be filled in with system assigned values. The address field in the nameseg parameter list will be set to an absolute address. Register 15 will contain a return code describing the success of the operation.

Return Codes

00	Successful
04	RNAME Invalid
08	DNAME Invalid
12	Segment not available to user class
16	Invalid address
20	Segment group overlap
32	Insufficient space available
40	System error

Register 1 contains the address of the nameseg parameter list.

Note: The DSECT, CHANSG covers the nameseg parameter list.

Programming Notes: The return codes in register 15 may be used to construct a branch table to handle the varying results from execution of the CONSEG macro.

Upon expansion of this macro, a set of input flags is constructed in the nameseg parameter list. They are:

X'80'	DNAME Specified
X'40'	RNAME Specified
X'20'	ADDRESS Specified

Upon execution of CONSEG, a set of output flags will be constructed with the above values including:

X'10'	BOUND=Y

If BOUND=Y was specified at disconnect time.

Caution: If the disconnected segment group being connected was specified with BOUND=Y at disconnect time, operands RNAME and ADDRESS are ignored. Also, the total area to which the group is connected must be unassigned.

COPYDS -- Copy Existing Data Set (S)

The COPYDS macro instruction copies a complete data set or one or more members of a partitioned data set. The resulting new data set is assigned the data set name furnished (as an operand) by the user. In addition, COPYDS may renumber the lines of a line data set. A copy of a member may be specified either as a new member of a partitioned data set, as a new data set by itself, or as a replacement for an existing member with the same name. A VAM data set may be copied as (that is, become) a member of a partitioned data set.

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	COPYDS	{address of operand string|
		'data set name 1[(member,...)],
		data set name 2[(member)],
		[ERASE={Y|N}],[starting line],
		[increment][,REPLACE={R|I}]'}

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	COPYDS	'data set name 1[(member,...)],
		data set name 2[(member)],[ERASE={Y|N}],
		[starting line],[increment][,REPLACE={R|I}]',MF=L

Note: A symbol is required in the name field.

E-form:

Name	Operation	Operand
[symbol]	COPYDS	MF=(E,list)

address of operand string
    specifies the address of the first operand in the operand string (see "Operand Strings" in Part II, Section 1).

    Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified in the E-form macro operand as a character string enclosed in apostrophes, as shown.

data set name 1
    specifies the data set name of the data set being copied. It must be cataloged or have been defined in a DDEF macro instruction or command.

    Specified as: The fully qualified name of:

    • a data set with one member name or a list of member names enclosed in parentheses, or

    • a data set with no members specified.

data set name 2
    specifies the data set name to be assigned to the copy of the data set. The data set must have been defined in a DDEF macro instruction or command unless a member of a cataloged partitioned data set is specified.

    Specified as: The fully qualified name of:

    • a data set with one member name enclosed in parentheses, or

    • a data set with no members specified.

    When the original and copy data sets are both partitioned and no member name is specified in this operand, the members are moved from the original to the copy data set with user data and aliases. Duplicate members are processed as described under the REPLACE operand. If a member name is specified, the original data set name must have exactly one member name specified; no user data or aliases are copied.

ERASE=
    specifies the disposition of the original data set member or members after being copied. This applies only to data sets on direct access devices. If a shared data set is to be copied and then erased, unlimited access to the data set must have been permitted.

    Specified as:
    Y - erase original data set member after it is copied.
    N - do not erase original data set member.

    Default: N

starting line
    specifies the starting line number of the data set copy if it is a line data set and renumbering is desired.

Specified as: A three- to seven-digit number, the last two digits of which must be zero. An all-zero starting line number is invalid.

Default: If increment is also defaulted, line numbering is not performed. If increment is not defaulted, the starting line number of the copy data set will be 100.

increment
specifies the value by which line numbers in the data set copy (if it is a line data set) are to be incremented when renumbering is desired.

Specified as: A three- to seven-digit number, the last two digits of which must be 0. An all-zero increment is invalid.

Default: If the starting line number is also defaulted, line numbering is not performed. If the starting line number is not defaulted, an increment of 100 is assumed.

REPLACE=
specifies the disposition of the members of the original data set that have duplicate names in the copy data set. This operand is only used if user data and aliases are to be copied.

Specified as:
R - the duplicates are to be replaced.
I - the duplicates are to be ignored.

Default: R

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: If both data sets specified are partitioned data sets and if the copy data set has no member name specified, the COPYDS macro instruction will copy:

- one member of the original data set,
- any number of members of the original data set, or
- all the members of the original data set, if the original data set has no member names specified. User data and aliases are also copied.

When multiple members of the original data set are specified, the copy data set must be partitioned. COPYDS replaces an existing member of the copy data set with the member of the original data set. If a member of the original data set has a duplicate name already in the copy data set, the REPLACE operand specifies that the member is to replace the duplicate (REPLACE=R), or is to be ignored (REPLACE=I). If an alias for a member of the original data set already appears as an alias for a different member of the copy data set, the member will not be copied, regardless of the REPLACE operand.

The COPYDS macro instruction does not differentiate between object modules and other members of partitioned data sets. For any member, user data and member name aliases are transferred along with the data and member name.

The COPYDS macro instruction is restricted to data sets on direct access or magnetic tape volumes. Data set organization is not altered by the use of a COPYDS macro instruction. The original and copy data sets must be defined with the same data set organization and record format. For example, the copy of a physical sequential data set has physical sequential organization, even though the device type may be changed. A VISAM data set can, however, be copied as VSAM and vice versa.

The user may specify a VISAM organization in the COPYDS macro instruction for a data set copy, even though the original data set organization is VSAM. In this case, each record of the original data set must contain a key. In addition, the user should define -- in the DDEF macro instruction or command for the data set copy -- the key length (KEYLEN), padding (PAD), and record key displacement (RKP) values. If he does not provide these values, no copy is made.

The user can copy only those data sets that belong to him or those to which he has been given access.

Return Data: At completion of execution of the COPYDS macro instruction, the low-order byte of register 15 contains one of the following hexadecimal codes:

Code	Significance
00	Successful completion
04	Invalid input parameters
08	Name of original data set not in catalog or task definition table (TDT)
0C	Data set not in catalog and no DDEF macro instruction or command has been executed for it
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 for new data set
1C	Original data set not VAM or SAM
20	Data set not on direct access or tape; command ignored
24	New data set member name already exists in POD
28	Data set copied. Old data set not erased; user does not have proper access
2C	Data set copied. New data set not renumbered; not a line data set
30	Data set copied and renumbered. Old data set not erased; user does not have proper access
34	Data set copied and original erased. New data set not renumbered; not a line data set
38	Data set copied; new data set not renumbered, and old data set not erased

Examples: In EX1, the operands are presented as a character string. In EX2, an address designates the location of the operands.

```
EX1      COPYDS    'DATASET,U'
EX2      COPYDS    OPLISTC
```

In EX3, the original data set name is VP1 and has members A,B,C. The copy data set name is VP2 and has members C, D, and E. Following the execution of EX3, VP1 contains C2, and VP2 contains A, B, the original C, D, and E (C was not copied because the REPLACE operand specifies that members with duplicate names are to be ignored).

```
EX3      COPYDS    'VP1(A,B,C),VP2,ERASE=Y,,,REPLACE=I'
```

CSTORE -- Control Section Store (S)

The CSTORE macro instruction enables the user, during program execution, to transform any set of contiguous virtual storage bytes into an object module consisting of a single control section.

Standard form:

Name	Operation	Operand
[symbol]	CSTORE	module name, entry point name,data address, data length,attribute

L-form:

Name	Operation	Operand
symbol	CSTORE	module name,entry point name,[data address], [data length],[attribute],MF=L

Note: A symbol is required in the name field.

E-form:

Name	Operation	Operand
[symbol]	CSTORE	,,[data address],[data length], [attribute],MF=(E,list)

Note: If an operand is omitted in the L-form, it must be specified in the E-form.

module name
: specifies the name to be assigned to the module created to contain the control section.

  Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

entry point name
: specifies the entry point name to be assigned to the specified address location.

  Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

data address
: specifies the address of the first byte of data to be included in the control section.

  Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, as register notation (2 through 12); in the E-form only, also as an RX address. |

data length
: specifies the number of bytes of data to be included in the control section.

  Specified as: An absolute expression; in the standard form and in the E-form, register notation (2 through 12).

attribute
specifies the attribute byte of the control section.

Specified as: An absolute expression; in the standard form and in the E-form, also in register notation (2 through 12). The meaning of the contents of the attribute byte is shown in Figure 8:

Bit	Control Section Attribute	
	Bit off	Bit on
0		system
1	nonprivileged	privileged
2		common
3	CSECT	prototype (PSECT)
4	private	public
5	read-write	read-only
6	fixed-length	variable-length
7	not used	

Figure 8. Control section attribute byte

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The module created by this macro instruction is stowed in the current JOBLIB. It can then be loaded by the program that created it, or by a subsequent program. When the module is loaded, no relocation takes place; therefore, it may contain no relocatable items. The resulting module consists of an unnamed control section that contains a copy of the hexadecimal text, beginning at the page boundary corresponding to or preceding the address specified as the starting address parameter, and terminating at the page boundary corresponding to or following the address computed from the fourth parameter. Thus, the resulting control section is always an integral number of pages in length.

When the module is loaded by the user, the module name, as well as the entry point name, points to the address computed by adding to the load address of the new module the page offset (if any) implied by the starting address. For example, assume that the user requests that a control section of 4093 bytes be created from the bytes beginning at virtual storage address 5D050. Two pages of hexadecimal text beginning at the page boundary address (in this case 5D000) corresponding to or preceding the specified starting address are transformed into an object module. The module and entry point names are offset from the page boundary by 50 to reflect the actual address (5D050) of the hexadecimal text which the user desires to place in a control section. Assuming that the new module is later loaded at 70000, the loaded module and control section will occupy two full pages beginning at 70000. The second page is required so that the new control section will include the last two bytes requested by the user. The new module and entry point names will be adjusted to reflect the offset and will both point to 70050.

The maximum control section size is one segment. The control section is created from any contiguous set of bytes, and is an integral number of pages in length. A control section is not built if the module or entry point names are duplicates of existing names in the current JOBLIB.

Subsequent loading of the created module is accomplished implicitly, by using an R- and V-type address constant for the entry point name or module name, or explicitly, by use of the LOAD macro instruction.

The common attribute (bit 2), if specified, will be ignored by the dynamic loader, since it treats all unnamed control sections as unique. The created module may contain no relocatable words (adcons) and can be referred to by the control section name or module name offsets.

Return Data: Upon completion of execution of the CSTORE macro instruction, the low-order byte of register 15 contains one of these hexadecimal codes:

Code	Significance
00	Successful completion
04	Module name or entry point name already in use

Example: This example indicates the macro instruction used to create a module named MYMODULE which contains one unnamed control section. The control section consists of the two pages of text taken from the bytes beginning at HERE, which is on a page boundary. The entry point name EPNAME points to the beginning of the control section, which has public and read-only attributes.

```
EX1    CSTORE    MYMODULE,EPNAME,HERE,8000,12
```

DCB -- Construct a Data Control Block (O)

The DCB macro instruction is one of the major sources (see Appendix F) by which the attributes of a data set may be described to the system. The attributes of a data set that can be provided via this macro instruction and the formats in which particular attributes can be specified are indicated below by access method (for MSAM, see System Programmer's Guide).

Format: The format of the DCB macro instruction varies, depending on the data set organization and the access method that is to be used, or was previously used, to perform I/O on that data set. All of the possible parameters that might be specified by a nonprivileged user in a DCB macro instruction are indicated by applicable access method in Figure 9.

Note that the operands can appear in any order, but must be in keyword form, separated by commas.

Name	Operation	Operand
[symbol]	DCB	(see figure below)

Operands	Applicable Access Methods					
	VSAM	VISAM	VPAM	BSAM	QSAM	IOREQ
[,DDNAME=data definition name]	X	X	X	X	X	X
[,DSORG=organization]	X	X	X	X	X	X
[,RECFM=record format]	X	X	X	X	X	
[,LRECL=record length]	X	X	X	X	X	
[,EODAD=end-of-data address]	X	X	X	X	X	
[,OPTCD=optional service]	X	X	X	X	X	
[,SYNAD=error routine]		X	X*	X	X	X
[,PAD=available space]		X	X*			
[,RKP=key field displacement]		X	X*			
[,DEVD=device]		1.	1.	X	X	X
[,KEYLEN=key length]		X	X*	X		
[,TRTCH=recording technique]				X	X	
[,MACRF=macro type]				X	X	
[,BLKSIZE=maximum block length]				X	X	
[,IMSK=error procedures]				X	X	X
[,EXLST=exit list address]				X	X	
[,NCP=check number]				X		X
[,BUFNO=number of buffers]				X		
[,BFALN=buffer alignment]				X		
[,BUFL=buffer length]				X		
[,BFTEK=buffering technique]				X		
[,BUFCB=buffer control block]				X		
[,EROPT=error option]					X	
* = VISAM members of a partitioned data set.						
1. = a value is assumed by the system.						

Figure 9. DCB operands and applicable access methods

DDNAME= (all access methods)
specifies the symbolic data definition name associated with a particular data set. This symbol provides the link that connects the attributes of the data set described by the DCB macro instruction with those specified by the DDEF macro instruction (or command), thereby providing the system with all the information necessary for processing the data set.

<u>Specified as</u>: A symbol (one to eight alphameric characters, the first of which must be alphabetic). The name specified for this parameter must be identical to the DDNAME parameter of the DDEF macro instruction that defines this data set. The only alternate source for this information is the user's program.

DSORG= (all access methods)
specifies the organization of the data set.

<u>Specified as</u>: The codes by which the various data set organizations can be specified, and the access methods with which they are applicable are:

Code		Organization	Applicable Access Methods
PS	--	physical sequential organization	BSAM,QSAM
PSU	--	physical sequential unmovable organization in which the data set contains location-dependent information with respect to this data set. Treated as PS by TSS.	BSAM,QSAM
VS	--	virtual sequential organization	VSAM
VI	--	virtual index sequential organization	VISAM
VP	--	virtual partitioned organization	VPAM
VIP	--	virtual index sequential member of a partitioned data set	VPAM
VSP	--	virtual sequential member of a partitioned data set	VPAM
RX	--	I/O request facility is being used	IOREQ

For an existing VP data set, only VP need be specified. The organization of the member (virtual sequential or virtual index sequential) is determined by FIND and placed in the DCB. However, when creating a new member, the user must specify either VIP or VSP.

This information can also be supplied by the user's program or the DDEF macro instruction (or command), but must be supplied before issuing an OPEN macro instruction.

RECFM= (all access methods)
specifies the format of the records in the data set.

<u>Specified as</u>: One of the following:

<u>For BSAM and QSAM</u>:
```
U[T] [A|M]
V[B|T] [A|M]
F[B|S|T|BS|BT|BST|ST] [A|M]
```

Where the record format is:

U -- undefined-format records
V -- variable-length records
F -- fixed-length records

Where the physical attributes are:

B -- blocked records
S -- standard data set; no truncated blocks or unfilled tracks
T -- track overflow employed

Where the record contains:

A -- FORTRAN control character
M -- machine code control character

If A or M is not specified, no control character is assumed. Refer to Appendix D for a discussion of control characters.

Absence of any of the physical attribute mnemonics implies the opposite of that attribute. For instance, writing RECFM=V implies: variable-length, unblocked records, no control character, and no track overflow feature.

This information can also be supplied by the user's program, the DDEF macro instruction (or command), or the data set label.

For VAM data sets: All VAM data sets can be organized as fixed- or variable-length records, but only VSAM and VPAM records can be specified as having undefined format.

```
U[A|M]   (applicable to VSAM, VPAM only)
V[A|M]   (applicable to VSAM, VISAM, or VPAM)
F[A|M]   (applicable to VSAM, VISAM, or VPAM)
```

Where the record format is:

U -- undefined-format records
V -- variable-length records
F -- fixed-length records

Where the record contains:

A -- FORTRAN control character
M -- machine code control character

If A or M is not specified, no control character is assumed. Refer to Appendix D for a discussion of control characters.

This information can also be supplied by the user's program, the DDEF macro instruction (or command), or the data set label.

LRECL (VAM, BSAM, and QSAM)
specifies the length in bytes of a logical record. For format-F records, this operand specifies the length of each record in the data set. For format-V and -U records, the user must insert the maximum expected value before the data set is opened. When reading format-U or -V records, the corresponding field in the data control block (DCBLRE) contains the length in bytes of the record just read.

Specified as: An absolute expression. The maximum that may be specified for BSAM data sets is 32,760, the maximum for VSAM is 1,048,576, and the maximum for VISAM is 4000.

This information can also be supplied by the user's program, the DDEF macro instruction (or command), or the data set label.

EODAD= (VAM, BSAM, and QSAM)
specifies the address of the user's end-of-data routine for input data sets. This routine is entered if the user requests a record when there are no more records in the data set. If no routine has been provided, and the end-of-data condition has been encountered, the task is abnormally terminated. (Refer to Appendix C.)

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

If the symbol supplied is an external symbol, it must also appear as the operand of an assembler language EXTRN statement in the same object module as the DCB macro instruction.

The only alternate source for this information is the user's program.

OPTCD= (VAM, BSAM or QSAM)
specifies an optional service to be provided.

Specified as:
W- perform a write validity check; for direct access devices only
A- ASCII tape request

Default: No service is performed unless the code is specified from an alternate source.

This information can also be supplied by the user's program, the DDEF macro instruction (or command), or the data set label. If not supplied by any source, the service is not performed.

SYNAD= (VISAM data sets, VISAM members, BSAM, QSAM, or IOREQ)
specifies the address of the user's synchronous error exit routine. The routine is entered if input/output errors result from an attempt to process data records.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

If the address specified is an external symbol, the symbol must also appear as the operand of an assembler language EXTRN statement in the same program module as the DCB macro instruction.

The only alternate source for this information is the user's program.

Default: If no routine is specified and the system encounters a condition that would cause control to be given to the SYNAD routine, the task is abnormally terminated.

PAD= (VISAM data sets or VISAM members)
specifies the percentage of space to be left available within the pages of a virtual index sequential data set, thus providing for insertions within the pages.

Specified as: An absolute expression; the maximum value that may be specified is 50.

This information can also be supplied by the user's program, the DDEF macro instruction (or command), or the data set label.

RKP= (VISAM data sets or VISAM members)
specifies the displacement (relative key position) of the key field from the first byte of a logical record.

Note: For format-V records, the logical record includes the length field as the first four bytes.

Specified as: An absolute expression.

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

DEVD= (BSAM, QSAM, IOREQ, or VAM)
specifies the device on which the data set resides. Additional keyword operands are available, as shown below, to provide device-dependent information to device-dependent parameter bytes in the data control block.

Specified as: One of the following codes:
DA - direct access device
TA - magnetic tape
PR - printer (IOREQ)
RD - card reader (IOREQ)
PC - card punch (IOREQ)

If DA is specified, KEYLEN may also be specified; if TA is specified, TRTCH may also be specified. For VAM, DA is assumed, and the user can supply the KEYLEN operand if desired.

Note: Since nonprivileged users cannot address unit record devices directly, they may not specify PR (printer), RD (card reader), or PC (card punch). These devices may be specified only by users with proper system authorization. See System Programmer's Guide.

This information can also be supplied by the user's program, the DDEF macro instruction or command, or the data set label.

KEYLEN= (VISAM data sets or VISAM members)
specifies the length in bytes of the key associated with a physical record. When a record is read or written, the number of bytes transmitted is equal to the key length plus the record length. This operand is specified only if DA is specified.

Specified as: An absolute expression, maximum value 255.

This information can also be supplied by the user's program or the DDEF macro instruction or command.

TRTCH= (BSAM, QSAM)
specifies the recording technique for 7-track tape. This operand is specified only if TA is specified.

Specified as: C,E,T,TE, or ET, where:
C -- Data conversion feature available. If data conversion is not available, only format-F and format-U are supported.
E -- Even parity is used.
T -- BCD to EBCDIC translation is required.

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

Default: If not supplied by any source, odd parity and no translation is assumed.

Note: The system standard for 7-track tapes is TE: even parity, BCDIC translated.

MACRF= (BSAM and QSAM only)
specifies the type of macro instructions to be used in processing a particular data set.

Specified as: One of the following:

For BSAM:

```
(R[C|P])
(W[C|P])
(R[C|P],W[C|P])
```

R -- READ macro instructions
W -- WRITE macro instructions

Optional modifiers:

C -- CNTRL macro instruction
P -- POINT macro instruction

For QSAM:

```
(G[S])
(P[S])
(G[S],P[S])
```

G -- GET macro instructions
P -- PUT macro instructions

Optional modifier:

S -- SETL macro instruction

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

BLKSIZE= (BSAM or QSAM)
specifies a decimal value for the maximum block length in bytes. Maximum value of BLKSIZE is 32,760.

Specified as: An absolute expression, maximum value 32,760.

This information can also be supplied by the user's program, the DDEF macro instruction (or command), or the data set label.

IMSK= (BSAM or QSAM error recovery mask)
specifies which system error handling procedures, if any, are to be invoked.

Specified as: A four-byte hexadecimal number whose bit pattern indicates the procedures to be invoked.

If FFFFFFFF is written, the system is to apply all optional error recovery procedures.

If 00000000 is written, the system is to apply none of its optional error recovery procedures.

If any other four-byte hexadecimal number is written, and if an error occurs, the system applies its error recovery procedures to those errors indicated by a 1-bit in the mask.

The first two bytes correspond to the first two bytes of the channel status word, and the other two bytes correspond to the first two sense bytes. Bit positions in each byte for specification of system error recovery procedures are:

```
XXXXXXXX     XXXXXXXX     ABCDEFYY     YYYYYYYY
```

where a 1-bit in a given position indicates that the system is to handle the associated error condition:

X = System never tests this bit to determine entry to retry routines
Y = Device-dependent conditions
A = Command reject
B = Intervention required
C = Busout check
D = Equipment Check
E = Data Check
F = Overrun

Default: FFFFFFFF

IMSK= (IOREQ error recording mask)
specifies a four-byte hexadecimal number whose pattern indicates what system errors are to be recorded.

If IMSK=FFFFFFFF (system default value) is specified, no error recording occurs. Although this default value invokes no error recording, channel control check, interface control check, and channel data check errors are always recorded.

CAUTION: Unlike most mask fields, which request testing for a condition when the mask bit is set to one, IOREQ tests for required error recording when the mask bit is set to zero.

The first two IMSK bytes correspond to the two channel status word status bytes. IOREQ does not check these bytes in determining if error recording is required. The second two IMSK bytes correspond to the first two bytes of sense information. The IMSK bytes have the following format:

```
XXXXXXXX     XXXXXXXX     CCCCCCCC     CCCCCCCC
```

Where:

- X is a status bit (not checked in determining if error recording is required). Any hexadecimal number may be placed in these bytes.

- C is a sense bit. When any of the first 16 sense information bits are set to one and the corresponding C bit is set to zero, the corresponding error is recorded. (One or more sense bits are set to one when a successful sense operation is performed after a unit check status condition has occurred.)

Note: A 0-bit in a given position indicates that recording of the corresponding error is required. Error recording occurs if one or more sense bits are set to zero. Error recording should only be requested for equipment errors. Software (that is, command reject) and operational (that is, intervention required) problems should not be recorded.

EXLST= (BSAM or QSAM)
specifies the address of an exit list supplied by the user. See Appendix A for explanation of the exit list.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

This information can also be supplied by the user's program.

NCP= (BSAM or IOREQ)
specifies the number of consecutive READ or WRITE macro instructions that may be issued before a CHECK macro instruction.

Specified as: An absolute expression, maximum value 99.

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

BUFNO= (BSAM)
specifies the number of buffers to be assigned to the data control block.

Specified as: A binary absolute expression, maximum value 255.

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

BFALN= (BSAM)
specifies boundary alignment of buffers. This field is ignored in TSS. Every buffer is automatically aligned on a doubleword boundary.

BUFL= (BSAM)
specifies a decimal number which is the length in bytes of each buffer to be obtained for a buffer pool.

Specified as: An absolute expression, maximum value 32,760.

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

Default: If not supplied by any source, the length is considered equal to the BLKSIZE operand.

BFTEK= (BSAM)
specifies that simple buffering is to be employed. In simple buffering, a data set is associated with a specific group of buffers. A data set always uses buffers obtained from the pool assigned to its data control block at the time it is opened. Records can be moved between a buffer and an independent work area, processed within a buffer, or moved from an input buffer to an output buffer.

Specified as: BFTEK=S

This information can also be supplied by the user's program or the DDEF macro instruction (or command).

Default: BFTEK=S

BUFCB= (BSAM)
specifies the address of a buffer control block.

Specified as: Register notation (2 through 12), or a relocatable expression.

This information can also be supplied by the user's program.

EROPT= (QSAM)
When using GET/PUT macro instructions to process a sequential data set, an I/O error may occur. The user may specify one of three automatic error options to be used if there is no SYNAD routine or if the SYNAD routine returns control to the user's program.

Specified as:

ACC -- accept the erroneous block and continue processing
SKP -- skip the erroneous block and process the next record
ABE -- abnormally terminate the task

Note: If the EROPT and SYNAD fields are not completed, the ABE option is assumed.

The choice of action that can be specified depends on which processing method (option) is specified in the OPEN macro instruction for the data set. The allowable combinations are as follows:

Action Operand	OPEN Option
ACC	INPUT, RDBACK, or UPDAT
SKP	INPUT, RDBACK, or UPDAT
ABE	INPUT, OUTPUT, RDBACK, or UPDAT

Programming Notes: During the assembly of a source program, the DCB macro instruction reserves storage space in a user program in which the attributes of a data set being described to the system may subsequently be placed. This storage area is known as a data control block (DCB) and is created at assembly time, in line, wherever the DCB macro instruction appears in a user's source program. The reserved control block has a fixed length and consists of two contiguous parts: a common portion, in which all information that is access method independent is to be placed, and an access method dependent portion.

In addition to furnishing the storage area for holding the attributes describing a data set, the DCB macro instruction can also be used optionally, at execution time, to specify many of a data set's attributes. A user might furnish the system with such information as the data set organization, its record format, whether or not buffering is to be used during I/O operations, the type of device the data set resides on, and the addresses of user written routines for handling I/O errors, processing labels, end-of-data-set processing, and data control block modification routines. Any such attributes specified with a DCB macro instruction are automatically placed in appropriate positions in the reserved storage area.

When the storage area reserved by the DCB macro instruction is filled with the attributes of a data set, it becomes the principal control block used to supply the system with information describing a particular data set or device. Once optional attributes have been placed in the control block, the DCB routine returns to the user's program. All data management macro instructions provided with TSS refer to this control block for pertinent data when they are executed.

DCBD -- Provide Symbolic Names for a Data Control Block (O)

The DCBD macro instruction generates a dummy control section (DSECT) that provides symbolic names for the fields in a data control block. With proper initialization of a base register, the user may gain access to all fields of a data control block.

The following conventions have been adopted:

1. The name of the dummy control section is CHADCB. (An EQU is included in the DSECT to allow use of the alternative OS name IHADCB).

2. The name of each field begins with the characters DCB, followed by the keyword operand that represents the field in the DCB macro instruction. If the resulting name is longer than six characters, it is truncated on the right to six characters; that is, the field represented by the operand BLKSIZE= should be written DCBBLK. (Refer to Appendix F.)

The attributes of each data control block field are defined in the dummy control section (DSECT). Data control block fields containing addresses are aligned on fullword boundaries.

Name	Operation	Operand
[symbol]	DCBD	

Note: A symbol may be present in the name field but will not be generated. There are no operands.

CAUTION: The DCBD macro instruction may be used only once in an assembly module.

Programming Notes: The macro instruction may appear at any point in a control section. The data control blocks to be accessed need not appear in the same control section as the DCBD macro instruction.

Example: This example illustrates how a program can gain access to a field in a data control block through use of the DCBD macro instruction. The load address (LA) instruction is used to place the address of the data control block in register 5.

A USING statement establishes a base register for CHADCB. The store operation (ST) places the value contained in register 6 into the specified field of the data control block pointed to by register 5. DCBLRE is the field associated with logical record length. The user previously loaded register 6 with the value he desired to be in DCBLRE.

```
         .
         .
         .
MYDCB    DCB        DDNAME=MYDCB,MACRF=G  (other DCB operands)
         .
         .
         .
         LA         5,MYDCB
         USING      CHADCB,5
         ST         6,DCBLRE
         .
         .
         .
         DCBD
```

DDEF -- Define a Data Set (S)

The DDEF macro instruction defines a data set and describes its characteristics to the system. Every data set that is referred to by an object program during execution must be defined by a DDEF macro instruction or command. Each DDEF macro instruction is valid only during the session in which it is issued; thus data sets defined for one session must be redefined at every session that involves reference to them.

Note: The following description applies to the DDEF macro instruction used to define a VAM data set on public storage. To define nonstandard public data sets or any private data set, refer to the detailed description of the DDEF macro instruction given in Appendix G.

Standard form (See "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	DDEF	{address of operand string| 'data definition name[,{VI|VP|VS},DSNAME=name]'| PCSOUT}

L-form (See "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	DDEF	{'data def name[,{VI|VP|VS},DSNAME=dsname]',MF=L| PCSOUT}

Note: A symbol is required in the name field.

E-form:

Name	Operation	Operand
[symbol]	DDEF	MF=(E,list)

address of operand string
specifies the address of the first operand of the operand string.

Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified as a character string enclosed in apostrophes, as shown.

data definition name
specifies the symbolic data definition name associated with this data set definition. It provides the link between the data control block in the program and the data set definition.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic). The user is not allowed to use a data definition name that begins with SYS, since system-reserved data definition names are prefixed with those characters.

PCSOUT
specifies that the program control system is being used and a data set is being defined for dumps. A PCSOUT type of DDEF command or

macro instruction is required in a task if the DUMP command is to be employed.

Specified as: PCSOUT

{VI|VP|VS}
specifies the organization of the data set.

Specified as:
VI - virtual index sequential
VP - virtual partitioned
VS - virtual sequential

Default: If neither VI nor VP nor VS is specified, the data set organization assigned during system generation is assumed.

DSNAME=
specifies the name of the data set being defined; that is, the name under which the data set may be cataloged or temporarily referred to.

Specified as: The fully qualified name of: a partitioned or non-partitioned data set, a member of a partitioned data set, or a partitioned or nonpartitioned generation of a generation data group (identified by an absolute generation name or relative generation number).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: Before the user can employ the DUMP command in his task, he must issue a PCSOUT type of DDEF macro instruction or command. Such a DDEF macro instruction or command requires PCSOUT as the first operand, followed by the dsname operand.

At least the definition name and DSNAME operands are required for a previously cataloged data set. Only the data definition name is needed for a new VAM data set. In either case, the data set conforms to the current installation standards.

The DDEF macro instruction or command causes a system entry to be established for the DDEF information so that allocation routines and access methods can refer to it. The link between this information and the problem program's reference to the data set (that is, the data control block) is the data definition name. The entry containing the DDEF information is maintained until the task is concluded or until, through the RELEASE macro instruction or command, the data set is released.

The DDEF macro instruction or command may be used in conversational and nonconversational tasks.

If the user's problem program is being executed in conversational mode and an undefined data definition name is referred to, prompting messages for DDEF operands are issued to the user regardless of confirmation option.

The user may change the data definition name assigned in a previous DDEF macro instruction or command by using a DDEF macro instruction with a new data definition name. The only operands used in this case are data definition name, DSNAME, and disposition (OLD). (See Appendix G.)

The new data definition name is then assigned and the old data definition name eliminated.

Return Data: At completion of execution of a DDEF macro instruction, the low-order byte of register 15 contains one of the following hexadecimal codes:

Code	Significance
00	Successful completion
04	Data set name undefined
08	Data set name not unique
0C	Attention interruption
10	DSORG in DDEF parameter list is not the same as DSORG in the catalog
14	Non-existent generation name specified
18	DSNAME not fully qualified
1C	Volume could not be mounted
20	Space not available
40	Data definition name not unique
80	Other

DEL -- Delete Catalog Entry (S)

The DEL macro instruction deletes one or more catalog entries for a data set or group of data sets. When a generation data group name is supplied, the macro instruction deletes the catalog entries for all generations in that group. Similarly, a partially qualified data set name results in catalog entries being deleted for all data sets with the same initial name component.

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	DEL	{address of data set name|'data set name'}

L-form (See "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	DEL	'data set name',MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	DEL	MF=(E,list)

address of data set name
specifies the address of a location containing the data set name (see below).

Specified as: Register notation (2 through 12) or a relocatable expression. Note that the data set name can also be specified in the macro operand as a character string enclosed in apostrophes, as shown.

data set name
specifies the name of the data set whose catalog entry is to be deleted.

Specified as: A character string enclosed in apostrophes. The data set name can be:

- The fully qualified name of: a partitioned or nonpartitioned data set, or a partitioned or nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number).

- The partially qualified name of any type of data set, including a generation data group.

If the data set is not shared, it must reside on a private volume; the data set name may be the sharer's name for a data set owned by another user.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix F). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTIONS: This macro instruction deletes the catalog entries for data sets on private volumes only. A macro instruction that attempts to uncatalog data sets residing in public storage is ignored and a diagnostic message is produced if it is issued in conversational mode. Only the ERASE command can be used to remove such data sets from the system. However, the DEL macro instruction can be used to delete a sharing descriptor from the sharer's catalog.

If a user issues a DEL macro instruction against a shared data set for which a BULKIO request is pending, the delete request is not honored.

Programming Notes: When a cataloged entry for a private VAM data set is deleted, that data set can only be recataloged by issuance of the EVV command (see Command System User's Guide, GC28-2001). Data sets on public volumes must be erased if they are to be uncataloged. The user must, therefore, use the ERASE command to remove those data sets from the system, except when he is a sharer.

Return Data: At completion of execution of the DEL macro instruction, the low-order byte of register 15 contains one of these hexadecimal codes:

Code	Significance
00	Successful completion
08	Invalid return from NEXTPAR
0C	Invalid data set name (input preceded by left parenthe-

	sis) - NEXTPAR
10	No data set name supplied after verb
14	Return code from CHFCXDS was not divisible by four
24	Data set not cataloged
28	Data set on a public volume
2C	Data set name is a member of a partitioned data set
34	Sharer does not have unlimited access to data set

DELETE -- Delete a Loaded Module (R)

The DELETE macro instruction indicates that a copy of a specified module, which had been placed in virtual storage, is no longer required. This specified module must have been previously acquired by the issuance of a LOAD macro instruction or an explicit CALL macro instruction. Upon execution of this macro instruction, the specified module, and any associated modules, are deleted from the issuing task's virtual storage.

Name	Operation	Operand
[symbol]	DELETE	{EP=symbol|EPLOC=adcon group address}

EP=
specifies the external name of the module to be deleted. This external name must be the name of a control section, the name in the operand field of an assembler language ENTRY statement, or a module name.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

EPLOC=
specifies the address of the delete adcon group representing the module to be deleted.

This delete adcon group is generated by:

```
ADCON      DELETE,EP=external name
```

| Specified as: Register notation (1 through 12), or the RX address of the adcon group.

Examples: 1) If the module associated with the external name EARL is to be deleted, and the following ADCON macro instruction is supplied:

```
DAVE    ADCON      DELETE,EP=EARL
```

then the macro instruction MAX DELETE EPLOC=DAVE causes the module associated with EARL to be deleted.

2) The module associated with the external symbol ALPHA is deleted.

```
SARP    DELETE     EP=ALPHA
```

3) Before this DELETE macro instruction is executed, the address of the delete adcon group must be loaded into register 1; for example, LA 1,EARL. The effect of this macro instruction is then the same as in Example 1.

```
NAM     DELETE     EPLOC=(1)
```

DELREC -- Delete a Record (R)

The DELREC macro instruction (for VISAM) deletes a specified record from a virtual index sequential data set. The record may be specified by its key or its retrieval address.

Name	Operation	Operand
[symbol]	DELREC	dcb address,{K|R},limit

dcb address
: specifies the address of the data control block opened for the data set being processed.

  Specified as: Register notation (1 through 12), or an RX address.

{K|R}
: specifies whether the record will be deleted by key or retrieval address.

  Specified as:
  K - record key
  R - retrieval address as obtained by the user from DCBLPA in the data control block

limit
: specifies the address of a field containing either the record key or the retrieval address. The retrieval address must be in a four-byte field, beginning on a doubleword boundary.

  Specified as: Register notation (0 or 2 through 12), or an RX address.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Exceptional conditions, including "invalid retrieval address" and "key not found", resulting from the execution of a DELREC macro instruction, cause control to be passed to the user's synchronous error exit (SYNAD) routine. In this case, the general registers and the exceptional condition fields of the data control block are set as shown in Appendixes B and F. DELREC by retrieval address may not be used with a shared data set.

Programming Note: This macro instruction releases any page-level interlocks established by other macro instructions referring to the same DCB. Rules for sharing VISAM data sets are given in Appendix K.

DELSEG -- Delete Disconnected Segment Group (O)

The DELSEG macro instruction deletes a disconnected segment group. The name and length are forgotten by the system. Space allocated on auxiliary storage will be returned to the system.

Name	Operation	Operand
[symbol]	DELSEG	DNAME=disconnected segment group name

DNAME=

Specifies the eight character EBCDIC name of an existing disconnected segment group.

<u>Specified As</u>: Disconnected segment group name enclosed within apostrophies, the address of DNAME expressed as relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the disconnected segment group name.

<u>Default</u>: None.

<u>Return Data</u>: On return from DELSEG, register 15 will contain a return code describing the success of the operation.

<u>Return Codes</u>

00	Successful
08	DNAME Invalid
12	Segment group not available to user class
40	System error

<u>CAUTION</u>: Any user specified DNAME beginning with SYS will be rejected by the system.

<u>Examples</u>:

```
1.  DL1    DELSEG      DNAME='DNAME1'

2.  DL2    DELSEG      DNAME=DNM
             .
             .
             .
    DNM    DC          CL8'DNAME'

3.  DL3    DELSEG      DNAME=(3)
```

Execution of example 3 assumes that the disconnected segment group name is contained in registers 3 and 4.

<u>DEQ -- Dequeue Resource Access Request (R)</u>

The DEQ macro instruction is used to release a resource access request for a resource, or to delete all resource access requests for a particular task.

Name	Operation	Operand
symbol	DEQ	NAME=name of resource [,VMADDR={Y|N}] [,ECB=address of ECB] [,ALL={Y|N}] [,TASKID=taskid]

NAME

specifies the name of the resource.

Specified as: an RX address of an eight byte field; the full field is used by the system as a name. If VMADDR=Y is specified, NAME contains the address of the resource and is assumed to be only four bytes in length.

Default: none.

VMADDR
specifies whether or not NAME is a virtual memory address of a shared resource.

Specified as:

Y - NAME is the address of a four byte field
N - NAME is the address of an eight byte field

Default: N

ECB
specifies the address of the event control block to be posted with the successful or unsuccessful completion of the ENQ request.

Specified as: an RX address of a 16 byte field aligned on a fullword boundary.

Default: the ECB address is zero.

ALL
specifies that all ENQ requests for the specified task, or for this task if taskid is not specified, are to be posted as removed.

Specified as: the character Y

Default: N

TASKID
identifies the task whose resource access requests are to be removed.

Specified as: the RX address of a halfword containing the taskid.

Default: the task issuing the DEQ is assumed.

Programming notes: the taskid operand is only allowed for privileged modules. A privileged module may purge any ENQ request, but a nonprivileged module may purge only ENQs issued by a nonprivileged routine(s) within the task.

Return codes: the following codes are returned in register 15:

Code	Meaning
0	successful DEQ request
4	no ENQ request found to purge
8	parameter error on request

Note: for code 8 above, an error message prompt id will be in register 1.

DIR -- Delete Interrupt Routine (S)

The DIR macro instruction deletes control references to a previously specified interrupt control block. The interruption routine specified in the ICB cannot service interruptions unless the ICB is respecified by a SIR macro instruction.

Name	Operation	Operand
[symbol]	DIR	(icb address,...)[,MF={L|(E,list)}]

Note: A symbol in the name field is required with MF=L. If the MF operand is omitted, the standard form is assumed.

icb address
specifies the address of an interrupt control block established by a SPEC, SAEC, SIEC, SSEC, STEC, or SPEC macro instruction. This can be the symbol in the name field of these macro instructions. In the E-form of the macro instruction, this operand may refer to the same ICB list that is used by the SIR macro instruction.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, as register notation (2 through 12); in the E-form only, as an RX address.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: A DIR macro instruction deletes an active routine (one currently processing or interrupted) or prevents a routine from receiving subsequent interruptions through use of an E-form SPEC or SAEC macro instruction, using the NULL code for the INTTYP operand.

Return Data: On execution of DIR, the following conditions cause a return, with a return code in register 15 and the address of the invalid ICB in register 1.

Return Code	Condition
04	ICB contains invalid DCB (for input/output and asynchronous ICBs only) or an invalid time interval or clock number was specified (for timer).
08	No routine specified.
0C	The interruption servicing routine is active (no further interruptions will be presented to the interruption routine until it has completed its current servicing action).
10	Invalid parameter (an invalid length was specified or a nonprivileged user has attempted to DIR a privileged routine).

DISCSEG -- Disconnect Segment Group (O)

The DISCSEG macro instruction disconnects from an address space, a virtual storage segment group and assigns a unique eight character EBCDIC name to the disconnected segment group. The address space previously occupied by the disconnected segment group is marked unassigned.

| Note: this macro instruction has one or more operands that can be
| used only by a systems programmer; these operand(s) are defined and
| specified in the System Programmer's Guide manual.

L-form

Name	Operation	Operand
Symbol	DISCSEG	[DNAME=,LENGTH=,BOUND=,RNAME=,] MF=L

E-form

Name	Operation	Operand
[symbol]	DISCSEG	[DNAME=,LENGTH=,BOUND=,RNAME=,ADDRESS=,] MF=(E,LIST)

Standard-Form

Name	Operation	Operand
[symbol]	DISCSEG	[DNAME=,LENGTH=,BOUND=,RNAME=,ADDRESS=,]

Note: all operands are keyword.

DNAME=
Specifies the eight character EBCDIC name to be assigned to the disconnected segment group.

Specified as: Name enclosed within apostrophies; in E or standard form only, as the address of DNAME expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the disconnected segment group name.

Default: If this operand is omitted, the system will assign a unique eight character EBCDIC name to the disconnected segment group in the form of $$$XXXXX, where zero is less than or equal to XXXXX less than or equal to 99999.

CAUTION: Any user specified DNAME beginning with $$$ or SYS will be rejected by the system.

LENGTH=
specifies the number of contiguous virtual storage segments to be disconnected.

Specified as: An absolute expression; in the E or standard form only, the address of a halfword expressed as a relocatable expression, RX address or register notation. If register notation is used, the value must be given as a binary number placed in the low order two bytes of the register, right adjusted. If a relocatable expression or RX address is used, the address pointed to must be two bytes long, with the length right adjusted in the field.

Default: If this operand is omitted, the system will assign one of two possible default values. They are:

1. One virtual storage segment if RNAME is not specified.

2. Length of RNAME minus relative address offset.

BOUND=
specifies whether the disconnected segment group _must_ be reconnected at its disconnected address.

Specified as:
1. Y -- disconnected segment group must be reconnected at its disconnected address.
2. N -- disconnected segment group may be reconnected at any available segment aligned address.

Default: L and standard form -- N.
E-form -- value not changed in nameseq parameter list.

RNAME=
specifies the reserved segment group from which the segment group is to be disconnected.

Specified as: Name enclosed within apostrophies; in E or standard form only, as the address of RNAME expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the reserved segment group name.

Default: If this operand is omitted, the system will use the 'ADDRESS' specified.

ADDRESS=
specifies the segment aligned relative address from which the segment group is to be disconnected. If RNAME is specified, ADDRESS is the relative address offset from the beginning of RNAME. If RNAME is not specified, ADDRESS is the relative address offset from zero (i.e., an absolute address).

Specified as: In the E or standard form only, the address of ADDRESS is expressed as a relocatable expression, RX address, or register notation.

Default: Relative zero.

Return data: On return from execution of DISCSEG, all defaulted operands will be filled in with system assigned values. The address field in the nameseq parameter list will be set to an absolute address. Register 15 will contain a return code describing the success of the operation.

Return Codes

00	Successful
04	RNAME invalid
08	DNAME invalid
12	Segment not available to user class
16	Invalid address
20	Segment group overlap
24	Invalid length
28	Invalid bound option
32	Insufficient space available
36	User generated system reserved name
40	System error or system limit reached

Register 1 contains the address of the Nameseq Parameter List.

Note: The DSECT, CHANSG covers the Nameseq Parameter List.

Programming Notes: The return code in register 15 may be used to construct a branch table to handle the varying results from execution of the DISCSEG macro.

Upon execution of this macro, a set of input flags is constructed in the Nameseg Parameter List. They are:

X'80'	DNAME specified
X'40'	RNAME specified
X'20'	ADDRESS specified
X'10'	BOUND=Y
X'08'	LENGTH specified

Upon execution of DISCSEG, a set of output flags will be constructed with the above values.

<u>CAUTION</u>: If a disconnected segment group is reconnected and BOUND=Y was specified at DISCSEG time, the total area to which the disconnected segment group is to be attached, starting at its disconnected address, must be unassigned.

<u>L- and E-form Use Example</u>:

```
    DLIST     DISCSEG     MF=L
              DISCSEG     DNAME=D1,ADDRESS=DADD,MF=(E,DLIST)
                 .
                 .
                 .
    D1        DC          CL8'MYNAME'
    DADD      DC          F'0'
```

In the expansion of the L- form, a nameseg parameter list will be created in the following format:

```
          DLIST     +0   .0 A.B 6.    .    .
                     .
                     .
                     .
                    +24   .    .   .0 0.    .
                    +28   .    .    .    .    .
```

Upon successful execution of E-form:

```
          DLIST     +0   .0 A.B 6.    .    .
                     .
                     .
                     .
                    +12   . M . Y . D . N .   Where aa0000 is a segment
                    +16   . A . M . E . b .   aligned address previously
                    +20   .0 0.a a.0 0.0 0.   placed at 'DADD'
                    +24   .0 0.0 1.A 0.A 8.
                    +28   .    .    .    .    .
```

<u>DQDECB -- Remove Unchecked DECBs From a Data Set's DECB Queue (R)</u>

The DQDECB macro instruction (for BSAM) removes all unchecked data event control blocks (DECBs) from a queue of unchecked DECBs maintained by the system. If all of the DECBs within the queue have not been posted complete, the I/O requests associated with them are purged. DQDECB will not proceed until all DECBs have been posted complete either due to the purge or the fact that they have actually completed.

Name	Operation	Operand
[symbol]	DQDECB	decb address

decb address
: specifies the address of a data event control block (DECB) associated with the data set for which the DECB dequeueing will be performed. The DECB need not currently be in the DECB queue.

  <u>Specified as</u>: Register notation (1 through 12), or an RX address.

<u>Initialization</u>: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

<u>Programming Notes</u>: The DQDECB macro instruction is normally used in the SYNAD routine when multiple READ or WRITE macro instructions have been issued without an intervening CHECK. If DQDECB is issued, all unchecked READ or WRITE requests must be reissued. Unchecked I/O operations associated with the data set are removed from the system. If any of these DECBs are checked after the DQDECB without an intervening READ or WRITE, the CHECK will be treated as a NOP.

This facility is of use to users of the IMSK facilities of the DCB when they have multiple READ or WRITE requests unchecked and want to initiate their own error retry procedures, or to the user with multiple unchecked READ or WRITE requests who wants to reinitiate the sequence of I/O operations.

<u>Return Data</u>: Upon return from DQDECB, register 0 contains a count of the number of unchecked DECBs in the queue, and register 1 contains a pointer to the list of unchecked DECBs. This queue is read-only and is only valid until the next I/O operation is initiated on the data set.

Page of GC28-2004-6
Deleted 9 May 1979
By TNL GN20-3941

Page of GC28-2004-6
Deleted 9 May 1979
By TNL GN20-3941

EBCDTIME -- Convert System Time into EBCDIC Format (S)

The EBCDTIME macro instruction converts time from the format in which it is maintained by the system into various EBCDIC formats specified by the user. System time can be translated into any combination of years, months, days, hours, minutes, seconds, and tenths and hundredths of seconds by the EBCDTIME macro instruction.

Standard form:

Name	Operation	Operand
[symbol]	EBCDTIME	[{address of format map|'format map'}], [time] [,L=length]

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	EBCDTIME	['format map'],[time][,L=length],MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	EBCDTIME	[address of format map], [time][,L=length],MF=(E,list)

address of format map
specifies the location of the format map (see below).

Specified as: Register notation (0 or 2 through 12) or a relocatable expression. In the E-form, an RX address can also be used. Note that the format map can also be specified as a character string enclosed in apostrophes, as shown.

format map
specifies a character string, including the special character groups that are to be converted into the time and/or date: any characters in the map other than the special character groups are not converted.

Specified as: A string of up to 50 characters enclosed in apostrophes. The desired conversion format is requested by including one or more of the following groups of special characters in the string (the character groups can be specified in any order and can be separated by other characters):

CHARACTER GROUP	CONVERTED TO
YYYY	year, from 1900 to 1999
YY	year, from 00 to 99
DDD	day of year, from 001 to 366
MO	numeric month, from 01 to 12
DD	day of month, from 01 to 31
HH	hours, from 00 to 23
MM	minutes, from 00 to 59
SS	seconds, from 00 to 59
SSS	tenths of seconds, from 000 to 599
SSSS	hundredths of seconds, from 0000 to 5999
MON	first 3 characters of month
DAY	first 3 characters of day
DAYW	first 4 characters of day

Default: MO/DD/YY HH:MM

time
specifies the address of a doubleword binary number of microseconds to be converted to time and/or date, as directed by the user-specified format map. If the time is to be converted to a date, March 1, 1900, is used as the base for the conversion.

Specified as: In the standard and L-forms, a relocatable expression; in the standard and E-form, in register notation (0 or 2 through 12); in the E-form only, also as an RX address.

Default: The system-maintained time (that is, the binary number of microseconds that have elapsed since March 1, 1900) is converted as directed by the format map.

L

specifies a halfword containing the length of the format map (2 to

50 bytes). This operand need only be specified if the address of the format map is specified in the first operand (that is, not the map itself, in which case the system automatically calculates the length of the map). If less than two bytes are specified when this operand is required, the 14-byte default map is used. Normally, when the length is greater than 50, the format map is truncated on the right; however, if register notation is used for the length, and a length operand greater than 50 is specified, the system abnormally terminates the task.

Specified as: In all forms, a number from 2 to 50 inclusive. In the standard and E-form, register notation (0 or 2 through 12) can also be used.

Default: If the address of the format map is specified in the first operand and the length is defaulted, the 14-byte default map MO/DD/YY HH:MM is assumed and, in addition, a warning message is generated indicating that the length was not specified.

CAUTION: Only upper case characters will be processed as part of a special character group.

If the address of the format map is specified in the first operand, then the map should be reset after each use of EBCDTIME macro instruction because each execution of the macro would alter the map.

Programming Notes: The parameter list generated by the EBCDTIME macro instruction is:

```
┌───────────────────┐           ┌────────────────────────────────┐
│Register 1         ├──────────►│Address of a halfword containing│  Word 0
└───────────────────┘           │the length in bytes of the text │
                                │in which the format map has been│
                                │specified.                      │
                                ├────────────────────────────────┤
                                │Address of the area in which the│  Word 1
                                │user has designated a special   │
                                │character map and in which the  │
                                │converted time shall be placed at│
                                │completion of the EBCDTIME macro│
                                │execution.                      │
                                ├────────────────────────────────┤
                                │Address of a binary number to be│  Word 2
                                │converted to the time and/or    │
                                │date.  If defaulted to the system│
                                │maintained time, this field is  │
                                │set to zeros.                   │
                                └────────────────────────────────┘
```

The length in bytes of the format map is placed by the macro expansion in a halfword immediately following word 2 of the parameter list. Similarly, the format map is placed in a field immediately following the length field. If the user constructs his own parameter list, the bytes containing these parameters may be placed in other locations.

Return Data: After successful execution of the EBCDTIME macro instruction, the binary year (YY) is returned in bits 0-15 and the binary day of the year (DDD) in bits 16-31 of register 15.

If no translation is made by the EBCDTIME macro instruction, all bits of register 15 are set to 0.

<u>L- and E-Form Use</u>: In the E-form, the optional format map operand, if specified, usually points to an updated map that is to overlay the map defined by the L-form of the macro instruction; the updated map can be no longer than the original map. The L-form results in the generation of an in-line parameter list.

<u>Examples</u>: In EX1, the user has defined PRINT1 elsewhere in his program as:

```
   PRINT1  DC   CL21'THE DATE IS DD MON YY'
```

He issues the macro instruction:

```
   EX1   EBCDTIME   PRINT1,L=21
```

On output, on the given date, PRINT1 contains:

```
   THE DATE IS 24 FEB 71
```

In EX2, the user issues

```
   EX2   EBCDTIME 'THE DATE IS DD MON YY'
```

In this example, the format map is defined in the operand. Following execution, register 1 contains an address of a two-word field; the first word contains the address of a field containing the length of the format map and the second word contains the address of the map itself.

ENQ -- Enqueue on Resource Name (R)

The ENQ macro instruction is used to request exclusive or ahared read only access to a resource and to record the fact it has access to the resource.

Name	Operation	Operand
symbol	ENQ	NAME=name of resource [,ECB=address of ECB] [,ACCESS=type of access] [,WAIT=amount of time to wait] [,RESTYP=type of resource [,VMADDR={Y|N}]]

NAME
specifies the name of the resource.

<u>Specified as</u>: an RX address of an eight byte field; the full field is used by the system as a name. If VMADDR=Y is specified, NAME contains the address of the resource and is assumed to be only four bytes in length.

<u>Default</u>: none.

ECB
specifies the address of the event control block to be posted with the successful or unsuccessful completion of the ENQ request.

<u>Specified as</u>: an RX address of a 16 byte field aligned on a full-word boundary.

Default: the ECB address is zero, provided WAIT=IMMED; if WAIT is any other value, ECB cannot be defaulted.

ACCESS
specifies the type of access for the resource requested by the ENQ issuer.

Specified as:

RD - shared read only access; more than one user will be allowed access at any one time.

WR - exclusive access requested; only the 'requesting' user will be allowed access. A WR request must wait for all previous users to DEQ before the WR request is allowed.

Default: RD

WAIT
specifies the amount of time the requestor is willing to wait for the resource to become available for the requestor's exclusive use.

Specified as:

IMMED - return without waiting if the resource is unavailale. No ECB is required when this operand is specified. Upon return, 15 will be zero if the resource was available.

SHORT - wait for the amount of time specified as a short in the sysgen process.

MEDIUM - wait for the amount of time specified as a medium in the sysgen process.

LONG - wait for the amount of time specified as a long in the sysgen process.

INFINITE - wait until the resource is available, or until a DEQ is issued.

Default: INFINITE

RESTYP
identifies the controller of the resource.

Specified as:

SYSTEM - system owns and controls the resource.

USERCTL - user action controls the resource.

USER - user owns the resource.

Default: determined by DCLASS as follows:

SYSTEM for PRIVILEGED DCLASS
USER for USER DCLASS

VMADDR
specifies whether or not NAME is a virtual memory address of a shared resource.

Specified as:

Y - NAME is the address of a four byte field
N - NAME is the address of an eight byte field

Default: N

Programming notes: (1) Only modules with a DCLASS PRIVILEGED may specify RESTYP=SYSTEM. (2) The ECB address must be user read/write access when an ENQ is issued by a non-privileged module. (3) The issuer may choose to be called when the ECB has been posted by marking byte 1 of the ECB with an X'80' and placing the V and RCON of the entry point to be invoked in bytes 8 through 15 of the ECB. Upon posting the ECB, a QLE to the entry point will be queued. Upon entry, register 1 will point to the posted ECB.

The ENQ macro uses registers 0, 1, 14, and 15 to pass parameters to the supervisor. Upon return, these registers will have been altered to contain one of the following return codes in register 15 and possibly, an error prompt message id in register 1:

Code	Meaning
0	access gained to named resource; ECB has been posted
4	named resource is in use, request queued; ECB will be posted when available
8	parameter error in request; register 1 contains a message prompt id

The ECB will be marked as follows and should be tested by the ENQ issuer after issuing the ENQ request:

```
ECB  -  byte 0       - X'80'  wait for access
                       X'7F'  access granted
                       X'7E'  request purged by DEQ
                       X'41'  wait time expired

        byte 1       - X'80'  QLE to provided entry point

        bytes 2-3    - TWAIT SVC  to be issued by the waiter

        bytes 4-5    - X'0000'  reserved

        bytes 6-7    - taskid of task holding resource if
                       wait time has expired

        bytes 8-11   - VCON of entry point to receive control
                       when ECB is posted

        bytes 12-15  - RCON of entry point to receive control
                       when ECB is posted
```

ERASE -- Remove a Data Set from Direct Access Storage (S)

The ERASE macro instruction releases for other use the direct access storage assigned to a data set. In addition, it removes the entry for a cataloged data set from the catalog.

Standard form:

Name	Operation	Operand
[symbol]	ERASE	{address of data set name|'data set name'}

L-form:

Name	Operation	Operand
symbol	ERASE	'data set name',MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	ERASE	MF=(E,list)

address of data set name
specifies the location of the data set name (see below); at that location, the name of the data set must be followed by a X'27'.

Specified as: Register notation (2 through 12), or a relocatable expression.

data set name
specifies the name of any data set residing on direct access storage. (See "Data Set Name" in Part II, Section 1.) The data set name must be cataloged or must already be defined within the current task.

Specified as:

- The fully qualified name of a partitioned or nonpartitioned data set, a member or alias of a partitioned data set, or a partitioned or nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number).
- The partially qualified name of any type of data set, including a generation data group.

If the data set name does not involve a member name, the direct access storage occupied by that data set is erased (that is, released for other use). The name is removed from the catalog if the data set was cataloged.

If the data set name designates a particular member of a partitioned data set, the member's name is deleted from the partitioned organization directory (POD) of that data set. If an alias is specified instead of the member name, the member name is still deleted from the POD.

If the data set name is a partially qualified name or the name of a generation data group, all data sets (or generations) indexed under that name are erased and their catalog entries are removed.

If the name of a partitioned data set is supplied without a member name, the storage for the entire partitioned data set is released, and its name is removed from the catalog.

Specified as: The name of the data set, enclosed in apostrophes. (See "Data Set Name" in Part II, Section 1.)

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: The ERASE macro instruction cannot be used to erase data sets on magnetic tape; it applies to data sets on direct access storage only.

Programming Notes: If a shared data set is opened by several users concurrently, a particular user cannot erase that data set until every other sharer actively using that data set has issued a CLOSE macro instruction to deactivate his use of that data set. Any effort to erase an actively shared open data set will be ignored and a warning message will be issued. Once a user is the only currently active user of a shared data set he may erase that data set regardless of whether he has closed the data set, provided he has unlimited access to the data set (set by an operand of the PERMIT command).

Return Data: After execution of the ERASE macro instruction, a hexadecimal code will be returned in the fourth byte of general register 15:

Code	Significance
00	No error detected

04	Not class D or batch monitor entry
08	Invalid return code from system module called by ERASE
0C	Invalid delimiters in data set name
10	No data set name supplied
14	Invalid return code from CHEKDS module
18	Data set name not in catalog or TDT
1C	Partitioned data set not fully qualified name
20	Member of partitioned data set not found in POD
24	Data set not cataloged
28	Data set on public volume
2C	Data set is member of partitioned data set
30	User does not own data set in ERASE batch monitor entry
34	Sharing/access conflicts prevent processing
38	No catalog entry for ERASE batch monitor entry
3C	Data set name undefined (return code from DDEF)
44	Data set not on direct access storage
48	Volume not found
4C	Data set belongs to system - cannot be erased
58	Data set in use
5C	Resources exceeded,volume cannot be mounted

Examples: EX1 erases the data set A.B.C. EX2 erases all data sets cataloged under the partially qualified name A.B. EX3 erases the data set whose name is stored at location NAMLOC. EX4 removes member LAURA from the partitioned data set R.L.T. EX5 generates the parameter list for erasing data set M.P.S., and EX6 erases M.P.S.

```
EX1      ERASE     'A.B.C'
EX2      ERASE     'A.B'
EX3      ERASE     NAMLOC
EX4      ERASE     'R.L.T(LAURA)'
EX5      ERASE     'M.P.S',MF=L
EX6      ERASE     MF=(E,EX5)
```

ESETL -- Release Shared Data Set (R)

The ESETL macro instruction (for VISAM) releases a page-level READ interlock imposed by another macro instruction (for example, GET or READ). This macro instruction does not release the write interlock caused by a type KX READ. See the description of the RELEX macro instruction in this section.

Name	Operation	Operand
[symbol]	ESETL	dcb address

dcb address
specifies the address of the data control block opened for the data set being processed.

Specified as: Register notation (1 through 12), or an RX address.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Exceptional conditions resulting from the execution of a ESETL macro instruction cause control to be passed to the user's synchronous error exit (SYNAD) routine. In this case, the general registers and the exceptional condition fields in the data control block are set as shown in Appendixes B and F.

Programming Note: Rules for sharing VISAM data sets are given in Appendix K.

EXCSEG -- Exchange Segment Group (O)

The EXCSEG macro instruction performs the CONSEG and DISCSEG macro instructions in one operation.

Note: this macro instruction has one or more operands that can be used only by a systems programmer; these operand(s) are defined and specified in the System Programmer's Guide manual.

L-form

Name	Operation	Operand
Symbol	EXCSEG	[DNAME=,LENGTH=,BOUND=,RNAME=,] MF=L

E-form

Name	Operation	Operand
[symbol]	EXCSEG	[DNAME=,LENGTH=,BOUND=,RNAME=,ADDRESS=,] MF=(E,LIST)

Standard-Form

Name	Operation	Operand
[symbol]	EXCSEG	DNAME=[, LENGTH=,BOUND=,RNAME=,ADDRESS=,]

Note: all operands are keyword.

DNAME=
specifies the eight character EBCDIC name of an existing disconnected segment group. This name will be assigned to the segment group being disconnected.

Specified as: Name enclosed within apostrophies; in E or standard form only, as the address of DNAME expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the disconnected segment group name.

Default: none

CAUTION: Any user specified DNAME beginning with SYS will be rejected by the system.

LENGTH=
specifies the number of contiguous virtual storage segments to be disconnected.

Specified as: An absolute expression; in the E or standard form only, the address of a halfword expressed as a relocatable expression, RX address or register notation. If register notation is used, the value must be given as a binary number placed in the low order two bytes of the register, right adjusted. If a relocatable expression or RX address is used, the address pointed to must be two bytes long, with the length right adjusted in the field.

Default: If this operand is omitted, the system will assign one of two possible default values. They are:

1. One virtual storage segment if RNAME is not specified.

2. Length of RNAME minus relative address offset.

BOUND=
specifies whether the disconnected segment group must be reconnected at its disconnected address.

Specified as:
1. Y -- disconnected segment group must be reconnected at its disconnected address.
2. N -- disconnected segment group may be reconnected at any available segment aligned address.

Default: L and standard form -- N.
E-form -- value not changed in nameseg parameter list.

RNAME=
specifies the reserved segment group from which the segment groups are exchanged.

Specified as: Name enclosed within apostrophies; in E or standard form only, as the address of RNAME expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the reserved segment group name.

Default: If this operand is omitted, the system will use the 'ADDRESS' specified.

ADDRESS=
specifies the segment aligned relative address from which the segment group is to be disconnected. If RNAME is specified, ADDRESS is the relative address offset from the beginning of RNAME. If RNAME is not specified, ADDRESS is the relative address offset from zero (i.e., an absolute address).

Specified as: In the E or standard form only, the address of ADDRESS is expressed as a relocatable expression, RX address, or register notation.

Default: Relative zero.

Return data: On return from execution of EXCSEG, all defaulted operands will be filled in with system assigned values. The address field in the nameseg parameter list will be set to an absolute address. Register 15 will contain a return code describing the success of the operation.

Return Codes

00	Successful
04	RNAME invalid
08	DNAME invalid
12	Segment not available to user class
16	Invalid address
20	Segment group overlap
24	Invalid length
28	Invalid bound option
32	Insufficient space available
36	User generated system reserved name
40	System error or system limit reached

Register 1 contains the address of the Nameseg Parameter List.

| Note: The DSECT, CHANSG covers the Nameseg Parameter List.

| Programming Notes: The return code in register 15 may be used to construct a branch table to handle the varying results from execution of the EXCSEG macro.

| Upon execution of this macro, a set of input flags is constructed in the Nameseg Parameter List. They are:

X'80'	DNAME specified
X'40'	RNAME specified
X'20'	ADDRESS specified
X'10'	BOUND=Y
X'08'	LENGTH specified
X'04'	RELEAS=Y specified (for system programmers only)

| Upon execution of EXCSEG, a set of output flags will be constructed with the above values.

EXIT -- Normal Program End (R)

The EXIT macro instruction terminates program execution and switches the task to command mode. The words "EXIT, RELEASE ALL UNNEEDED DEVICES", followed by the message specified in the macro instruction are written on SYSOUT. If the NOMSG operand is specified, neither the system message nor the user-specified message is written.

Name	Operation	Operand
[symbol]	EXIT	[{address of message|'message text'}] [,NOMSG]

address or message
specifies the location in storage that contains the message to be issued. The first byte of the message must contain the length of the message (in bytes).

Specified as: Register notation (1 through 12), or an RX address.

message text
specifies the actual text of the optional message to be issued.

Specified as: The message itself, enclosed in apostrophes.

NOMSG
specifies that no messages are to be printed on SYSOUT when the exit is taken.

Specified as: NOMSG.

Default: The messages are printed.

Initialization: This macro instruction cannot be assembled in a privileged module unless the most recently issued DCLASS macro instruction in the assembly specified USER (see Appendix M), or the DCLASS option is USER by default.

Programming Notes: If EXIT is issued in a conversational task, the message is written on the user's terminal and the next command is taken from the terminal. If issued by a nonconversational task, the message is written on the SYSOUT data set and the next command is taken from the SYSIN data set.

The EXIT macro instruction returns control to the Command Analyzer.

Examples: In EX1, the user supplies the message text as a character string. In EX2, the message text is given at location MSGTEXT. In EX3 and EX4, no messages will be printed on SYSOUT.

```
    EX1      EXIT      'COMPLETED ARDUOUS'
    EX2      EXIT      MSGTEXT
    EX3      EXIT      ,NOMSG
    EX4      EXIT      'PRINT THIS ',NOMSG
```

FEOV -- Force End of Volume (R)

The FEOV macro instruction (for BSAM) directs TSS to advance to the next volume of a data set before the end of the current volume is reached. This macro instruction is applicable to BSAM data sets mounted on magnetic tape or on direct access devices.

Name	Operation	Operand
[symbol]	FEOV	dcb address

dcb address
: specifies the address of the data control block opened for the data set being processed.

| Specified as: Register notation (1 through 12), or an RX address.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: The following errors cause the results indicated:

Errors	Result
The dcb address operand specifies the address of a data control block that is not open.	No action
The dcb address operand specifies the address of an invalid data control block.	Task terminated
The data set is not being processed by BSAM (magnetic tape or direct access devices).	Task terminated
Not all BSAM READ or WRITE instructions on the data sets have been checked.	Task terminated

Example: In the following example, the control program is directed to advance to the next volume of the data set associated with the data control block REPORT.

```
     EX1      FEOV      REPORT
```

FIND -- Find a Member of a Partitioned Data Set (S)

The FIND macro instruction (for VPAM) searches a partitioned organization directory to locate a directory entry for a member and optionally places the user's data associated with the member into the specified area. The member is opened and positioned for processing.

Standard form:

Name	Operation	Operand
[symbol]	FIND	dcb address,name[,area,length]

L- and E-form:

Name	Operation	Operand
[symbol]	FIND	[dcb address],[name] [,area,length],MF={L|(E,list)}

Note: A symbol is required in the name field with the L-form. If either of the first two operands is omitted from the L-form, it must be supplied with the E-form.

dcb address
specifies the address of the data control block opened for the data set being processed.

Specified as: A relocatable expression; in the standard and E-forms, as register notation (2 through 12); in the E-form only, also as an RX address.

name
specifies the location of the eight-character member name, or alias, that is to be used to locate the member.

Specified as: Same as the first operand.

area
specifies the location of the user data area into which the user's data associated with the member is to be placed. If the area operand is specified, the length operand must also be specified.

Specified as: Same as the first operand.

length
specifies the number of bytes in the area provided for reading in the user data.

Specified as: In the standard and E-forms, as an absolute expression or in register notation (2 through 12); in the L-form, as an absolute expression only.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix E). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: The FIND macro instruction causes an abnormal termination if any conditions are discovered that make continuation impossible.

Programming Notes: If a DCB is opened with a DSORG of either VIP or VSP, only members with a matching DSORG are processed by FIND. If a mismatch is detected by FIND, a X'0C' code is returned to the user indicating the mismatch. (The user can still process mixed member VPAM data sets by specifying DSORG=VP in the DCB.)

If a DCB is opened with a DSORG of either VIP or VSP, only members with a matching DSORG are processed by FIND. If a mismatch is detected by FIND, a X'0C' code is returned to the user indicating the mismatch. (The user can still process mixed member VPAM data sets by specifying DSORG=VP in the DCB.)

If the length specified is less than the actual length of the user data in the POD, both the area and length operands are ignored and general register 15 contains an appropriate error code (X'10'). Rules for sharing VPAM data sets are given in Appendix K.

For shared VPAM data sets, the following interlocks are set by a FIND macro instruction:

1. VISAM members are:

   - write interlocked when opened for OUTPUT.

   - read interlocked when opened with any other option.

2. VSAM members are:

   - read interlocked when opened for INPUT.
   - write interlocked when opened with any other option.

Return Data: After execution of the FIND macro instruction, register 0 contains the length of the user data in the POD. Register 1 points to the parameter list shown below.

```
                              PARAMETER LIST
┌───────────────────┐        ┌──────────────────────────────────────┐
│ Register 1        ├────────┤Dcb address                           │Word 0
└───────────────────┘        ├──────────────────────────────────────┤
                             │Member name                           │Word 1
                             ├──────────────────────────────────────┤
                             │*Pointer to the user data area        │Word 2
                             ├──────────────────────────────────────┤
                             │*Pointer to the length, in bytes, of the │Word 3
                             │ user data area                       │
                             └──────────────────────────────────────┘
                              *These are zero if not supplied in the
                               macro instruction.
```

The length, in bytes, of the user data area is placed, by the macro expansion, in a word immediately following word 3 of the parameter list. However, if the user constructs his own parameter list, the word containing this length may be placed in some other location.

After execution of the FIND macro instruction, bits 24 through 31 of register 15 contain one of the following hexadecimal codes, indicating the status of the operation. The user should take appropriate action depending on the code returned.

Code	Definition
00	Successful completion of FIND.
04	Member or alias was not located by FIND.
08	The data control block indicated in the macro instruction is in use for creating a member. Execution of a STOW must be complete before this FIND can be executed.
0C	DSORG of member to be located does not match DSORG in DCB (this return code can only occur if the DSORG specified in the DCB is VIP or VSP).
10	The length specified in the macro instruction is not large enough to contain user data.
14	The member to be located is already open for this data control block, due to a previous FIND.

If the FIND macro instruction is used for a library search, the area operand must specify a length of 24 bytes. After execution of the macro instruction, the six words of the area contain:

Word 1	Relative page number of the program module dictionary (PMD)
Word 2	Length of the program module dictionary
Word 3	Relative page number of the text
Word 4	Length of the text
Word 5	Relative page number of the internal symbol dictionary (ISD)
word 6	Length of the internal symbol dictionary

FINDDS -- Locate JFCB Corresponding to Data Set Name (S)

The FINDDS macro instruction is used to obtain the location of the JFCB corresponding to a given data set name. If the data set name is not in the task data definition table (TDT), but is in the catalog, the user can request that a JFCB be created.

Standard form:

Name	Operation	Operand
[symbol]	FINDDS	data set name,byte,area

L-form:

Name	Operation	Operand
symbol	FINDDS	[data set name],[byte],[area],MF=L

Note: A symbol is required in the name field. Any operands omitted must be specified in the E-form.

E-form:

Name	Operation	Operand
[symbol]	FINDDS	[data set name],[byte],[area],MF=(E,list)

Note: If E-form operands are specified, they will overlay those specified in the L-form. The list operand must specify the symbol in the name field of the L-form; or the symbol (a relocatable expression) may be loaded into register 1 and the list operand specified as (1).

data set name
    specifies the address of a fully qualified data set name. The data set name located at the specified address must be padded on the right with blanks if less than thirty-five characters.

    Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

byte
    specifies the address of a byte that the user has set to zero if he wants a JFCB created for a cataloged data set, or to non-zero if he does not want a JFCB created.

    Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

area
    specifies the address of a word in which the pointer to the JFCB is to be placed.

    Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Return Data: A hexadecimal code is returned in register 15:

Code	Meaning
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	Data set name invalid; CHEKDS return code indicates dsname invalid form.

14	No JFCB found; DDEF return code indicates volume could not be mounted.

FINDJFCB -- Locate JFCB and Ensure Volume Mounting (S)

The FINDJFCB macro instruction is used to locate the JFCB for a given data definition name and, optionally, to ensure that the volumes specified in that JFCB are mounted.

Standard form:

Name	Operation	Operand
[symbol]	FINDJFCB	ddname,byte,area

L-form:

Name	Operation	Operand
symbol	FINDJFCB	[ddname],[byte],[area],MF=L

Note: A symbol is required in the name field. Any operands omitted must be specified in the E-form.

E-form:

Name	Operation	Operand
[symbol]	FINDJFCB	[ddname],[byte],[area],MF=(E,list)

Note: If E-form operands are specified, they will overlay those specified in the L-form. The list operand must specify the symbol in the name field of the L-form; or the symbol (a relocatable expression) may be loaded into register 1 and the list operand specified as (1).

ddname
: specifies the address of an 8-byte field containing the data definition name. If the data definition name in the field has fewer than 8 characters, it must be left-aligned with trailing blanks.

: Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

byte
: specifies the address of a 1-byte field containing a code indicating the processing action that is to be taken, whether the JFCB is found or cannot be found. The codes and their meanings are:

Code	Meaning
00	If JFCB is found, mount volumes, and return with appropriate data; if not found, issue diagnostics and an ABEND for the task.
01	If JFCB is found, mount volumes; whether JFCB is found or not, return to issuing program with appropriate data.
02	If JFCB is found, do not mount volumes; whether JFCB is found or not, return to the issuing program with appropriate return data.

: Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through

12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

area
specifies the address of a 4-byte field in which the address of the JFCB is to be placed.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Execution: The task data definition table (TDT) is searched for a JFCB with the specified name. If the JFCB is not found, the conversational user is asked whether he wants to define a data definition name. If he indicates yes, DDEF is called to build the JFCB. If he indicates no, or if the task is nonconversational, the action taken depends on the processing option code. When the JFCB is found, or created, a check is made to see if the proper volumes are mounted (unless the processing option of 2 was specified). When mounted, a pointer to the JFCB is set in the output area, and control is returned to the issuing program.

Return Data: The output area is set to zeros if the JFCB is not found (except for processing option 0) and to the address of the JFCB if it is found.

FREEBUF -- Return a Buffer to a Pool (R)

The FREEBUF macro instruction (for BSAM) returns a buffer (previously obtained by a GETBUF macro instruction) to a buffer pool, so that it will be freed and can be obtained again by GETBUF. It is not necessary to free all buffers prior to issuing the CLOSE macro instruction.

Name	Operation	Operand
[symbol]	FREEBUF	dcb address,buffer address

dcb address
specifies the address of the data control block opened for the data set being processed.

Specified as: Register notation (1 through 12), or an RX address.

buffer address
specifies the register that contains the address of the buffer being returned to the pool.

Specified as: An absolute expression, 2 through 12.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Error conditions that result in termination of the task are:

1. An invalid data control block is specified.

2. The buffer pool address is not in the data control block (GETBUF was not invoked before FREEBUF).

3. The buffer address specified by the user does not belong to the buffer pool.

4. The buffer specified by the user is not in use (GETBUF was not used to obtain the buffer).

Programming Notes: To release a buffer by FREEBUF, a buffer pool must have been assigned to the data control block, and the specified buffer must have been obtained by the GETBUF macro instruction.

Example: See the example in the GETBUF macro instruction description.

FREEMAIN -- Release Allocated Virtual Storage (R)

The FREEMAIN macro instruction releases a virtual storage area previously allocated by a GETMAIN macro instruction. This virtual storage area can be released by units of pages or 8-byte multiples.

Name	Operation	Operand
[symbol]	FREEMAIN	{PAGE [,VAR]|R},LV=length,A=address

PAGE

specifies that a number of pages of virtual storage are to be released.

Specified as: PAGE

VAR

specifies the release of an area of virtual storage obtained through a PAGE,VAR GETMAIN macro instruction. This operand is only specified if PAGE is specified.

Specified as: VAR

R

specifies that a number of bytes of virtual storage is to be released (LV must specify a multiple of 8 bytes).

Specified as: R

LV

specifies the length, in pages or in bytes (as specified by PAGE or R), of the virtual storage area to be released. The LV= operand must be written as in the corresponding GETMAIN macro instruction.

Specified as: An absolute expression, or register notation (0 or 2 through 12). If register notation is used, the length must be given as a binary number placed in the low-order three bytes of the register specified, right adjusted. The high-order byte of the register must be 0.

A

specifies the address of a fullword containing the address of the virtual storage area to be released.

| Specified as: Register notation (1 through 12), or an RX address.

If register notation is used, the address of the virtual storage area (not the address of a fullword containing the virtual storage area address) must be loaded into the register before execution of this macro instruction. If bytes are specified, the address of the virtual storage area must be on a doubleword boundary (or an error code of X'08' is returned in register 15).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, except when R is specified as the first operand, the address of a save area must be placed in register 13 before this macro instruction is executed.

During execution of the FREEMAIN macro instruction, the task issuing the FREEMAIN macro instruction is abnormally terminated if:

- the area to be released is privileged or contains privileged areas, or
- the area to be released was not allocated by a GETMAIN.

Return Data: If FREEMAIN is unable to locate the page or doubleword boundary containing the virtual storage to be released, or if any of the virtual storage has never been assigned or has already been released, a return code of X'04' is placed in register 15. If a doubleword boundary is not specified, X'08' is returned in register 15.

Examples:

EX1 requests the release of a 16-page virtual storage area whose address is in register 1. EX2 requests the release of an area whose address is in the fullword at ADD1 and whose length, in pages, is in register 0. EX3 requests the release of an area whose length is two pages more than the value specified during system generation (see the description of the GETMAIN macro instruction), and whose address is in the fullword at ADD2. EX4 requests the release of 200 bytes of virtual storage whose address is in the fullword at ADD3.

```
EX1      FREEMAIN  PAGE,LV=16,A=(1)
EX2      FREEMAIN  PAGE,LV=(0),A=ADD1
EX3      FREEMAIN  PAGE,VAR,LV=2,A=ADD2
EX4      FREEMAIN  R,LV=200,A=ADD3
```

FREEPOOL -- Free a Buffer Pool (R)

The FREEPOOL macro instruction (for BSAM) releases an area that had previously been assigned as a buffer pool to a specified data control block. The area must have been acquired through either the execution of a GETPOOL macro instruction or by the buffer option described in the DCB macro instruction; that is, when the DCB macro instruction was written, BUFNO= and BUFL= were included. FREEPOOL need not be issued if a CLOSE macro instruction is issued for the data control block to which the buffer pool is assigned.

Name	Operation	Operand
[symbol]	FREEPOOL	dcb address

dcb address
specifies the address of the data control block to which the buffer pool was assigned.

| Specified as: Register notation (1 through 12), or an RX address.

CAUTION: If the dcb address operand does not specify the address of a valid data control block, the task is terminated.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix F). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: If the associated data set is processed by means of BSAM, FREEPOOL may be issued as soon as the buffers are no longer required.

Examples: EX1 releases the buffer area assigned to the data control block whose address is OUTPUT. EX2 releases the buffer area assigned to the data control block whose address is in register 1.

```
EX1      FREEPOOL  OUTPUT
EX2      FREEPOOL  (1)
```

See also the example in the GETBUF macro instruction description.

GATRD -- Get Record from SYSIN (S)

The GATRD macro instruction reads a record from the user's SYSIN and places it in a specified area.

Standard form:

Name	Operation	Operand
[symbol]	GATRD	input area,length[,SIC]

L- and E-form:

Name	Operation	Operand
[symbol]	GATRD	[input area][,length][,SIC],MF={L|(E,list)}

Note: A symbol is required in the name field with MF=L. Either of the first two operands that is omitted from the L-form must be supplied with the E-form. The operands specified with the E-form will overlay those specified with the L-form. If the MF operand is omitted, the standard form is assumed.

input area
specifies the address of the area into which the input record is to

be placed. The user must define the length of this area by the length operand.

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, register notation (2 through 12); | in the E-form only, an RX address.

length
specifies the address of a fullword containing the length of the expected input record; the maximum length of the line depends upon the input source:

VAM data set	128 characters
1050	130 characters
2741	130 characters
Model 33 and 35 KSR[1]	72 characters

Note: This operand must be specified in either the L-form or the E-form of the macro instruction. On return, the actual record length is stored at the same address.

Specified as: Register notation (2 through 12), or a relocatable expression.

SIC
indicates whether characters within SYSIN representing control functions (specified as such in the Character Translation Table, CTT) are to be regarded as input characters by the GATRD macro instruction. If SIC is specified, all characters in the CTT (located in the task profile) are translated to internal code and transmitted to storage regardless of the functional code assigned to it in the CTT.

Specified as: SIC

Default: Omission of this operand requests the standard mode in which only characters assigned the translation code (00) in the CTT are translated and transferred to storage, while characters assigned to other functional codes are not transferred to storage. Thus, in the standard mode, characters within a line of SYSIN that are assigned unique functional codes in the Character Translation Table, such as the backspace or cancel control functions, are not read into storage as part of the SYSIN input line.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Records whose length is no longer than one line (from the terminal keyboard), or one card image (from the terminal card reader) are read by one GATRD macro instruction. Records longer than one line or card image are truncated if continuation of the record is not indicated | (See the description of the CONT operand of the MCAST macro

----

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

instruction.) If continuation is indicated, succeeding GATRD macro instructions can be used to read the remainder of the record.

Programming Note: GATRD cannot be used to recover a record that was read by an earlier GATRD.

If the SIC operand is specified, both input characters and control characters must be included in the length count specified via the length operand of the GATRD macro instruction.

In the standard mode, a character is transmitted to the message area only if it satisfies the following four conditions:

1. It is assigned the translation code (00) in the Character Translation Table (see Command System User's Guide for additional information pertaining to the CTT).

2. It is not deleted by the action of any characters that are assigned the backspace or cancel functions. The user is cautioned that there may be extraneous characters beyond the text returned in the input area. This arises from the fact that the backspace function on the end of the line causes the message length to be adjusted only so that terminal characters removed by a backspace may show in the unused portion the input area.

3. It appears on a record to the left of all characters assigned the end-of-message function.

4. Space is available for it in the area specified by the input area operand.

If a GATRD is executed in a loop and the user wishes to have the same value for the expected record length each time, he must reinitialize the length in field each time GATRD is to be executed.

Note: Only that portion of the record from the pointer on is available to the user. See Appendix I of Assembler Programmer's Guide for more information on record formats.

Return Data: On return from GATRD, register 15 contains two bytes of coded information (hexadecimal) in bits 16-31, as shown in Figure 10.

Example: A 120-character record (that is, 120 characters assigned the translation function within the CTT) is to be fetched from SYSIN and placed in the area READIN:

```
EX1      GATRD     READIN,ILENGTH
```

In this example, the user has defined the length elsewhere in the program:

```
ILENGTH   DC       F'120'
```

Note that the absence of the SIC parameter defaults to the standard mode, in which only those characters assigned the 00 translation code in the CTT are to be translated by the standard Character Translation Table and transmitted to the input area.

Bits 16-23 Code	Significance
0	Input record contains no continuation code; record is therefore complete.
1	Input record contains a continuation code. Issue a GATRD to get next portion of record.
2	Record was truncated because it exceeded maximum length specified by the user.
Bits 24-31 Code	
0	SYSIN is non-conversational
8	Attention interruption occurred; record, if any, is unpredictable.
10	SYSIN is from terminal keyboard.

Figure 10. Return codes from read-only and write-with-read GATE macro instructions

GATWR -- Write Record on SYSOUT (S)

The GATWR macro instruction writes a message on the user's SYSOUT from an area in storage.

Standard form:

Name	Operation	Operand
[symbol]	GATWR	message,length[,SIC]

L- and E-form:

Name	Operation	Operand
[symbol]	GATWR	[message],[length][,SIC][,MF={L|(E,list)}]

Note: A symbol is required in the name field with MF=L. Either of the first two operands that is omitted from the L-form must be supplied with the E-form. The operands specified with the E-form will overlay those specified in the L-form. If the MF operand is omitted, the standard form is assumed.

message
specifies the address of the area containing the message text. The message may include any characters that can be represented in the terminal character set, including blanks, parentheses, and commas.

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, register notation (2 through 12); in the E-form only, an RX address.

length
specifies the address of a fullword that contains the length of

the message to be issued. If the message is longer than the maximum line length of SYSOUT, GATWR will write as many lines (or records) as are necessary up to a maximum message length of 512 bytes.

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, register notation (2 through 12); in the E-form only, an RX address.

SIC

specifies how the characters of the message are to be handled before transmission to the user's SYSOUT. The procedure followed when SIC is not specified is described in the Programming Notes. If SIC is specified, characters with 00 function codes are transmitted to the user's SYSOUT without translation. Control characters are handled as when SIC is not specified (the control function is performed, and the characters are not translated or transmitted to SYSOUT).

Specified as: SIC

Default: Message characters are translated and control functions are performed as described in the Programming Notes.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix F). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

If SIC is not specified, handling of the message characters is as follows:

The Output Character Translation Table consists of two 256-byte sections; the first section contains the translation table to be used for translating the characters of the message, and the second section contains function codes that control the handling of the corresponding message characters. For 00 function codes, the message characters are translated as determined by the first section of the table and transmitted to the user's SYSOUT. Nonzero function codes indicate a control function, such as backspace or cancel; when a character with a nonzero code is encountered, the indicated function is performed, but the character is neither translated nor transmitted to the user's SYSOUT.

The Output Character Translation Table is defined by the system. However, the user may create and use his own translation table; see Command System User's Guide and the OCTT operand of the MCAST macro instruction for more information.

Return Data: On return from GATWR, the low-order byte of register 15 contains the return code shown in Figure 11.

Bits 24-31 Code	Significance
06	Attention interruption occurred; record, if any, is unpredictable.
10	SYSOUT is the terminal keyboard.

Figure 11. Return codes from GATWR and GTWPC macro instructions

Example: A 16-character record is to be written on SYSOUT:

```
EX1      GATWR      RECOUT,LENGTH
```

In this example, the user has coded elsewhere in the program:

```
RFCOUT   DC        C'COMPLETED ROUND1'
LENGTH   DC        F'16'
```

GDV -- Get Default Value (S)

The GDV macro instruction searches the profile member of the user library associated with the current user (that is, a private library assigned to each user when he joins the system) to find any predefined parameter default values.

All forms:

Name	Operation	Operand
[symbol]	GDV	[{parameter address|'parameter'}] [,MF={L|(E,list)}]

Note: If the MF operand is omitted, the standard form is assumed. A symbol in the name field is required in the L-form. If the first operand is omitted in the L-form, it must be supplied with the E-form.

parameter address
: specifies the address of a particular parameter. The actual parameter must be preceded in storage by its length (one byte) .

  Specified as: An RX address, or register notation (1 through 12).

parameter
: specifies the parameter.

  Specified as: The parameter itself, enclosed in apostrophes.

  Default: It is assumed that the issuer has placed the parameter address in register 1.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming notes: the GDV macro instruction is useful within user-coded routines for locating any default values in the user profile.

The parameter specified by the GDV operand must be the same as the parameter indicated in a DEFAULT command.

GDV is most useful in a routine not associated with any particular command -- for which there is no reasonable way to provide a parameter value via a command entry and several levels of call, or for which it is not reasonable to associate a parameter with a command, or for which it is not reasonable to associate the same parameter with a number of commands which all use the service routine. (See the BPKDS macro instruction.)

Return Data: If there is no predefined default value in the user library corresponding to the parameter name indicated in the GDV macro instruction, register 1 is set to zero and control is returned to the command expansion routine.

If the GDV routine finds a default value in the user library, the virtual storage address of the default value is placed in register 1 and control is returned to the command processing routine (the byte preceding the default value contains the length of the default value).

L- And E-Form Use: The parameters specified in the E-form of this macro instruction will overlay those specified in the L-form. The E-form may not specify more operands than are specified in the corresponding L-form. For example:

```
SUE         GDV    'DPAR1',MF=L
            GDV    INSTEAD,MF=(E,SUE)
             .
             .
             .
INSTEAD     DC     C'DPAR2'
```

When the E-form of this macro instruction is executed, the parameter specified in the L-form (DPAR1) will be replaced by the parameter specified in the E-form (DPAR2).

Example: If a user has created a command to be issued at the terminal, by use of the BPKDS macro instruction and the BUILTIN command, the command processing routine coded by the user might employ the GDV macro instruction as described below.

Terminal Commands	User-Coded Command Processing Routine
.	CSECTA CSECT
.	.
BUILTIN TROT,BPKLABEL	LABELA EQU *
.	GDV 'KEYWORD2'
.	.
DEFAULT KEYWORD2=200	PSECTA PSECT
.	.
.	BPKDS LABELA,KEYWORD1,KEYWORD2
.	.
.	.
.	.
TROT 50	

The command TROT is created by the user, and is issued with a defaulted second parameter. In such cases, the command processing routine then executes the GDV macro instruction to search the user library for any defaulted value that may have been previously specified. When the default value is located, the command processing routine can then insert the appropriate data into the parameter list generated by the BPKDS macro instruction and continue processing. In the example above, the defaulted KEYWORD2 operand is found equal to 200.

GET -- Get a Record (R)

The GET macro instruction (for VSAM, VISAM, and QSAM) can be specified in either locate mode or move mode. In locate mode, the GET macro instruction locates the next sequential record of an input data set and places its address in register 1. The user may then operate on the record where it is, or move it to a work area. In move mode, the GET macro instruction acquires the next sequential record and moves it from an input buffer to a user-specified area in virtual storage.

Name	Operation	Operand
[symbol]	GET	dcb address[,area]

dcb address
specifies the address of the data control block opened for the data set being processed.

| <u>Specified as</u>: Register notation (2 through 12), or an RX address.

area (for move mode only)
specifies the address of the user's work area into which the record is moved. The absence of this operand indicates a locate-mode GET.

| <u>Specified as</u>: Register notation (0 or 2 through 12), or an RX address.

<u>Initialization</u>: The address of a save area must be placed in register 13 before execution of this macro instruction.

<u>CAUTIONS</u>:

<u>For VSAM</u>: The contents of register 1 are not guaranteed at the conclusion of a move mode GET.

<u>For VISAM</u>: Any exceptional condition (that is, logical record out of sequence) resulting from the execution of a GET macro instruction causes control to be passed to the user's synchronous error exit (SYNAD) routine. In this case, the general registers and the exceptional condition fields in the data control block are set as shown in Appendixes B and F.

The buffer address of the record will remain in register 1 on return from a move mode GET.

<u>For QSAM</u>: If any of the following error conditions exists as a result of the execution of the GET macro instruction, control will be passed to the synchronous error exit (SYNAD) routine specified in the data control block:

1. The next record to be processed starts a block that could not be read satisfactorily because of an error condition.

2. A preceding PUTX macro instruction could not be executed without resulting in an error condition. This situation is discovered by the GET macro instruction when working in update mode.

3. When processing variable-length records, the length of a block does not equal the actual block size.

4. When processing variable-length records, the lengths of each individual record within a variable-length block do not add up to the length indication of the block.

When the SYNAD routine is given control, the general registers and status indicators are set as shown in Appendix B.

<u>Programming Notes</u>: If a GET is requested beyond the end of a data set as a result of a sequential operation or SETL macro instruction, the user EODAD exit is taken. See Appendix C.

The GET macro normally retrieves the record following the record at which the data set is currently positioned. However, a GET macro instruction preceded by a SETL retrieves the record to which the data set is positioned by the SETL.

For VSAM: For undefined-format records, the user must set the DCBLRE field to the length of the record to be retrieved before issuing GET. Rules for sharing VSAM data sets are given in Appendix K. When retrieving variable-length records, the GET macro instruction returns with the length of the logical record in the DCBLRE field of the data control block.

For VISAM: A page-level read interlock is imposed on the page referred to by execution of this macro instruction. The interlock is released by any macro instruction referring to the same DCB that refers to another page. Rules for sharing VISAM data sets are given in Appendix K. When retrieving variable-length records, the GET macro instruction returns with the record length in the DCBLRE field of the data control block.

| For QSAM: In locate mode, the control program returns the address of the next logical record in register 1, and places the record length in the logical record length (DCBLRECL) field of the data control block. In the move mode, the area address provided by the user is returned in register 1 and the logical record length of the accessed record is placed in DCBLRECL. Because QSAM does not support the substitute-mode GET, this feature (that is, return of the area address) provides compatibility that allows TSS to use the move mode in order to execute programs originally written to use the substitute-mode GET.

Examples: In example 1, move mode, the next record from the data set associated with the DCB labeled STAT is moved to the work area labeled SAMPLES. The address of the work area is returned to the user in register 1.

```
EX1    GET     STAT,SAMPLES
        .
        .
STAT   DCB     DSORG=......
SAMPLES DS     20F
        .
        .
        .
```

Before execution of the locate mode GET in example 2, register 1 is loaded with the address of the data control block. After execution of the GET, register 1 contains the address of the next sequential record, which the user then can move into a work area.

```
EX2       LA    1,DCBADR
          GET   (1)
           .
           .
DCBADR    DCB   DSORG=.....
```

GETBUF -- Get a Buffer From a Pool (R)

The GETBUF macro instruction (for BSAM) obtains a buffer from a specified buffer pool. Buffers acquired by a GETBUF must be returned by a FREEBUF before they can be obtained again. However, it is not necessary to free all buffers prior to issuing the CLOSE macro instruction.

Name	Operation	Operand
[symbol]	GETBUF	dcb address,register

dcb address
specifies the address of the data control block opened for the data set being processed.

| <u>Specified as</u>: Register notation (1 through 12), or an RX address.

register
specifies a register into which the control program is to place the address of the buffer.

<u>Specified as</u>: An absolute expression.

<u>CAUTION</u>: The following error conditions result in termination of the task:

1. An invalid data control block is specified.

2. The buffer size is 0 or greater than 32,760.

3. The number of buffers in the pool is 0 or greater than 255.

4. The data control block is not open.

<u>Initialization</u>: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

<u>Programming Notes</u>: A buffer pool must have been assigned to the data control block by use of a GETPOOL or by the buffer option in the DCB macro instruction (that is, BUFL= and BUFNO= are supplied in the DCB macro instruction). Each successive GETBUF macro instruction issued obtains a buffer in the order in which it exists in the buffer pool. For example, if a buffer pool contains five buffers, five successive GETBUF macro instructions would obtain five successive buffers from the buffer pool.

Buffers must be returned to the pool by the FREEBUF macro instruction before they can be obtained again.

<u>Return Data</u>: If no buffer is available within the pool, the contents of the register specified in the GETBUF macro instruction will be set to zero rather than to an address.

The address of the buffer pool is placed in the DCBBCN field of the data control block.

<u>Example</u>: The GETPOOL macro instruction is used to define a buffer pool of 10 buffers of 100 bytes each. The GETBUF macro instruction is used to obtain the address of an available buffer in register 5. That buffer is then used to hold an input block when a data set is being read. (The length operand is not required in the READ macro instruction). The buffer is released by the use of the FREEBUF macro instruction; the buffer pool is eventually released by a FREEPOOL macro instruction.

```
         GETPOOL       INDCB,10,100
            .
            .
            .
         OPEN          (INDCB,(INPUT))
            .
            .
            .
         GETBUF        INDCB,(5)
            .
            .
            .
         READ          DECB1,SF,INDCB,(5)
            .
            .
            .
         FREEBUF       INDCB,(5)
            .
            .
            .
         FREEPOOL      INDCB
            .
            .
            .
            .
| INDCB DCB            DSORG=PS,...
```

GETDV -- Get Dictionary Value (O)

The GETDV macro instruction locates a specified symbol in the TSS Dictionary and returns a pointer to that value in register 1.

Note: The GETDV macro plus the SETDV macro give the programmer the capability of creating, updating and deleting entries in the TSS Dictionary.

Name	Operation	Operand
	GETDV	name,TYPE={DEF|SYN|CSW},CONV={Y|N},MF=

name
address of the symbol preceded by a one-byte length field or the symbol name in character format.

Specified as: a one-to-eight character alphameric name; the first character must be alphabetic.

TYPE=
the type of entry to search for.

Specified as: one of the following types:

DEF - default

SYN - synonym

CSW - command symbol word

Default: DEF

CONV=
specifies whether or not the command symbol word is to be converted to printable EBCDIC.

Specified as: Y (yes) or N (no)

Default: N

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Return codes: register 0 contains the type code of the dictionary entry, and register 1 contains the address of the dictionary value, preceded by a one-byte length. The valid return codes in register 15 are as follows:

Code	Meaning
X'00'	successful locate for the symbol
X'04'	symbol not found
X'08'	incorrect or invalid name given
X'0C'	invalid type given

Example 1:

```
GETDV 'SYSIN'
```

Example 2:

```
              LA    R3,SYMBOL
             GETDV   (R3),TYPE='DEF'
                .
                .
                .
                DC   AL1(L'SYMBOL)
       SYMBOL   DC   C'SYSOUT'
```

GETMAIN -- Get Virtual Storage (R)

The GETMAIN macro instruction requests a contiguous area of virtual storage for a user's task during program execution. The areas of virtual storage allocated by GETMAIN contain binary zeros.

Name	Operation	Operand
[symbol]	GETMAIN	{PAGE[,VAR]|R}, LV={length[,PR=class][,PACK=mode]|(15)} [,EXIT=RETURN] [,RNAME=]

PAGE
: specifies that a number of pages of virtual storage is to be allocated.

Specified as: PAGE

VAR
: specifies that an additional number of pages is to be allocated. This number will have been defined by the installation during system generation. These additional pages are added to those specified in the LV operand. If PAGE,VAR is specified and LV is specified as zero, the system generated number of pages (in field ISAVAR) is requested.

Specified as: VAR

R

specifies that a number of bytes of virtual storage is to be allocated. RNAME cannot be specified.

Specified as: R

LV=

specifies the desired number of bytes, pages, or additional pages of virtual storage, depending on whether R, PAGE, or PAGE, VAR is specified.

Specified as: An absolute expression or register notation. For PAGE requests using register notation format, only register 15 is allowed. For R requests using register notation, 0 or 2 through 12 can be used with the value given as a binary number placed in the low order three bytes of the register, right adjusted, and the high order byte of register 0 containing binary zeroes in bits 0-3, and the protection class in binary in bits 4-7 (see the PP operand). If a request is made for a number of bytes that is not a multiple of 8, the next higher multiple of 8 is allotted.

The length of the specified virtual storage request may not exceed the amount of virtual storage available at execution time. Refer to the EXIT and RNAME operands.

Note: See initialization section of this macro for PAGE requests. No recoding or reassembly is required for programs using the TSS pre-Release 2.0 register notation format of this macro instruction.

PR=

specifies the protection class to be assigned to the requested virtual storage.

Specified as:
0 - User read-and-write
1 - User read-only
2 - Private privileged

This parameter has meaning only for privileged users. If bytes were specified and an invalid protection class is specified, a return code of X'08' is placed in register 15.

Default: 0

PACK=

specifies that the requested virtual storage is to be put into a unique segment or packed into the first available space.

Specified as:
0 - put into a unique segment, or pack into the first available space, depending on system parameters and the type of request
1 - pack into the first available space, regardless of any system parameters or the type of request
2 - put into a unique segment regardless of any system parameters or the type of request

For the PACK parameter specified as 0 or 2, multiples of 16-page requests will be on a 16-page boundary, and multiples of 256-page requests will be on a 256-page boundary.

Default: 0

EXIT=RETURN

specifies that, if the request for virtual storage cannot be satisfied, a return code of X'04' is placed in register 15.

<u>Specified as</u>: EXIT=RETURN

RNAME=

specifies that the requested virtual storage is to be obtained from the reserved segment group. Reserved segment group names are eight-character EBCDIC names (refer to the RSVSEG macro instruction).

Specified as: a reserved segment group enclosed within apostrophies; as the address of the reserved segment group name expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the reserved segment group name.

Default: If this operand is omitted, the system obtains requested virtual storage from any non-reserved segment group.

Note: If RNAME is specified, the system will not allocate requested virtual storage outside the range of the reserved segment group. Refer to EXIT operand.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must specify the privilege class of the module which will release (FREEMAIN) the area obtained.

This macro generates the ADDPG macro in line and uses registers 0, 1, and 15 as parameter registers to the ADDPG macro for PAGE requests.

The format of these registers is:

Register 0 and 1 contain the reserved segment group name.

Register 15 contains flags and length value as:

Byte 0	X'01'	Variable request
	X'02'	System requested function
	X'04'	EXIT=RETURN
	X'40'	RNAME specified
Byte 1	Bits 0,1	Not used
	Bits 2,3	Packing parameter
	Bits 4-7	Protection class
Bytes 2 and 3		Number of pages requested

CAUTION: If a request for virtual storage cannot be satisfied, and the EXIT operand is omitted, an abnormal task termination occurs.

Programming Notes: Two sequential GETMAIN macro instructions do not guarantee the allocation of two contiguous areas. The only way to ensure a contiguous allocation of n pages is by issuing a GETMAIN macro instruction specifying an area whose length is n.

No recoding or reassembly is required for modules using the TSS pre-Release 2.0 format of this macro instruction.

Return Data: The address of the allocated virtual storage is returned in register 1. The area begins on a page boundary if pages of virtual storage were requested, and on a doubleword boundary if bytes were requested.

If the GETMAIN macro instruction is executed successfully, a return code of X'00' is placed in register 15; if the request is unsuccessful, the return code is X'04'.

Example:

EX1 specifies a request for pages and indicates that register 15 has been loaded with the number of pages of virtual storage requested (and with zeroes in the high-order 2 bytes of the register). EX1 also specifies that a return code of X'04' be issued if the request cannot be satisfied. EX2 requests allocation of 6 pages of virtual storage. EX3 indicates a request for pages. Before execution of this macro instruction, the user loads register 15 with the length of the required area and loads zeroes in the high-order two bytes of the register. If the virtual storage cannot be allocated, the task is abnormally terminated. EX4 requests two more pages than the number specified during system generation. EX5 specifies 50 bytes of virtual storage to be allocated; the system will assign 56 bytes. EX6 specifies a request for pages and indicates that register 15 has been loaded with the number of pages and that zeroes are in the two high-order bytes of the register. The virtual storage will be obtained from the reserved segment group whose name has been loaded into registers 2 and 3. EX7 specifies a request for 4 pages to be obtained from the reserved segment group 'MYRNAME'.

```
     EX1 GETMAIN PAGE,LV=(15),EXIT=RETURN
     EX2 GETMAIN PAGE,LV=6
     EX3 GETMAIN PAGE,LV=(15)
     EX4 GETMAIN PAGE,VAR,LV=2
     EX5 GETMAIN R,LV=50
     EX6 GETMAIN PAGE,LV=(15),PNAME=(2)
|    EX7 GETMAIN PAGE,LV=4,PNAME=RNAME1
                 .
                 .
                 .
   RNAME1    DC     CL8'MYRNAME'
```

GETPOOL -- Get a Buffer Pool (R)

The GETPOOL macro instruction (for BSAM) requests allocation of an area of virtual storage for use as a buffer pool. The buffer pool is assigned to the specified data control block.

Name	Operation	Operand
[symbol]	GETPOOL	dcb address,number,length

dcb address
  specifies the address of the data control block to which the buffer pool is to be assigned.

| <u>Specified as</u>: Register notation (1 through 12), or an RX address.

number
  specifies the number of buffers to be in the pool.

  <u>Specified as</u>: Register notation (2 through 12), or an absolute expression. The maximum that may be specified is 255. See the length operand for the use of (0).

length
  specifies the number of bytes in each buffer. The value is increased, if necessary, by the GETPOOL routine to be a doubleword multiple.

Specified as: Register notation (2 through 12), or an absolute expression. The maximum that may be specified is 32,760 bytes. (0) may also be used to specify both the number and length operands, in which case the number of buffers must be in the two high-order bytes of register 0, and the length of each buffer must be in the two low-order bytes of register 0, prior to execution of the macro instruction.

CAUTION: Failure to observe the following restrictions results in termination of the task:

1. Only one buffer pool may be assigned to a data control block at one time.
2. The buffer length must be less than or equal to 32,760.
3. The number of buffers must be less than or equal to 255.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: If the GETPOOL macro instruction is used, it must be executed prior to the execution of any GETBUF macro instruction that refers to the buffer pool allocated by GETPOOL.

The FREEPOOL macro instruction should be issued to return the allocated buffer pool to the system, unless a CLOSE is issued for the data control block to which the buffer pool is assigned.

Examples: EX1 constructs a buffer pool consisting of two buffers, each 136 bytes long, in an area of virtual storage. This buffer pool is assigned to the data control block REPORT. EX2 indicates that the required parameters were in registers 1 and 0 prior to execution of the macro instruction.

```
EX1      GETPOOL    REPORT,2,136
EX2      GETPOOL    (1),(0)
```

See also the example in the GETBUF macro instruction description.

GETSEG -- Get a Page from a Disconnected Segment Group (O)

The GETSEG macro instruction will get a page from a disconnected segment group and place it in a buffer specified by the user.

L-form

Name	Operation	Operation
symbol	GETSEG	[DNAME=,] MF=L

E-form

Name	Operation	Operation
[symbol]	GETSEG	[DNAME=,] ADDRESS=,BUFFER=,MF=(E,list)

Standard form

Name	Operation	Operation
[symbol]	GETSEG	DNAME=,ADDRESS=,BUFFER=

Note: All operands are keyword.

DNAME=
specifies the eight character EBCDIC name of the disconnected segment group.

Specified as: name enclosed within apostrophies: in the E or standard form only, as the address of DNAME expressed as a relocatable expression, RX address, or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the disconnected segment group name.

Default: none.

CAUTION: any user specified DNAME beginning with SYS will be rejected by the system.

ADDRESS=
specifies the relative page address of the disconnected segment group page.

Specified as: in the E or standard form only, the address of a word containing the relative page address expressed as a relocatable expression, RX address, or register notation.

Default: none

BUFFER=
specifies the page aligned address into which the disconnected page will be placed.

Specified as: in the E or standard form only, the address of a word containing the virtual storage address expressed as a relocatable expression, RX address, or register notation.

Default: none.

Return codes: upon return from execution of GETSEG, register 15 will contain a return code as follows:

Code	Meaning
X'00'	successful
X'08'	DNAME invalid
X'12'	segment not available to user class
X'16'	invalid address
X'40'	system error

Register 1 will contain the address of the Nameseg Parameter List.

Note: the DSECT CHANSG covers the Nameseg Parameter List.

Example:

```
        .
        .
        .
        LA   R5,32          32 pages in disconnected segment group
```

```
|              SR   R6,R6              initialize disconnected page address
|              LA   R7,WORKPAGE        get address of VM work page

|    ARRAYLOP  DS   0H
|              GETSEG  DNAME=ARRAY1,ADDRESS=(R6),BUFFER=(R7)
                 .
                 .
                 .
|              A    R6,=F'4096'        address of next disconnected page
|              BCT  R5,ARRAYLOP        process all pages
                 .
                 .
                 .
|    ARRAY1    DC   CL8'DIARRAY'       disconnected segment group name
```

GTWAR -- Write Record on SYSOUT and Read Response from SYSIN (S)

The GTWAR macro instruction writes a message on the user's SYSOUT, then reads the next available record from the user's SYSIN into the designated area of the user's available virtual storage.

Standard form:

Name	Operation	Operand
[symbol]	GTWAR	message,length of message,response, length of response [,translation code]

L- and E-form:

Name	Operation	Operand
[symbol]	GTWAR	[message],[length of message],[response], [length of response][,translation code] ,MF={L|(E,list)}

Note: A symbol is required in the name field with the L-form. Any of the first four operands that is omitted in the L-form must be supplied in the E-form.

message
: specifies the address of the area containing the message text. The message may include characters that can be represented in the terminal character set, including blanks, parentheses, and commas.

  Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, an RX address.

length of message
: specifies the address of a fullword that contains the length of the message to be issued. If the message is longer than the maximum line length for SYSOUT, GTWAR will write as many lines (or records) as are necessary, up to a maximum message length of 512 bytes.

  Specified as: Same as the first operand.

response
: specifies the address of the area into which the input record is to be placed.

Specified as: Same as the first operand.

length of response
specifies the address of a fullword containing the length of the expected input record. On return, the actual record length is stored in the address specified by this operand.

Specified as: Same as the first operand.

translation code
specifies how the characters of the input and output messages are to be handled before transmission to the response area or to the user's SYSOUT. The procedure followed when no translation code is specified is described in the Programming Notes. If a translation code is specified, the procedure becomes:

For input (response) data: All characters (with both 00 and non-zero function codes) are transmitted to the response area without translation. The functions defined for control characters are not performed.

For output messages: Characters with 00 function codes are transmitted to the user's SYSOUT without translation. Control characters are handled as when no translation code is specified (the control function is performed, and the characters are not translated or transmitted to SYSOUT).

Specified as:

SIC or 1 - no translation on input
2 - no translation on output
3 - no translation on input or output

Default: Translation on input and output.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

| Programming Notes: GTWAR is executed as a 'write with available response'. If the user has buffered input active (INMODE=S), the output message data is ignored and not displayed to the user, and the next record from the input queue is returned to the program. If the programmer wants to ensure that the message is written and that the input record is in response to the message, the GTWSR macro should be used.

| If no translation code is specified, handling of the input and output message characters is as follows:

For input (response) data: The Input Character Translation Table consists of two 256-byte sections; the first section contains the translation table to be used for translating the input characters, and the second section contains the function codes that control the handling of the corresponding input characters. For 00 function codes, the input characters are translated as determined by the first section of the table and transmitted to the input area. Nonzero function codes indicate a control function, such as backspace or cancel; when a character with a nonzero function code is encountered, the indicated function is performed, but the character is neither translated nor transmitted to the input area.

For output messages: The Output Character Translation Table is used for output messages in the same way that the Input Character Translation Table is used for input data: to determine whether the message characters are to be translated and transmitted to the user's SYSOUT or whether the message characters indicate that a control function is to be executed.

Both the character translation tables are defined by the system. However, the user may create and use his own translation tables; see Command System User's Guide and the CTT and OCTT operands of the MCAST macro instruction for more information.

If a continuation is indicated (the record extends over more than one print line), the user must provide a GATRD macro instruction to fetch the next portion of the record.

| A response is truncated only if it is longer than the length specified. Truncation begins with the rightmost character.

If GTWAR is executed in a loop and the user wishes to have the same value for the expected record length each time, he must reinitialize the length of response field each time GTWAR is to be executed.

Note: Only that portion of the record from the pointer on is available to the user. See Appendix I of Assembler Programmer's Guide for more information on record formats.

Return Data: At conclusion of execution of the GTWAR macro instruction, register 15 contains two bytes of coded information (hexadecimal) in bits 16-31; see Figure 10 (see the description of GATRD) for these codes. On return, the actual record length (in bytes) is stored in the address specified by the length of response operand.

Example: A 16-byte message (that is, 16 bytes assigned the translation code in the CTT) is written on SYSOUT and a 120-byte record is read from SYSIN into an area called ADLE.

```
     EX1      GTWAR       VICTOR,LARRY,ADLE,DAZE
```

In this example, the user has coded elsewhere in the program:

```
     VICTOR   DC          C'COMPLETED FIRSTR'
     LARRY    DC          F'16'
     ADLE     DC          CL120
     DAZE     DC          F'120'
```

GTWRC -- Write Record on SYSOUT with Carriage Control (S)

The GTWRC macro instruction writes a message on the user's SYSOUT, from an area in storage, with an extended FORTRAN carriage control character (the interpretation given the character in conversational tasks is described below). The carriage control character is not written (see the programming notes).

Standard form:

Name	Operation	Operand
[symbol]	GTWRC	message area,message length[,SIC]

L- and E-form:

Name	Operation	Operand
[symbol]	GTWRC	[message area][,message length][,SIC] ,MF={L|(E,list)}

Note: A symbol is required in the name field with MF=L. Any operand that is omitted in the L-form must be supplied in the E-form.

message area
: specifies the address of the area containing the message text. The message may include any characters that can be represented in the terminal character set, including blanks, parentheses, and commas.

  Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, an RX address.

message length
: specifies the address of a fullword that contains the length of the message to be issued. If the message is longer than the maximum line length of SYSOUT, GTWRC will write as many lines (or records) as are necessary up to a maximum message length of 512 bytes. The carriage control character should be included in the length count.

  Specified as: Same as the first operand.

SIC
: specifies how the characters of the message are to be handled before transmission to the user's SYSOUT. The procedure followed when SIC is not specified is described in the Programming Notes. If SIC is specified, characters with 00 function codes are transmitted to the user's SYSOUT without translation. Control characters are handled as when SIC is not specified (the control function is performed, and the characters are not translated or transmitted to SYSOUT).

Specified as: SIC

Default: Message characters are translated and control functions are performed as described in the Programming Notes.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Return Data: The return codes from GTWRC are shown in Figure 11 (see the description of GATWR).

Programming Notes: When using GTWRC, a single line passed to the GATE routine may be up to 513 characters. The carriage control character must be in the first position of the output line.

The GATE routine will ensure that all nonconversational SYSOUT records will be generated with carriage control characters. Normally, for nonconversational output from GATWR and GTWAR, GATE inserts a blank carriage control character into each record. However, after a given number of lines, a skip to channel 1 will be inserted instead. The number of lines per page used to control page space skipping is 54, unless changed by a DEFAULT command.

In conversational tasks, the GATE routine interprets the FORTRAN carriage control characters (see Appendix D) as follows:

1. For a "space before printing" character, a number of carriage returns one less than the number of spaces required are inserted at the beginning of the text as a separate message.

2. A "skip to channel" character is treated as a "triple space before printing" character (see 1. above).

3. For either "space suppression" or zero spaces before printing, a "single space before printing" character is assumed (thus none of the anomalies of the various terminal devices need be recognized).

4. A screen command carriage control character (S) for terminals other than 3270s causes the GTWRC to be NOOPed and a successful return is made to the program.

5. A character other than a carriage control character is treated as a "single space before printing" character.

Note: The spacing character at the beginning of the line is not printed.

Example: A 16-character record is to be written on SYSOUT:

```
EX1      GTWRC      RECOUT,LENGTH
```

In this example the user has coded elsewhere in the program:

```
RECOUT   DC         C'-COMPLETED ROUND1'    skip 3 lines
LENGTH   DC         F'17'                   before printing
```

GTWSR -- Write Record on SYSOUT and Read Record from Terminal SYSIN (S)

The GTWSR macro instruction writes a message on SYSOUT and reads the response from the terminal keyboard (SYSIN) in conversational tasks only. Use of this macro instruction in a nonconversational task causes termination of the task; however, the message is written on SYSOUT.

Standard form:

Name	Operation	Operand
[symbol]	GTWSR	message,length of message,response, length of response[,translation code]

L- and E-form:

Name	Operation	Operand
[symbol]	GTWSR	[message],[length of message],[response], [length of response],[translation code] ,MF={L|(E,list)}

Note: A symbol is required in the name field with MF=L. Any of the first four operands that is omitted from the L-form must be supplied with the E-form.

message
: specifies the address of the area containing the message text. The message may include any characters that can be represented in the terminal character set, including blanks, parentheses, and commas.

  Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also as register notation (2 through 12); in the E-form only, also as an RX address.

length of message
: specifies the address of a fullword containing the length of the message to be issued. If the message is longer than the maximum line length for SYSOUT, GTWSR writes as many lines (or records) as are necessary up to a maximum message length of 512 bytes.

  Specified as: Same as the first operand.

response
: specifies the address of the area into which the expected input record is to be placed.

  Specified as: Same as the first operand.

length of response
: specifies the address of a fullword containing the length of the expected input record. On return, the actual record length is stored in the address specified by this operand.

  Specified as: Same as the first operand.

translation code
: specifies how the characters of the input and output messages are to be handled before transmission to the response area or to the user's SYSOUT. The procedure followed when no translation code is specified is described in the Programming Notes. If a translation code is specified, the procedure becomes:

  For input (response) data: All characters (with both 00 and non-zero function codes) are transmitted to the response area without translation. The functions defined for control characters are not performed.

For output messages: Characters with 00 function codes are transmitted to the user's SYSOUT without translation. Control characters are handled as when no translation code is specified (the control function is performed, and the characters are not translated or transmitted to SYSOUT).

Specified as:
SIC or 1 - no translation on input
2 - no translation on output
3 - no translation on input or output

Default: Translation on input and output.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: If no translation code is specified, handling of the input and output message characters is as follows:

For input (response) data: The Input Character Translation Table consists of two 256-byte sections; the first section contains the translation table to be used for translating the input characters, and the second section contains the function codes that control the handling of the corresponding input characters. For 00 function codes, the input characters are translated as determined by the first section of the table and transmitted to the input area. Nonzero function codes indicate a control function, such as backspace or cancel; when a character with a nonzero function code is encountered, the indicated function is performed, but the character is neither translated nor transmitted to the input area.

For output messages: The Output Character Translation Table is used for output messages in the same way that the Input Character Translation Table is used for input data: to determine whether the message characters are to be translated and transmitted to the user's SYSOUT or whether the message characters indicate that a control function is to be executed.

Both the character translation tables are defined by the system. However, the user may create and use his own translation tables; see Command System User's Guide and the CTT and OCTT operands of the MCAST macro instruction for more information.

If a continuation is indicated (the record extends more than one print line), the user must provide a GATRD macro instruction to fetch the next portion of the record.

A response is truncated only if it is longer than the length specified by the user. Truncation begins with the rightmost character.

Return Data: At completion of execution of the GTWSR macro instruction, register 15 contains two bytes of coded information (hexadecimal) in bits 16-31. These codes are shown in Figure 10; however, since this macro instruction cannot be executed in nonconversational mode, the nonconversational mode return codes are not issued.

On return, the actual length of the input record (in bytes) is stored at the address specified by the length of response operand.

If GTWSR is executed in a loop and the user wishes to have the same value for the expected record length each time, he must reinitialize the length of response field each time GTWSR is to be executed.

Note: Only that portion of the record from the pointer on is available to the user. See Appendix C of Data Management Facilities for more information on record formats.

Example: In the following example, a 16-byte message is written on SYSOUT and a 120-byte record is read from the user's terminal (SYSIN) into area READIN:

```
 EXI        GTWSR      OAREA,OLENGTH,READIN,ILENGTH
```

In the example, the user has coded elsewhere in the program:

```
 OAREA      DC         C'COMPLETED FIRSTR'
 OLENGTH    DC         F'16'
 READIN     DC         CL120
 ILENGTH    DC         F'120'
```

HASH -- Provide a Hash Value (R)

The HASH macro instruction provides the issuer with a hash value for a specified name (for example, a dictionary entry).

Name	Operation	Operand
[symbol]	HASH	name,place

name
: specifies the address of an eight-byte field that contains a name of one to eight characters. If less than eight characters, the name must be left-aligned and padded with trailing blanks. Invalid characters will cause a diagnostic error message to be issued.

  Specified as: An RX address, or as register notation. If register notation is used, the address must first be loaded into the specified register.

place
: specifies the address where the one-byte hash value is to be placed.

  Specified as: An RX address, or as register notation. If register notation is used, the address must first be loaded into the specified register.

Execution: The eight-character string specified in the first operand is folded and the result multiplied by 5^{13}. The product is masked by 127, and the result is multiplied by 2.

Return Data: The low-order byte of the result obtained from the hashing algorithm is placed at the location specified by the second operand.

INTINQ -- Interruption Inquiry (O)

The INTINQ macro instruction relinquishes control until more information is available, maintains control in a wait state, or sets up a conditional branch. It causes an examination of interruption information queued for an interrupt control block (ICB) defined as available to the system by a SIR macro instruction. The INTINQ macro instruction can be issued only from an interruption routine.

Name	Operation	Operand
[symbol]	INTINQ	icb address [,MODE={R|W|CLEAR|(C,branch address),TYP=code}]

icb address
specifies the address of an ICB that has been defined as being available to the system by means of a SIR macro instruction. This ICB should not be one that has been defined to the system with a lower priority than the ICB by which current entry to this routine was made, if both include the address of the same data control block.

Specified as: Register notation (2 through 12), or a relocatable expression.

MODE=
specifies one of four modes of inquiry.

Specified as:

R - specifies the interruption routine is to relinquish control until more interruption information of the type specified in the ICB associated with the interruption routine is available. If this information has already been queued by the system, this routine may immediately regain control. Control is to return to this routine at the instruction following the INTINQ macro instruction.

W - specifies the interruption routine is to enter a wait condition

pending availability of interrupt information of the type specified by the icb operand. Control is not to be given up although the wait condition may be interrupted by a routine although the wait condition may be interrupted by a routine of higher priority. At the end of the wait, execution is to be resumed with the instruction following the INTINQ macro instruction.

CLEAR - specifies that any interruptions queued for the routine indicated by the ICB operand are to be deleted. Processing is to continue with the next sequential instruction.

Note: These queued interruptions may or may not conform with interruption types currently defined in the ICB.

C - specifies a branch is to be taken to the location specified by the branch address operand, if the information specified by the TYP operand is found in the queue of interruption information. If it is not found, execution is to be resumed with the next sequential instruction. The branch address and TYP operands must be written if the C option is chosen.

Default: MODE=R

branch address
specifies the address to which control is to be transferred if interruption information of the type specified by the TYP operand is available. This operand is only specified if MODE=C is specified.

Specified as: Register notation (2 through 12), or a relocatable expression.

TYP=
specifies the type of interruption information to be the condition for the branch.

Specified as: Any of the INTTYP codes (as described in the SPEC, SAEC, SSEC, STEC, and SEEC macro instruction descriptions) as long as the INTTYP is consistent with the type of ICB defined by the icb address operand in this macro instruction. ANY is written if any interruption information of the type specified in the associated ICB is desired. TYP can be something other than the INTTYP specified in the associated ICB. TYP associated with SIEC should specify ANY.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The INTINQ macro instruction inspects the queue of interruption information; the subsequent course of action is determined by the availability of queued interruption information and the mode specified in the macro instruction. Determination of the subsequent action for modes R, W, and C is illustrated in the following chart. Additional information pertaining to the INTINQ macro instruction can be found under "Writing Interruption Servicing Routines" in Appendix I.

Mode	Required Interruption Information	Action
R	Available	Continue execution with next sequential instruction
	Not available	Relinquish control; resume execution with next sequential instruction when information available
W	Available	Continue execution with next sequential instruction
	Not available	Enter wait state until information available; then continue with next sequential instruction
C	Available	Branch to specified branch address
	Not available	Continue execution with next sequential instruction

Return Data: Conditions that cause special return codes are listed below, with the associated hexadecimal return code. These conditions apply only to the modes stated.

Code	Meaning
04	Undefined routine specified (modes C, W, CLEAR)
08	Erroneous parameter list, the conditions specified can never be met (modes W, C)

IOREQ -- Request an Input/Output Operation (S)

The IOREQ macro instruction (for the IOREQ facility) initiates an input/output operation that is specified by a virtual channel command word (VCCW). See the description of the VCCW macro instruction.

After an IOREQ macro instruction is issued, control returns to the problem program before the I/O operation is completed. The CHECK macro instruction must be used to ensure the completion of the I/O operation.

If an IOREQ macro instruction is used, the IMSK operand of the DCB macro instruction must be specified.

Standard form:

Name	Operation	Operand
[symbol]	IOREQ	decb name,{N|B},dcb address,vccw address, vccw number,starting vccw

L- and E-form:

Name	Operation	Operand
[symbol]	IOREQ	decb name,[{N|B}],[dcb address],[vccw address], [vccw number],[starting vccw][,MF={L|(E,list)}]

Note: The decb name specified in both the L- and E-forms identifies the parameter list; a symbol is not required in the name field of the L-form. Any optional operand that is omitted from the L-form must be supplied with the E-form.

decb name
specifies the name to be assigned to the data event control block (DECB) built by the macro expansion.

Specified as: In the standard and L-form, a symbol consisting of one to eight alphameric characters, the first of which is alphabetic; in the E-form, as an RX address, or register notation (1 through 12).

{N|B}
specifies whether the I/O operation is buffered (a buffer is needed to store the data; the request may be nonbuffered if the data buffer in the IORCB can be used for the data obtained).

Specified as:
N -- nonbuffered
B -- buffered

dcb address
specifies the address of the data control block opened for the requested I/O operation.

Specified as: In the standard and L-form, a relocatable expression; in the E-form only, also as an RX address; in the standard form or the E-form, also in register notation (2 through 12).

vccw address
specifies the address of a list of virtual channel command words built by the VCCW macro instruction.

Specified as: Same as the dcb address operand.

vccw number
specifies the number of virtual channel command words in the VCCW list.

Specified as: An absolute expression; in the standard and E-form, also in register notation (2 through 12).

starting vccw
specifies the number in the list of the VCCW that is to be executed first.

Specified as: An absolute expression; in the standard and E-form, also in register notation (2 through 12).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The IOREQ macro instruction builds a data event control block (DECB), whose address is the symbol coded for the decb name operand.

The format of the DECB is shown in Figure 12.

Offset	Size in Bytes	Field
+0	1	Event control block (ECB)
+1	3	Reserved by the system (user must not alter)
+4	2	Type field (buffered or nonbuffered IOREQ)
+6	2	Length field (for buffered only)
+8	4	DCB address
+12	4	Data area address (for buffered only)
+16	4	Pointer to status indicators
+20	4	VCCW list address
+21	2	Used by the system (user must not alter)
+26	1	Sense byte 0
+27	1	Sense byte 1
+28	1	VCCW list length in doublewords
+29	1	Offset from VCCW list in doublewords to starting VCCW
+30	2	Reserved by the system (user must not alter)
+32	8	Modified channel status word[1] (CSW)
+40	8	Sense bytes (0-7)

[1]Modified CSW differs only from CSW in that the first word contains the 32-bit address of the instruction causing a unit check or unit exception.

Figure 12. Data event control block (DECB) format

The DECB used for IOREQ must not be altered until the operation has been checked with a CHECK macro instruction.

If buffering is specified, the buffers built for read request VCCWs may have overlapping data areas. However, the complete buffer area needed for all the read request VCCWs must be a contiguous area. For write request VCCWs, buffer space is allocated for each VCCW, regardless of whether the areas used by the VCCWs have overlapping portions. Consequently, write request VCCWs do not have to form contiguous areas.

For buffered VCCW write requests, the contents of the given data address are used when the IOREQ macro instruction is issued, even if these contents will be changed by a read request in the VCCW.

Each IOREQ macro instruction that causes an input/output request to be executed accomplishes this request by building an IORCB. IORCBs are executed separately by the system unless they are "chained": chaining IORCBs saves time if a following IORCB channel program is executed before the previous IORCB's channel program is completed. Nonbuffered VCCW requests use the data buffer in the IORCB.

If chaining to the next IORCB is desired, the last command to be executed must be the last in the user's VCCW list and must have the IOC flag set (this instruction is usually a NOP). Chaining of IORCBs is accomplished by changing the last CCW in a command list to a TIC to the START command in the next IORCB. This starting CCW cannot be a TIC, and must be executable only once. IORCB chaining is allowed only between IORCBs on the same device. When chaining is requested, it is still necessary to check each IOREQ result by using the CHECK macro instruction.

Return Data: When execution of the IOREQ macro instruction is completed, register 15 contains a hexadecimal return code in its low-order byte:

Code	Significance
00	I/O initiated.

04	I/O not initiated. (The NCP value in the data control block is exceeded, the DECB is "active", or the DECB is in the "wait" state.)
08	I/O not initiated. The VCCW list contains an error. One of the first nine rules for forming VCCW lists has been violated (refer to the VCCW macro instruction).
0C	I/O not initiated. The area needed for IOREQ is too large. Reduce or change VCCW list.

LIBESRCH -- Locate Object Module in External Library (S)

The LIBESRCH macro instruction locates, in an external library, an object module that defines a specified symbolic name.

Standard form:

Name	Operation	Operand
[symbol]	LIBESRCH	list address,not-found exit

L- and E-form:

Name	Operation	Operand
[symbol]	LIBESRCH	[list address],[not-found exit],MF={L|(E,list)}

Note: In the L-form, a symbol is required in the name field. If the first two operands are not specified in the L-form, they must be specified in the E-form. Operands specified in the E-form overlay corresponding operands which were specified in the L-form. In MF=(E,list), list must specify the symbol in the name field of the L-form; or the symbol may be loaded into register 1 and list specified as (1).

list address
specifies the address of a five-word parameter list that you have provided. (See Initialization.)

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

not found exit
specifies the address that receives control when a defining module is not located or the input library index was zero.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address. If register notation is used, the address must first be loaded into the specified register.

Initialization: Prior to issuing LIBESRCH, parameters must be placed in a five-word list whose address is specified in the first operand. The list must contain:

Word 1	A pointer to an eight-character symbol for which an object module that defines the symbol is to be found.
Word 2	A library index, that is, a pointer to the header of the first DCB to be searched in the program library hierarchy. The entire program library chain, from the library defined by this pointer to and including SYSLIB, is searched until a

	definition for the symbol is found. (This may also be considered an output parameter since it is modified by LIBESRCH during processing to point to the current DCB header; see "Return Data" below.)
Word 3	A pointer to a nine-word location you have provided in which LIBESRCH places information associated with the located object module; the information placed in each word is shown in Table 2.
Words 4 and 5	An eight-byte location into which LIBESRCH places the data definition name associated with the located object module.

<u>CAUTIONS</u>: If you include this macro instruction in a module that is declared privileged (through use of the DCLASS macro instruction), you must place the address of a save area in register 13 before execution.

If the module in which the symbol is found is a member of a shared data set and you wish to change or erase the member before logoff, a STOW macro instruction (type-R) must be issued to release the read interlock set as a result of issuing LIBESRCH.

<u>Execution</u>: On execution of the instructions generated by LIBESRCH, the LIBESEARCH routine in the dynamic loader searches the program library hierarchy for an object module which contains a definition for the symbol used as an argument (specified by the issuer of the LIBESRCH macro instruction in the first word of the parameter list pointed to by the first operand). The LIBESEARCH routine uses the FIND macro instruction for the search.

<u>Return Data</u>: If an object module is located, the data definition name associated with the program library in which it was found is placed in words four and five of the parameter list provided by the issuer of LIBESRCH. A pointer to the header of the DCB that was current when the module was found is placed in the second word of the parameter list in place of the pointer provided by the user before issuing LIBESRCH. Detailed information on the object module is placed by LIBESEARCH in the nine-word list pointed to by word three of the list provided by the user. This information is described in Table 2.

If the second word in the user-provided parameter list is set to zero, the object module was not found, and a branch was taken to the location specified in the second operand.

<u>Programming Notes</u>: ISA location ISAJLC points to the first DCB header in the library chain (the last-defined JOBLIB). ISA location ISASLP points to the DCB header for SYSLIB.

Word 0	Address of JFCB for library in which name was located
Word 1	DCB address for library where name was located
Word 2	Retrieval address of PMD
Word 3	Length of PMD in bytes
Word 4	Retrieval address of text
Word 5	Length of text in bytes
Word 6	Retrieval address of ISD
Word 7	Length of ISD in bytes
Word 8	SYSLIB switch - zero if library where name was located is not SYSLIB, nonzero if it is

Table 2. Information returned by LIBESRCH if module located

LOAD -- Load and Retain a Module (R)

The LOAD macro instruction is used to load a specified object module into the user's virtual storage; all other object modules to which it (and they) are implicitly linked, are also loaded. The specified module cannot be released until the task logs off, executes a DELETE macro instruction, or executes an UNLOAD command (refer to Command System User's Guide). Note that this macro instruction does not initiate execution of the specified program.

Name	Operation	Operand
[symbol]	LOAD	{EP=entry point|EPLOC=address of adcon group}

EP=
specifies the symbolic name of an entry point in the module to be loaded. The name must be the name of a control section, the name in the operand field of an assembler language ENTRY statement, or a module name.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

EPLOC=
specifies the address of the explicit adcon group representing the module to be loaded.

Specified as: An RX address, or register notation (1 through 12).

The adcon group can be generated by including in the same program:

```
L     ADCON     LOAD,EP=entry point name
```

Programming Notes: If the module has already been loaded, this macro instruction is ignored.

The ADDC2L byte (see the description of the ADCON macro instruction) may be used to direct the dynamic loader to a course of action when the specified module cannot be loaded. If ADDC2L is set to 0, the loader takes a system-prescribed error exit.

If an explicit adcon group is to be used for a LOAD macro instruction, it must first be armed, unless it has not yet been used and it was generated by an ADCON macro instruction. (Refer to the ARM and ADCON macro instruction descriptions.) If a loaded module has been deleted and the user wants to load it again, the same explicit adcon group may be reused, provided it is rearmed.

| Return Data: If operand EP= is specified, register 15 contains the address of the specified entry point in the loaded module. If operand EPLOC= is specified, the contents of register 15 are not altered by the execution of the LOAD macro.

Examples: 1) If a module whose entry point name is ROGER, is to be loaded, the following ADCON macro instruction is specified:

```
     TERI       ADCON      LOAD,EP=ROGER
```

Upon execution, LOAD EPLOC=TERI causes the module associated with the entry-point name ROGER to be placed into virtual storage.

2) Upon execution of this LOAD macro instruction, a copy of the module associated with the entry-point name ALPHA is placed into virtual storage and register 15 contains the address of ALPHA

```
                LOAD EP=ALPHA
```

3) Before issuing this LOAD macro instruction, the user loads the address of TERI into register 1. The effect of this instruction is then the same as in Example 1.

```
                LOAD EPLOC=(1)
```

LPCEDIT -- Call Editor from LPC (O)

The LPCEDIT macro instruction is used by a language processor controller to invoke editing facilities for line data sets, or to prompt with an underscore for region data sets.

Name	Operation	Operand
[symbol]	LPCEDIT	

Note: There are no operands. See Initialization.

Initialization: If this macro instruction is executed in a privileged module, the most recently issued DCLASS macro instruction must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in Register 13 before this macro instruction is executed.

It is the user's responsibility to set up register 0 before this macro instruction is executed, to indicate (by a zero value in the register) that the LPC is willing to be implicitly ended by the invocation of another LPC (a nonzero value indicates that the LPC should not be implicitly ended).

Programming Notes: The LPCEDIT macro instruction is issued following the LPCINIT macro instruction. As described more fully under "Text Editing" in Command System User's Guide, if the key length of the source data set is seven, a line data set is assumed and the system prompts with the last line number plus 100; editing facilities such as REVISE and INSERT may then be used. If the key length is greater than seven, LPCEDIT assumes the user wants to create a region data set and prompts him with an underscore to enter a command.

LPCINIT -- Initialize Edit Controller for LPC (S)

The LPCINIT macro instruction identifies the module in which it is issued as a language processor controller; it initializes the text editor, allowing the use of system text-editing facilities.

Standard form:

Name	Operation	Operand
[symbol]	LPCINIT	name,dcb address,[transient entry point], [preproc entry point],[postproc entry point], [early end entry point],[scan address],[trantab], [enable],[base],[increment][,implicit end]

L- and E-form:

Name	Operation	Operand
[symbol]	LPCINIT	[name],[dcb address],[transient entry point], [preproc entry point],[postproc entry point], [early end entry point],[scan address],[trantab], [enable],[base],[increment],[implicit end], MF={L|(E,list)}

Note: A symbol is required in the name field of the L-form. If either of the first two operands is omitted from the L-form, it must be supplied with the E-form.

name
 specifies the name of the user's language processor.

 Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

dcb address
 specifies the data control block for the source data set.

 Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, as register notation (2 through 12); in the E-form only, also as an explicit or implicit address.

transient entry point
 specifies the address of a three-word field containing the V-type and R-type address constants of the entry point in the user's program to be taken for transient commands, and a pointer to the field into which the transient command is to be moved (this latter field must be preceded by a fullword field containing the maximum length that can be read into the specified area). A transient command is a command with a vertical (|) as its first character. This facility enables the user to perform his own command analysis.

 When control is passed to the specified entry point, register 1 contains a pointer to a pointer to the area containing the transient command string. (The transient command statement prefix character -- see the TRP operand of the MCAST macro instruction -- will have been stripped off.) The length field contains the actual length read in (in the low-order two bytes) and the SYSIN return code (in the high-order two bytes).

 Specified as: Same as the dcb address operand.

 Default: A message is returned and the command is cancelled.

preproc entry point
 specifies the address of a two-word field containing the V-type and R-type address constants of the entry point in the user's program to which control is returned when an END command is encountered. If there is an active language processor controller (LPC), and the

same or another LPC is activated by an LPCINIT macro instruction, an END command is assumed and the preproc entry point is taken (this is called an implicit END).

Specified as: Same as the dcb address operand.

Default: No preprocessor is invoked.

postproc entry point
specifies the address of a two-word field containing the V-type and R-type address constants of the entry point in the user's program to which control is returned after the LPC marks the current language processor complete and makes the text editor facilities no longer available.

Specified as: Same as the dcb address operand.

Default: No postprocessor is invoked.

early end entry point
specifies the address of a two-word field containing the V-type and R-type address constants of the entry point in the user's program to which control is passed when the language processor controller is terminated other than by an END command.

Specified as: Same as the dcb operand.

Default: The preprocessor entry point is used if there is one; if not, the current LPC is ended and the postprocessor entry point is used (if there is none, an error message is returned).

scan address
specifies the address of a two-word field containing the V-type and R-type address constants of the entry point of the routine that the language processor controller will use to scan new lines as they are entered.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, also as an RX address.

Default: A zero is placed in the parameter list.

trantab
specifies whether a transaction table is to be kept.

Specified as: Y or N

Default: N (a zero is placed in the parameter list).

enable
specifies whether the language processor controller is to run enabled or disabled.

Specified as: Y or N

Default: N (a zero is placed in the parameter list).

base
specifies the initial value for the current line pointer.

Specified as: A number that is a multiple of 100.

Default: A zero is placed in the parameter list.

increment
: specifies the line number increment to be used when none is specified in a command.

  Specified as: A number.

  Default: A zero is placed in the parameter list.

implicit end
: specifies whether the language processor controller will allow itself to be subject to an implicit end (the LPC is ended because another LPCINIT macro instruction initiates the same or another LPC).

  Specified as: Y or N

  Default: Y (if this operand is omitted, a zero is placed in the parameter list).

CAUTION: Specifying any value other than Y or N for the trantab, enable, and implicit operands causes an error message to be generated.

Initialization: If this macro instruction is issued in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must have been placed in register 13 before this macro instruction is executed.

Return Data: The following hexadecimal return codes are placed in register 15 when control is returned to the user:

Code	Meaning
00	Successful completion of LPCINIT.
04	A previous LPC is outstanding and has indicated it is not to be implicitly ended, and the user, when prompted, chose not to override that choice.
08	Specified data control block is not open.
0C	Specified read-in area for transient commands is not read-write access for the user.

| MARKRTRN -- Indicate Return from Called Program (O)

| The MARKRTRN macro instruction turns on the low order bit in the for-
| ward pointer (3rd word) of the caller's save area. This indicates to
| the calling program that the called program has returned.

Name	Operation	Operand
	MARKRTRN	

| Note: There are no operands.

| Example:

```
|        MARKRTRN
|    +   OI    11(13),X'01'       indicate return was done
```

| Initialization: Register 13 must be initialized with the address of
| your save area before using this macro. MARKRTRN normally follows imme-
| diately after a call-type macro.

MCAST -- Modify Character and Switch Table (S)

The MCAST macro instruction temporarily replaces the Character Translation Table (in a user's session profile) with a user-specified Character Translation Table (CTT) and temporarily overlays the control function characters, such as continuation characters or end-of-block characters (also in the session profile), with new functional control characters. The CTT and the Profile Character and Switch Table in the session profile are both overlaid for the duration of the user's terminal session. If desired, the changes can be permanently recorded in the user's profile by issuance of the PROFILE command.

All forms:

Name	Operation	Operand
[symbol]	MCAST	[CTT=address][,EOB=address][,CONT=address] [,CLP=address][,TRP=address][,PCC=address] [,SSM=address][,USM=address] [,PL=address,CP=address][,KC=address][,RS=address] [,OCTT=address][,MF={L|(E,list)}]

Note: A symbol is required in the name field of the L-form. An operand may only be specified in the E-form if it is also specified in the L-form. If the MF operand is omitted, the standard form is assumed. See also "L- and E-Form Use" below.

CTT=
specifies the address of a pointer to the 512-byte Character Translation Table (CTT) that is temporarily to replace the one in the user's session profile, (see Command System User's Guide for a discussion of the CTT).

Specified as: In the standard form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, as an RX address.

Default: The current value is retained.

EOB=
specifies the address of the source list end-of-block character that is to replace the one currently existing in the user's session profile. This character defines the end of an input block in the source list to the Command Analyzer. The value must be specified as X'26'.

Specified as: Same as the first operand.

Default: The current value is retained.

CONT=
specifies the address of the continuation character that is to replace the one in the user's session profile. If the last character before a carriage return is a control language continuation character, the line of input is continued past the carriage return to include the next line entered at the terminal. The initial CONT character is a hyphen, X'60'.

Specified as: Same as the first operand.

Default: The current value is retained.

CLP=
specifies the address of the control language prefix character that is to replace the one in the user's session profile. Entry of this

character at the terminal requests the system to execute the command following the character immediately. Initially, this character is defined as an underscore, X'6D'.

Specified as: Same as the first operand.

Default: The current value is retained.

TRP=

specifies the address of the transient command statement prefix character that is to replace the one in the user's session profile. When the user codes this as the first character of a command in a

command statement, it is recognized and control is passed to a predefined entry point in the language processor currently being executed. The language processor then immediately processes that command and either returns control to the next sequential command in the command statement or performs other processing. The initial TRP character is a vertical bar, X'4F'.

Specified as: Same as the first operand.

Default: The current value is retained.

RCC=

specifies for input the address of the record concatenation character that is to replace the previously-defined characters in the user's session profile. If concatenation is active (the user has issued DEFAULT CONREC=Y at his terminal), the text editor examines the last nonblank character of the input line. When it is the concatenation character defined by this operand or defined previously, the next input line is added to the line to become part of a single record. The system-supplied character is the colon (X'7A'). See the RS operand for concatenating output messages.

Specified as: Same as the first operand.

Default: The current value is retained.

SSM=

specifies the address of the new User Prompter System Scope Mask. This mask is used in conjunction with the explainable words of messages written to the terminal from the system message file. When the user requests an explanation for such a word (via the EXPLAIN command) this mask determines the pattern for searching through the hierarchy of word explanations in the message file. Each bit position in the one-byte mask corresponds to a byte in an eight-character label or message ID associated with a message containing the explainable word. Each bit that is set on (from right, 7-bit; to left, 0-bit) causes a different level message file to be searched once. A complete scan is made and all indicated searches are executed.

The number of bytes in the message ID compared in each search is equal to the number of bits to the left of the bit that is set on, plus 1, for the bit causing the search to be made. Thus if the 7-bit were set on, a search of 8 characters would be made; if the 1-bit were set on, a search of 2 characters would be made. The search for a particular level of explanation for a message begins by scanning the mask from right to left for bits that are on. If the first search does not locate the desired word, the scan continues to the next search-indicating bit, etc., until the complete mask has been scanned and all levels of search have been completed. The initial default is X'29'.

Specified as: Same as the first operand.

Default: The current value is retained.

USM=

specifies the address of the new User Scope Mask. Each bit represents a level at which a search and comparison is made to locate explainable words in a user-defined message file (located in the user library). The user may set this mask according to his own search logic (see the SSM operand above for further information). The initial default value for USM is also X'29'.

Specified as: Same as the first operand.

Default: The current value is retained.

PL=

specifies the address of a byte containing the length of the Command Prompt String. This length cannot exceed 8. This length initially reflects the 3-character default value of the command prompt string operand (see CP below).

Specified as: Same as the first operand.

Default: The current value is retained.

CP=

specifies the address of a system Command Prompt String that is to replace the one in the user's session profile. This may be a string of up to eight characters. The initial default is an underscore followed by a backspace and a carriage-return suppression character (X'7A'). The system uses this string to prompt the user to enter commands at the terminal. If this operand is specified, the PL operand must also be specified.

Specified as: Same as the first operand.

Default: The current value is retained.

KC=

specifies the address of a one-byte keyboard/card reader switch. This switch indicates the type of device from which input will be accepted by the system. It may be set with a K for a keyboard, or with an E to indicate either the keyboard or the card reader. E serves as the default parameter and causes the input device to be determined by examining the SYSIN parameter previously established in the user library by a DEFAULT command. The SYSIN parameter can be set to K or C; it is initially set to K. The KC operand is initially set to E.

Specified as: Same as the first operand.

Default: The current value is retained.

RS=

specifies the address of the carriage-return suppression character that is to replace the one in the user's session profile. Normally a carriage return is executed after every message written on the terminal by the GATE routine; however, when this character appears as the last character in a message, no carriage return occurs. The next message written on the terminal begins where the last one left off. The initial value for RS is the colon, X'7A'. The suppression character is not written on SYSOUT.

Specified as: Same as the first operand.

Default: The current value is retained.

OCTT

specifies the address of a pointer to the 512-byte Output Character Translation Table that is temporarily to replace the system-supplied table in the user's session profile. (See Command System User's Guide for a discussion of the OCTT).

Specified as: Same as the first operand.

Default: The system-supplied table is used.

CAUTION: If a user issues a PROFILE command via an OBEY macro instruction following MCAST, his user profile will be permanently changed. Users should make certain, for subsequent program executions, that when communicating with those programs the updated control and functional characters are employed.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The Character Translation Table consists of 512 contiguous bytes. The table is broken into two 256-byte sections. The first section contains the internal binary representation for each of the possible hexadecimal codes from 00 to FF, in sequential order. The second 256-byte section contains the function codes, each displaced 256 bytes from its related hexadecimal translation code. The available function codes and the Character Translation Table are described in Appendix C of Command System User's Guide; a copy of the table can be found in that publication. A user must generate his new Character Translation Table according to the prescribed format.

Since the MCAST macro instruction allows new interpretations for all current characters and control function switches, it should be particularly useful for text-editing applications, where unique character interpretation is desired and line control changes are needed.

With the varying line length capabilities of different devices, it may become necessary to divide a line of input. The RCC operand can be used to accomplish this. For ordinary printed text, a user might make this character a space: a line would then be broken between words.

L- and E-Form Use: An example of L- and E-form use is:

```
 SUE      MCAST     RCC=/,KC=K,MF=L
          MCAST     KC=B,MF=(E,SUE)
```

When the E-form of this macro instruction is executed, the specification of the SYSIN device indicated via the L-form (K) is replaced by the specification indicated in the E-form (B). The system will then accept input from the keyboard and the card reader.

Example: The user is replacing the Character Translation Table in the user's session profile with the characters indicated in the 512-byte table located at NEWTAB. In addition, the end-of-block character in the Profile Character and Switch Table in the user's session profile is being changed to an asterisk (*) and the command prompt string is being changed to a number sign (#).

```
                    MCAST     CTT=TABADDR,EOB=EOBCHAR,
                              PL=LNGTHCB,CP=NEWPRMPT
                      .
                      .
                      .
                      .
    LENGTHCB        DC        AL1(L'NEWPRMPT)
    NEWPRMPT        DC        C'#'
    EOBCHAR         DC        C'*'
    TABADDR         DC        AL3(NEWTAB)
    NEWTAB          DS        0CL512
                      .
                      .
```

|

NOTE -- Provide Position Feedback (R)

The NOTE macro instruction (for BSAM) causes the relative position within a volume of a block just read or written to be placed in register 1. This relative position identifies the block for subsequent repositioning of the volume.

NOTE provides a block count for magnetic tape. For direct-access volumes, the count is the track number relative to the beginning of the data set portion on the volume and the record number within the track.

The NOTE macro instruction normally provides information for a subsequent POINT macro instruction.

Name	Operation	Operand
[symbol]	NOTE	dcb address

dcb address
specifies the address of the data control block opened for the current operation.

| Specified as: Register notation (1 through 12), or an RX address.

CAUTION: Abnormal termination occurs if the data control block specified by the user is not opened.

For a data set on magnetic tape, the NOTE macro instruction should not be issued for an unlabeled data set or a data set containing non-standard labels, if the data set is opened under either of these conditions:

1. A DDEF macro instruction or command has a disposition parameter of MOD.

2. An OPEN macro instruction specifies RDBACK.

The current block count in the data control block is not valid under the above conditions.

For a data set on magnetic tape, a NOTE macro instruction issued after a POINT macro instruction, without an intervening READ or WRITE macro instruction, does not return the relative address of the last record read or written. NOTE returns the data control block count minus 1, if the last I/O operation was not a READ (type SB); or it returns the data control block count plus 1, if the last I/O operation was a READ (type SB).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: All READ or WRITE requests must be checked for completion before the NOTE macro instruction is executed. The block identification provided is always within the current volume.

Return Data: Following execution of the NOTE macro instruction, the system places the block identification of the last block read or written in register 1.

The form of the block identification depends on whether a magnetic tape or direct access device is being used, as follows:

Magnetic Tape: If magnetic tape is used, the block identification is a four-byte block count of the form zzCC, where:

zz	- binary zero bytes;
CC	- the block number (binary) within the volume.

The block identification may be used in the POINT macro instruction to reposition the magnetic tape to the location of the block.

Direct Access Device: If a direct access device is used, the block identification is a four-byte value of the form TTRz, where

TT	- the track number relative to the beginning of the data set on the current volume (first track equals 0).
R	- the block number on that track (first data block equals 0).
z	- a binary zero byte.

If the last operation was a WRITE, an additional parameter is provided by NOTE, in register 0, in the form zzLL, where:

zz	= binary zero bytes.
LL	= the number (in binary) of bytes remaining on that track.

The initial relative address for the first record on a direct access device is (TT=0, R=0). The initial block count for the first record on a magnetic tape device that was not opened for RDBACK or MOD is (CC=0). The initial block count for the first record on a magnetic tape that was opened for RDBACK or MOD is CC minus 1 (CC= trailer label block count). NOTE is applicable only to direct access and magnetic tape devices. The address that is sent back in register 1 for any other equipment type is the data control block count minus 1 and is preceded by two bytes of binary 0.

OBEY -- Execute a Command or Command Statement (O)

The OBEY macro instruction allows the user to execute a command or command statement even though not in command mode. Upon execution of the OBEY macro instruction, the command or command statement specified via the macro instruction operands is executed; control is then returned to the user's program. OBEY may be used anywhere in the user's program.

Standard and E-Form:

Name	Operation	Operand
[symbol]	OBEY	[{address of command|'command'}] [,MF=(E,list)]

Note: If the MF= operand is omitted, the standard form is assumed.

L-form:

Name	Operation	Operand
symbol	OBEY	['command',]MF=L

address of command
: specifies the address of a fullword that contains the address of a location containing the command or command statement character string. (The byte preceding the command string must contain the length of the string.)

  Specified as: A relocatable expression, or register notation (1 through 12).

  Default: If neither the command nor its address is specified, it is assumed that register 1 contains the address of the command or command statement.

command
: specifies the command or command statement character string to be executed.

  Specified as: The command or command statement itself enclosed in apostrophes.

  Default: If neither the command nor its address is specified, it is assumed that register 1 contains the address of the command or command statement and that the byte before the command contains its length.

CAUTION: If no operand is specified, the address of the command character string must have been loaded into register 1 before execution of this macro instruction.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: If the user specifies the address of the command, the address must point to the first byte of the command or command character string, and the byte that precedes the character string must contain a count of the bytes in the character string. No special alignment is required.

L- and E-Form Use: An example of L- and E-form use is:

```
SUE       OBEY   COMADDR,MF=L
          LA     1,NAME
           .
           .
          OBEY                       (Where no operand implies that
           .                         register 1 has been previously
           .                         loaded with the address of a
           .                         fullword (NAME) containing the
           .                         address of COMADDR, and the
           .                         number of bytes composing
           .                         COMADDR is in the byte preceding
           .                         COMADDR)
NAME      DC     A(COMADDR)
          DC     AL1(L'COMADDR)
COMADDR   DC     C'EXECUTE PROG2'
```

```
              DC      AL1(L'OTHERCOM)
     OTHERCOM DC      C'EXECUTE PROG3'
```

When the E-form of this macro instruction is executed, the program specified via the L-form (PROG2) is replaced in the command string by the program (PROG3) indicated via the E-form of the macro instruction.

Examples:

```
               OBEY  'PROCDEF PAR1'     Obey a command
               OBEY  'EXECUTE MYPROG'
               OBEY  'BACK THISPRG'
               OBEY  COMADDR
               OBEY                      (Where no operand implies
                .                        that register 1 has been
                .                        previously loaded with the
                .                        address of a fullword
                .                        containing the address
                .                        of COMADDR, and the
                .                        number of bytes composing
               DC    AL1(L'COMADDR)      COMADDR is in the byte
     COMADDR   DC    C'EXECUTE PROGA'    preceding COMADDR)
```

OPEN -- Connect a Data Set to the System (S)

The OPEN macro instruction:

- connects one or more data sets to the system by completing the data control blocks containing their attributes,
- indicates the manner in which a data set is to be processed,
- creates and catalogs new VAM data sets, and
- initially positions the data set for processing.

Input labels are analyzed and output labels are created. Control is given to exit routines as specified in the data control block's exit list (BSAM and QSAM only). Any number of data sets and their associated options may be specified in the OPEN macro instruction.

Standard form:

Name	Operation	Operand
[symbol]	OPEN	({dcb address[,(option[,{REREAD|LEAVE}])]},...)

L-form and E-form:

Name	Operation	Operand
[symbol]	OPEN	[({dcb address[,(option[,{REREAD|LEAVE}])]},...),] MF={L|(E,list)}

Note: A symbol is required in the name field of the L-form.

dcb address
specifies the address of the data control block containing the attributes of the data set that is to be initialized.

Specified as: In the standard and L-form, a relocatable expression; in the standard and E-form, also as register notation; in the E-form only, also as an RX address.

option
specifies the intended method of input/output processing of the data set being connected to the system. The processing method to be specified is dependent on the data set organization and access method that is being used to perform the I/O processing.

Specified as: The various processing options, their meanings, and the access methods with which they can be specified are shown in Figure 13.

Option	Meaning	VAM	BSAM	QSAM	IOREQ
INPUT	The data set can be used as input only. This option is assumed if this operand is defaulted.	X	X	X	X
OUTPUT	The data set can be used for output only (except for VAM, in which input is allowed).	X	X	X	X
INOUT	Both input and output operations are allowed. The data set is positioned to the first record.	X	X	--	X
OUTIN	Both output and input operations are allowed. The data set is positioned to the last record.	X	X	--	X
UPDAT	The data set can be updated (see the note below).	X	X	X	X
RDBACK	An INPUT data set is to be read backwards.	--	X	X	--

Figure 13. Data set processing options

Note: Opening a VISAM data set for INOUT or OUTIN is equivalent to opening for UPDAT. When a data set is opened for UPDAT, however, the user must position to the desired record in the data set. If a new VAM data set is being opened for INOUT or INPUT (either requested or defaulted), the option is changed to OUTPUT. All other options (UPDAT, OUTIN, OUTPUT) are allowed for a new VAM data set.

Default: INPUT

REREAD|LEAVE
specifies for compatibility with OS and OS/VS, a parameter that is ignored by TSS because volumes are not mounted in parallel.

Specified as: REREAD or LEAVE, or it may be omitted.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

<u>CAUTION</u>: The following errors cause the results indicated in Figure 14.

Error	Result
Opening a data control block that is already open	No action
Specifying the address of an invalid data control block	Task terminated
Opening a data control block when a DDNAME in the data control block has not been provided	Nonconversational task terminated; prompting given if task is conversational
Opening a privileged data set by a nonprivileged user (BSAM, QSAM, VPAM and IOREQ only)	Task terminated
Opening a read-only data set and specifying an option other than INPUT	Task terminated
Opening a data control block when the DDNAME in the data control block does not correspond to the DDNAME in the DDEF macro instruction (or command)	Nonconversational task terminated; prompting given if task is conversational
Opening a data control block containing an invalid DSORG specification	Task terminated

Figure 14. Results of errors in opening a data set

<u>Programming Notes</u>: Any number of data control block addresses and associated options may be specified in the OPEN macro instruction. This facility allows parallel opening of the data control blocks and their associated data sets, which is more efficient than to open them individually. One of the services performed at this time is processing of labels of data sets or volumes.

<u>For VSAM</u>: When a shared VSAM data set is opened, a data set interlock is set according to the option operand. If INPUT is specified, a read interlock is set; if OUTPUT, INOUT, OUTIN, or UPDAT is specified, a write interlock is set. Rules for sharing VSAM data sets are given in Appendix K.

<u>For VISAM</u>: When a shared VISAM data set is opened, a data set interlock is set according to the option operand. If INPUT, INOUT, OUTIN, or UPDAT is specified, a read interlock is set; if OUTPUT is specified, a write interlock is set. Rules for sharing VISAM data sets are given in Appendix K.

<u>For BSAM</u>: If a DCB exit routine or a user-label exit routine is to be executed, the exit list address must be provided in the data control block. The format of the exit list, its use during the opening process, and exit routine requirements are discussed in Appendix A.

<u>For VAM</u>: A SYNAD EXIT OPTION bypasses the call to ABEND when certain errors are detected by OPEN VAM. Instead, control will be returned to the SYNAD routine specified in the DCBSYV and DCBSYR fields of the DCB.

In order to use this option the DCBIM80 flag in the DCB must be set 'on' before issuing the OPEN macro. Should an error occur and the SYNAD routine be given control, the cause of the error may be determined by examining the DCBEX1 and DCBEX2 fields in the DCB.

Example:

```
PROGP     PSECT
MYDCB     DCB       DDNAME=ANYDD,SYNAD=PROGSYN
           .
           .
           .
PROGC     CSECT
           .
           .
           .
          LA        REG,MYDCB         load addr DCB
          USING     CHADCB,REG        cover with DSECT
          OI        DCBI,DCBIM80      indicate I want control on error
          OPEN      MYDCB
           .
           .
           .
          EXIT
PROGSYN   DS        0H
           .
           .
           .
          examine DCBEX1 and DECEX2 for error
           .
           .
           .
          BR        R14
          END
```

Asynchronous VAM page out is the system default for writing data pages; that is, task execution will overlap I/O to the data set. To run synchronously, the DCBIN40 flag must be set as shown for the DCBIN80 flag in the example above.

L- and E-Form Use: The parameters specified in the E-form will overlay parameters specified in the L-form. The E-form may not specify more DCB operands than are specified in the L-form. The format of the parameter list generated by the OPEN macro instruction is described in Appendix L.

For example:

```
JOE      OPEN      (DATASET,,MORSET,,),MF=L
DEB      OPEN      (,,FOSET,,NUSEM),MF=(E,JOE)
```

When the E-form macro instruction is executed, the data control block FOSET replaces MORSET in the parameter list. Data control blocks with symbolic addresses DATASET, FOSET, and NUSEM are opened.

Examples: EX1 opens the data control block INVEN as an input data set. EX2 opens the two data control blocks INVEN and REPORT with different options. EX3 opens the two data control blocks INVEN and MASTER; they are opened for input data sets since INPUT is assumed when the option operand is omitted. EX4 generates a parameter list for opening INVEN, and EX5 opens INVEN.

```
EX1      OPEN      (INVEN,(INPUT))
EX2      OPEN      (INVEN,(INPUT),REPORT,(OUTPUT,LEAVE))
EX3      OPEN      (INVEN,,MASTER)
EX4      OPEN      (INVEN,(INPUT)),MF=L
EX5      OPEN      MF=(E,EX4)
```

PAUSE -- Enter Command Mode (R)

The PAUSE macro instruction switches a conversational task from program mode to command mode. A PAUSE macro instruction issued in a nonconversational task is ignored. During program stoppage, the user may issue any command he wishes directly from the terminal. The task can be returned to program mode by issuing a RUN command.

The word PAUSE and any optional message specified in the operand are displayed on SYSOUT.

Name	Operation	Operand
[symbol]	PAUSE	{address of message|'message'}

address of message
    specifies the address of the location in storage that contains the text of the message to be issued. The first byte of the message must contain the length, in bytes, of the message.

    Specified as: An RX address, or register notation (1 through 12).

message
    specifies the message to be issued.

    Specified as: The text of the message itself, enclosed in apostrophes.

Initialization: This macro instruction cannot be assembled in a privileged module unless the most recently issued DCLASS macro instruction in the assembly specified USER (see Appendix M) or if the DCLASS option is USER by default.

Programming Notes: If the user has control of interruptions before issuing a PAUSE macro instruction, the system regains control of them until a RUN command is issued.

Examples: In EX1 the message is supplied as text. In EX2 the message is given at location DARRY.

```
EX1  PAUSE  'PROG DECISION AT STMT LOOP3'
EX2  PAUSE  DARRY
```

PIREC -- Program Interruption Protection in System Code (O)

The PIREC macro instruction validates an address before the address is actually used as an operand in a program. This protects the PIREC issuer from possible addressing exceptions (possibly because a bad address was passed to the issuer of PIREC). PIREC tests an address by executing a test instruction set up by the programmer with the address as an operand. If the address is invalid, a program interruption occurs, and control returns to the programmer's error routine.

Name	Operation	Operand
[symbol]	PIREC	error exit,instruction location

error exit
: specifies the location of the programmer's error routine that will be used if certain program interruptions occur; diagnostic messages about the error can be issued by this routine.

  Specified as: A relocatable expression or register notation.

instruction location
: specifies the location of the test instruction to be executed for validating an address operand.

  Specified as: A relocatable expression or register notation.

Initialization: If PIREC is issued in a privileged CSECT, a DCLASS macro instruction with the PRIVILEGED option should be previously coded within the CSECT. If PIREC is issued in a nonprivileged module, the user can accept the USER setting by default. In cases where several DCLASS macro instructions have been coded within a CSECT, the last one issued, prior to issuing PIREC, must have established the appropriate setting.

A copy statement for CHAISA must be issued prior to issuing PIREC.

Execution: PIREC tests the global symbol (&CHDCLS) setting -- established by default or a DCLASS macro instruction specification -- for the CSECT in which PIREC is issued and sets in the ISA, either ISAPPIR or ISANPIR. These flags indicate whether a privileged or nonprivileged program interruption might occur due to execution of the test instruction. The test instruction is then executed via an Execute instruction. If its execution generates a program interruption, the appropriate ISA flag is automatically reset by the system's interruption processor. The processor recognizes program interruptions 4 (protection check), 5 (addressing error), and 6 (specification error), in nonprivileged code, and 5 and 6 in privileged code, and returns control to PIREC for their processing. PIREC again checks &CHDCLS and, based on that setting, tests the appropriate ISA flag to see if it has been reset; if reset, it indicates that a program interruption did occur. If an interruption occurred, PIREC exits to the user specified error routine. If no program interruption occurred, PIREC resets the flag in the ISA.

Programming Notes: PIREC is often used in system modules (for example, in BPKD processing) to test addresses passed to them from users. It protects the system program from having program interruptions occur due to user errors.

PIREC can be coded in privileged or nonprivileged code. Its use in nonprivileged code can provide an efficient way of checking protection without using the CKCLS system macro instruction.

Some useful test instructions and the possible reason, or reasons, for program interruptions resulting from their execution, are listed below:

Test Instruction	Possible Reason Code	Meaning of Code
TM	1 or 2	1 = Unassigned storage
MVC	1, 2, or 3	2 = Fetch protection in nonprivileged code
L	1, 2, or 4	3 = Storage protection in nonprivileged code
ST	1, 2, 3, or 4	4 = Word boundary
LH	1, 2, 5	5 = Halfword boundary
STH	1, 2, 3, or 5	

For nonprivileged program interruptions 4, 5, or 6, processing by the PIREC recovery routine overrides any recovery routine already specified via a SIR macro instruction.

Example: A system programmer wants to verify an address that has been passed to a system module by a user routine. He might code:

```
         DCLASS    PRIVILEGED             ESTABLISH INTERRUPT TYPE EXPECTED
         .
         .
         .
         LR        5,1                    PUT ADDR TO BE TESTED IN R5
ATEST    PIREC     ERR,EXE
+        USING     CHAISA,0
+        OI        ISAPPIR,ISAPPIRM       SET APPROPRIATE ISA FLAG: BASED ON
                                           DCLASS
+        EX        0,EXE                  EXECUTE "TEST INSTRUCTION"
+        TM        ISAPPIR,ISAPPIRM       DID PRIVILEGED PROG. INT. OCCUR?
+        BZ        ERR                    YES; BRANCH IF PROG. INT. OCCURRED.
+        NI        ISAPPIR,255-ISAPPIRM   NO: RESET ISA FLAGS
         B         CONTINUE               ADDRESS IS OK
EXE      TM        0(5),0                 "TEST INSTRUCTION" TO BE EXECUTED
ERR      ST        5,PSECTADD
         .                                PROMPT USER TASK OR MODULE WITH
         .                                DIAGNOSTIC MESSAGE
         .
```

POINT -- Position to a Block (R)

The POINT macro instruction (for BSAM) repositions a magnetic tape or direct access volume to a specified block within a data set on that volume. Thus the POINT macro instruction permits reading or writing of a sequential data set from any specified position.

The NOTE macro instruction may be used to provide the positioning information that is required for the POINT macro instruction.

Name	Operation	Operand
[symbol]	POINT	dcb address,block identification

dcb address
    specifies the address of the data control block opened for the data set being processed.

```
     .
     .
     .
WRITE      OUTDECB,SF,MYDCB,(4),100      This is the record to which
                                         the program will reposition.
     .
     .
     .
CHECK      OUTDECB
FREEBUF    MYDCB,4
     .
     .
     .
NOTE       MYDCB                         Note the position of the rec-
ST         1,SAVE                        ord under consideration.
     .
     .
     .
GETBUF     MYDCB,4                       Reposition to the record
POINT      MYDCB,SAVE                    being considered and
READ       INDECB,SF,MYDCB,(4),100       read it.
     .
     .
     .
```

PR -- Print a Data Set (S)

The PR macro instruction causes the specified data set to be listed in nonconversational mode on a high-speed line printer and, optionally, erases it from the catalog when printing is finished.

> Note: this macro instruction has one or more operands that can be used only by a systems programmer; these operand(s) are defined and specified in the System Programmer's Guide manual.

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	PR	{address of operand string
		|'DSNAME=data set name
		[,STARTNO=starting position]
		[,ENDNO=ending position]
		{,PRTSP={EDIT|1|2|3}
		|[,HEADER=H] [,LINES=lines per page] [,PAGE=P]}
		[,ERASE={Y|N}] [,ERROPT={ACCEPT|SKIP|END}]
		[,FORM=standard default region name]
		[,STATION=station id]
		[,FCB=fcb name] [,PAPER=paper type]
		[,COPIES=(nnn[,(GP,...)])]
		[,FLASH=(overlay name[,COUNT])]
		[,SYSUCS=users sysucs dsname] [,BURST={Y|N}]
		[,COPYMOD=copy modification data set name]
		[,TRC={Y|N}]}
		[,NPRIORTY=transmission priority]
		[,NETACCT=network account number]
		[,DELIVER=([prgmrnam][,room][,dept][,bldg])]
		[,PRTCLASS=printing output class]
		[,INDEX=indexing offset]
		[,EXTWTR=external writer name]
		[,MODTRC=table reference character]'}

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	PR	{address of operand string
		|'DSNAME=data set name
		[,STARTNO=starting position]
		[,ENDNO=ending position]
		{,PRTSP={EDIT|1|2|3}
		|[,HEADER=H] [,LINES=lines per page] [,PAGE=P]}
		[,ERASE={Y|N}] [,ERROPT={ACCEPT|SKIP|END}]
		[,FORM=standard default region name]
		[,STATION=station id]
		[,FCB=fcb name] [,PAPER=paper type]
		[,COPIES=(nnn[,(GP,...)])]
		[,FLASH=(overlay name[,COUNT])]
		[,SYSUCS=users sysucs dsname] [,BURST={Y|N}]
		[,COPYMOD=copy modification data set name]
		[,TRC={Y|N}]}
		[,NPRIORTY=transmission priority]
		[,NETACCT=network account number]
		[,DELIVER=([prgmrnam][,room][,dept][,bldg])]
		[,PRTCLASS=printing output class]
		[,INDEX=indexing offset]
		[,EXTWTR=external writer name]
		[,MODTRC=table reference character]'}
		,MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	PR	MF=(E,list)

address of operand string
  specifies the address of the first operand in the operand string.

  Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified in the standard form of the macro operand as a character string enclosed in apostrophes, as shown.

DSNAME
  identifies the data set that is to be printed; VAM data sets must be cataloged; BSAM data sets must be defined within the current task by a DDEF command or must be cataloged.

  Specified as: a fully qualified data set name.

  Default: none.

STARTNO
  specifies the byte number at which printing is to start for each data set record.

  Specified as: A one-to-six digit number.

  Default: Printing starts with the first byte of each record.

  Note: in a VISAM data set with no regions, the data begins in position 9.

ENDNO
specifies the byte number at which printing is to stop for each data set record. This end byte is printed.

Specified as: A one-to-six digit number.

Default: Printing continues to the last byte of each logical record or until the printer line length is reached, whichever occurs first. (The maximum printer line length is 132 characters.)

PRTSP
specifies the number of spaces to be skipped between lines.

Specified as:

EDIT - line spacing is controlled by a character in the first byte position of each logical record. The control characters may be either a FORTRAN control character (defined by American National Standard FORTRAN, ANSI X3.9-1966) or machine code (see Appendix D), but must be of the same type throughout the data set. The control character in each record is user-supplied.

1 - one space between lines.

2 - two spaces between lines.

3 - three spaces between lines.

Note: when EDIT is specified, HEADER, LINES, and PAGE operands must not be specified.

Default: 1

HEADER
specifies that the first logical record of the data set is to be repeated on each print page as a header line. The first 132 bytes or the entire first record, whichever is smaller, will be used as the header.

Specified as: H

Default: no header is printed.

lines
specifies the number of lines to be printed on a page.

Specified as: from one to four decimal digits; 9999 is maximum.

Default: 54 lines are printed on each page.

PAGE
specifies that pages are to be numbered.

Specified as: P

Default: Pages are not numbered.

ERASE
specifies the disposition of the cataloged data set after the print operation is complete.

Specified as:
ERASE or Y - erase the cataloged data set after the print operation

is complete.
N - do not erase the cataloged data set.

Default: The cataloged data set is not erased.

ERROROPT
specifies the action to be taken if an uncorrectable error is encountered while reading a data set record. This option applies only if the data set to be printed is on tape.

Specified as:
ACCEPT - the error record is accepted.
SKIP - the error record is skipped.
END - the print operation is terminated.

Default: END

form
designates the standard setup region of the SYSUCS data set which contains the defaults for the desired combination of paper forms, print chain, carriage control tape, etc.

Specified as: from one to six characters.

Default: PAPER.

STATION
specifies the remote job entry station to which the printed output is to be directed.

Specified as: one to eight alphameric characters.

Default: ID from Task Common is used.

Note: this parameter can be specified only if the user was assigned this capability when joined to the system.

CHARS
specifies the name of the character arrangement table to be used to load the UCS buffer in the 1403 and 3211, and the translate tables and WCGMS in the 3800 printer. For the 3800 only, up to four character sets may be specified, separated by commas and enclosed within apostrophies.

Specified as: one to six alphameric characters.

Default: P11.

FCB
specifies the name of the forms control buffer (FCB) region in the SYSUCS data set to be used to load the FCB.

Specified as: one to six alphameric characters.

Default: STD6.

PAPER
specifies the paper type to be used for this print request.

Specified as: one to eight alphameric characters.

Default: 1PLY.

COPIES
specifies the number of copies of the data set to be printed. If

the GP operand described below is defaulted, each copy will be one complete image of each page of the data set, and copy one will be completely printed before copy two is begun, etc.

Specified as: a decimal number indicating the number of copies to be printed; maximum is 255.

Default: one copy of the complete data set is printed.

GP (IBM 3800 Printer only)
describes how the printed copies are to be grouped. Each group value specifies the number of copies of each individual page to be printed (in a group) before starting the printing of the next page. Up to eight group values can be specified. No single group value can exceed 255, nor can the sum of those specified exceed 255, or the value of the COPIES operand, whichever is less. Note that the sum of all GP values (if coded) must equal the COPIES value. For example, if COPIES=6(1,3,2) was coded for a three-page data set, the sequence of printing the data set pages would be as follows:

```
page number sequence for group 1:   123
page number sequence for group 2:   111222333
page number sequence for group 3:   112233
```

Default: none.

FLASH (IBM 3800 Printer only)
identifies an overlay (page frame) to be used for printing.

Specified as: a one to eight character alphameric name.

Default: none.

COUNT
specifies (beginning with the first copy printed) the total number of copies to be printed with an overlay.

Specified as: a decimal number between 1 and 255. The maximum value cannot be greater than that specified for the COPIES operand.

Default: if FLASH is specified and COUNT is not specified, all copies (pages) have the overlay printed.

SYSUCS
identifies the user's SYSUCS dataset name to be used to perform the printer setup.

Specified as: a fully qualified data set name.

Default: TSS*****.SYSUCS(0)

BURST (IBM 3800 Printer only)
states whether or not the paper output is to go to the (optional) Burster Trimmer Stacker.

Specified as: Y meaning yes, or N meaning no.

Default: N

COPYMOD (IBM 3800 Printer only)
identifies the data set name that is to be used for modifying the printed copy (copies).

Specified as: a fully qualified data set name.

Default: no copy modifications are made during printing.

TRC
indicates whether or not a table reference character (TRC) is included as the first byte of each output data record (following the optional edit control character).

Specified as: Y meaning yes, or N meaning no.

Default: N

NPRIORITY
specifies a priority to be used by nodes in determining which data sets to transmit next.

Specified as: two decimal characters.

Default: 50

NETACCT
specifies the network accounting number to be used for this transmission.

Specified as: a maximum of eight alphameric characters.

Default: the value assigned when the user was JOINed to the system; if none, the default is blanks.

DELIVER
specifies a sublist of parameters to identify the recipient of the printed or punched output.

prgmrnam - specifies the programmer's name.

Specified as: a maximum of twenty alpha characters.

Default: blanks

room - specifies the programmer's room number.

Specified as: a maximum of eight alphameric characters.

Default: blanks

dept - specifies the programmer's department.

Specified as: a maximum of eight alphameric characters.

Default: blanks

bldg - specifies the programmer's building.

Specified as: a maximum of eight alphameric characters.

Default: blanks

PRTCLASS
specifies the output class for print data sets.

Specified as: one alphameric character.

Default: A

INDEX
specifies the indexing offset to be used when printing on a 3211 printer at a non-TSS node.

<u>Specified as</u>: two decimal characters.

<u>Default</u>: 00

EXTWTR
specifies the name of the external writer to be used at a non-TSS node.

<u>Specified as</u>: a maximum of eight alphameric characters.

<u>Default</u>: blanks

MODTRC
specifies a table reference character for use at a non-TSS node.

<u>Specified as</u>: 0, 1, 2, or 3

<u>Default</u>: none

<u>Initialization</u>: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

<u>Programming notes</u>: the PR macro instruction assigns the request to an independent nonconversational task, to which the system assigns a BSN for possible reference by the user. The specified data set is printed as it appears. Invalid print characters appear as blanks in the output. Data set records containing a read error (or an invalid control character, when the EDIT option is used) are printed in hexadecimal on SYSOUT. When the data set resides on seven track tape, the system makes the character adjustments required to ensure data validity.

If the user specified a form number, the system includes that number in its instructions to the system operator when the printer is readied for operation.

The data set name specified for a BSAM data set may or may not be cataloged. If not, it is placed in the catalog until printing is completed, and then erased, regardless of the ERASE option. If the data set is cataloged, the ERASE option can be used to erase after printing is completed.

When EDIT is specified, the first byte of each logical record is assumed to be the byte following the the control character, which is not printed and is not counted when determining where to begin printing a record.

If the data set to be printed was created via the EDIT or DATA commands, the first byte of each record contains an indicator of the origin of the record. The PR macro instruction translates the byte to a 'C' if the record was entered through a card reader and to a blank if it was entered through the keyboard. Unless the STARTNO operand is specified, this byte is printed as part of the record. If STARTNO is specified as 2 this byte is bypassed.

<u>CAUTIONS</u>: When the user issues the PR macro instruction for a BSAM data set that is defined in the task, the data set definition is released, and the data set is disconnected from the user's task.

| The PR macro is valid for BSAM, VSAM, and VISAM data sets only. It cannot be used to print a member of a VPAM data set. However, a VPAM member can be copied with the CDS command, and then the copy can be printed.

| A BSAM data set must reside on magnetic tape; a VSAM or VISAM data set must not have undefined (format U) records.

| The PR macro instruction should not be used for an uncataloged data set that is awaiting bulk I/O because PR causes the uncataloged data set to be automatically erased.

Return Data: At completion of execution of the PR macro instruction, register 1 contains the address of the batch sequence number assigned to the nonconversational task established by this macro instruction; the low-order byte of register 15 contains one of the following codes:

Code	Significance
00	PR request was accepted.
0C	PR request was not accepted.

Examples: In EX1, the operands are presented as a character string. In EX2, a symbolic address designates the location of the operand string.

```
EX1      PR        'DSNAME1,02,120,1'
EX2      PR        LSTTAG
```

Since EX2 specifies an address, the user has provided the operand string at location LSTTAG. When the macro instruction is executed, the necessary alphameric characters must be available in the string.

PRMPT -- Prompt System to Display a Particular Message (S)

The PRMPT macro instruction requests that the message associated with a particular message ID be displayed at the terminal and calls upon a system control program (the User Prompter) to handle the request.

Standard form:

Name	Operation	Operand
[symbol]	PRMPT	{address of message id|'message id'} {address of response code|response code}, [user response] [,({address of parameter|'parameter'},...)]

L-form:

Name	Operation	Operand
symbol	PRMPT	['message id],[response code], [user response][,('parameter',...)],MF=L

E-form:

Name	Operation	Operand
[symbol]	PRMPT	[{address of message id|'message id'}],
		[{address of response code|response code}],
		[user response]
		[,({address of parameter|'parameter'},...)],
		MF=(E,list)

Note: A symbol is required in the name field of the L-form. The parameters specified in the E-form will overlay those specified in the L-form. The E-form may not specify more operands than are specified in the corresponding L-form.

address of message id
specifies the location of an eight-byte field containing the message identification code (see below), left-aligned and filled out with blanks.

Specified as: Register notation (2 through 12); in the standard form, also as a relocatable expression; in the E-form, also as an RX address.

message id
specifies a unique eight-byte message identification code associated with a message residing in a user-provided message library.

Specified as: The identification code itself enclosed in apostrophes.

address of response code
specifies the address of a location that contains the response code (see below).

Specified as: Register notation (2 through 12); in the standard form, also as a relocatable expression; in the E-form, also as an RX address.

response code
specifies a one-byte code indicating the types of responses, if any, the User Prompter program should expect from the terminal when the message is displayed at the terminal.

Specified as:
N - No response should be expected from the terminal. This option causes the User Prompter to display the message at the terminal and return control to the program containing the PRMPT macro instruction. The user response field is not required with this code.
P - A predefined response should be expected from the terminal. This option causes the User Prompter to display the message at the terminal, read a user response from the terminal, and then compare the user response to an expected response that was predefined in the message library. If a matching response is received, a code attached to the predefined response in the library is returned to the caller in the user response field defined by the PRMPT macro instruction. If a matching response is not found, the user is prompted with all of the predefined responses to terminal responses. For conversational tasks, if the next response is also improper, the user response field is either set to zero and an error is indicated in register 15, or, if the response is to be defaulted, register 15 will be set to zero. Control is then returned to the

user's program. (See "Response Message" in Command System User's Guide for the means of defining the responses.)

R - Return message text. This option causes the user prompter to return the text of the message, with the inserts filled in, to the program containing the PRMPT macro instruction. Upon successful return from the PRMPT macro, register 1 contains the address of the text and register 0 contains the length of the text. The user response field is optional on this call; if specified it will contain the address of the text preceded by a word containing its length. Note; message text is returned only if the return code is X'00' or X'14'; on all other return codes the contents of registers 0 and 1 are unpredictable.

U - An unpredictable response other than those defined in the message file, such as a string of information, should be expected from the terminal. This option causes the User Prompter to display a message at the terminal and then read an undefined response from the terminal. For example, the message might be "Enter User ID", to which the response would be an actual user identification. In this case the User Prompter places in the user response field a pointer to the response read from the terminal. The byte preceding the response string must contain the length of that string (255 bytes maximum length).

user response
A one-word field in which the User Prompter indicates the type of user response to a message. For predictable responses, a unique predefined response code, indicating which of the possible predefined responses has been entered, is placed in the field, right-justified, by the User Prompter. For unpredictable responses, a pointer to the response string is placed in the field.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also in register notation; in the E-form only, also as an RX address.

address of parameter
specifies the location of a parameter (see below) that modifies the message being displayed. See Examples EX4 and EX5.

Specified as: In the standard form and L-form, a relocatable expression; in the standard and E-form, also in register notation (2 through 12); in the E-form only, also as an RX address. If a relocatable expression is used, the string to which it points must be supplied by the caller's program and immediately preceded by a byte containing the length of the string. The number of parameters and parameter addresses cannot exceed 20.

parameter
specifies information that is to be used to complete or alter the message being displayed at the terminal. The information is

substituted for variables ($1, $2,...) in the message text. The number of parameters or parameter addresses cannot exceed 20. Parameters are separated by commas.

Specified as: The parameter itself enclosed in apostrophes. If more than one parameter is specified, each must be enclosed in apostrophes and separated by commas. Parameters and parameter addresses can be intermingled.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The User Prompter is a centralized facility for storing and displaying messages, providing explanations of messages, and handling responses. This facility is available to both the system and user programmers. The message file used by the User Prompter is the system message file. The user may add to or alter the message file, using the procedure defined in Command System User's Guide, GC28-2001. All predefined responses set up in the message file must be preceded by a unique identification code.

Failure to supply variable message data parameters, if requested by the message in the message file, will result in the User Prompter inserting three asterisks (***) in place of the missing parameters.

Explanations of system messages displayed at the terminal can be requested by use of the EXPLAIN command (see the Command System User's Guide for further details).

Return Data: When control is returned from the User Prompter to the program containing the PRMPT macro instruction, register 15 contains one of the following hexadecimal return codes:

Code	Meaning
00	Successful completion.
04	an I/O error has occurred; either the message file SYSLIB (SYSMLF) is not open, or a SYNAD/EODAD occurred when the replace code is P.
10	Message filtered by user.
14	Insufficient output buffer space provided. Message truncated.
18	Explanation not found in message explanation routine. 'No explanation available' displayed at terminal.
1C	Matching response not found in message response routine.
20	Response code not specified as N, P, or U.
28	Attention interruption occurred during I/O operation.
2C	Too many reference messages in a chain - reference looping.
30	Invalid response code in response line of SYSMLF.
34	Response message not in SYSMLF (response code is P).

L- and E-Form Use: An example of L- and E-form use is:

```
SUE        PRMPT   MSGIDADR,N,,MF=L
             .
             .
             .
           PRMPT   'MSGIDB',P,RESP,MF=(E,SUE)
             .
             .
             .
MSGIDADR   DC      C'MSGIDA'
             .
             .
```

```
RESP      DS      F
```

When the E-form of this macro instruction is executed, the message (that is, the message whose ID is MSGIDA) is replaced in the parameter table generated by the L-form macro expansion, by the message specified in the E-form (that is, the message whose ID is MSGIDB). In addition a predictable response will be expected by the User Prompter instead of no response at all.

Examples: The user code, the message written at the terminal, and a user response, are in the examples below. It is assumed that BREVITY=S so that the message identifications are not printed.

In EX1, the user has defined the message MSGAA LOOP A executed, no response from the terminal is expected and control is returned to the user's program after the message is displayed.

```
EX1:    User Program:                  PRMPT 'MSGAA',N
        Message to Terminal:           LOOP A EXECUTED
```

EX2 expects predefined responses to be entered at the terminal. When the user responds by entering VS, the PRMPT routine searches a list of predefined responses (via the User Prompter) to find a match. A unique code, identifying which of the possible predefined responses the user has entered, is stored in the fullword parameter (RESP) in the user's program and control is returned to the program.

```
EX2:    User Program:                  PRMPT 'MSGAB',P,RESP
                                         .
                                         .
                                         .
                        RESP           DS      F
        Message to Terminal:           ENTER DSORG
        Response from Terminal:        VS
```

EX3 expects unpredictable responses to be entered at the terminal. When the user responds by entering his user ID, the routine determines if a valid ID has been entered, and if it is valid, places a pointer to the area in which the response is stored, in the user response option field (IDCODE).

```
EX3:    User Program:                  PRMPT 'MSGAC',U,IDCODE
                                         .
                                         .
                                         .
                                       DC      AL1(L'IDCODE)
                        IDCODE         DS      F
        Message to Terminal:           ENTER USER ID
        Response from Terminal:        SMITH3
```

EX4 requires no user response; variable message data parameters ('100' and the string at PARADD1) are inserted into the message contained in the message file and the completed message is written to the terminal. Thus, if the message is recorded in the message file as, LINE $1 IN REGION $2 DOES NOT EXIST, the variable entries $1 and $2 are replaced by the parameters provided via the operands of the PRMPT macro instruction as indicated in EX4.

```
EX4:    User Program:                  PRMPT 'MSGAD',N,,('100',PARADD1)
                                         .
                                         .
                                       DC      AL1(5)
                                         .
                                         .
```

```
                    PARADD1       DC    C'AD123'
          Message to Terminal:    CZSEB LINE 100 IN REGION
                                        AD123 DOES NOT EXIST
```

In EX5, register notation is used in the response code and parameter operands. No response is expected from the terminal and the parameters A, B, and C replace the variable entries $01, $02, $03 in the message.

```
EX5:      User Program:               LA 3,RECODE
                                      LA 4,CODELOC
                                      PRMPT 'MSGAE',(3),,('A',BCODE,(4))
                                       .
                                       .
                                       .
                                      DC      AL1(BCODE)
                    BCODE             DC      F'B'
                    RECODE            DC      F'N'
                    CODELOC           DC      F'C'
          Message to Terminal:        PARAMETER VALUES ARE A,B,C
```

PU -- Punch a Data Set (S)

In nonconversational mode, the PU macro instruction causes a specified VISAM or VSAM data set to be punched onto cards on a high-speed punch and, optionally, to be erased from the catalog when punching is finished. Any contiguous field of up to 80 bytes can be punched from each input record of an EBCDIC data set. The specified data set is punched as it stands, with no code conversions.

Note: Up to 160 bytes per card can be punched in a special column binary format, where bits 0 and 1 of each byte are ignored.

Standard form (See "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	PU	{ address of operand string 'data set name,,[start byte],[end byte], [stacker option],[{ERASE|Y|N}] [,card form]' }

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	PU	'data set name,,[start byte],[end byte], [stacker option],[{ERASE|Y|N}][,card form]',MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	PU	MF=(E,list)

Note: Two commas must be coded between the data set name and start byte operands.

address of operand string
specifies the address of the first operand in the operand string. See "Operand Strings" in Part II, Section 1.

Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified in the macro operand as a character string enclosed in apostrophes, as shown.

data set name
specifies the name of the data set to be punched. The data set name must previously have been defined by a DDEF macro instruction or command, or must be in the catalog.

Specified as: The fully qualified name of a non-partitioned data set or of a nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number). See "Data Set Name" in Part II, Section 1.

start byte
specifies the byte number at which punching is to start for each data set record.

Specified as: A number.

Default: Punching starts with the first byte of each record.

end byte
specifies the byte number at which punching is to stop for each data set record.

Specified as: A number.

Default: Punching continues to byte 80 (or, in binary, to byte 160) or to the end of the record, whichever occurs first.

stacker option
specifies the stacker select or edit option:

Specified as:

1 -	indicates pocket number P1
2 -	indicates pocket number P2
3 -	indicates pocket number P3
EDIT -	indicates that the first byte of each logical record in the data set contains a control character for stacker selection. This control character is user-supplied and may be in FORTRAN or machine code, but must be in the same code throughout the data set. (See Appendix D.)

Default: 1

ERASE|Y|N
specifies the disposition of the cataloged data set after the punch operation is complete.

Specified as:

ERASE or Y -	erase the cataloged data set after the punch operation is complete.
N -	do not erase the cataloged data set.

Default: N

Note: If ERASE or Y is specified for a shared data set that is currently being used by another user, a diagnostic message is issued and the data set is not erased.

card form
specifies the punch card form number of the cards to be used for this punch request.

Specified as: One to six characters.

Default: The installation's standard card form, as established at system generation, is used.

Note: The system does not check the validity of the card form specified; therefore, it is the responsibility of the user to convey the meaning of the specified card form to the system operator.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: When the user executes a PU macro instruction for a data set defined in his task, the data set is released and disconnected from the user's task. The PU macro instruction processes data sets that were created by using either the virtual sequential or virtual index sequential access method. The data set name must be in the catalog; if not, the user's task is terminated. The ERASE option can be used to erase the data set after punching is completed.

If a data set to be punched was created via the DATA command, the first byte of each record contains an indicator for the origin of the record. Unless the start byte operand is specified, this byte is punched as part of the record upon issuance of the PU macro instruction. In such a case, if the record was originally entered through a card reader, the indicator byte will be punched as a C; if it was entered through a terminal, the byte will be punched as a blank character. When the start byte operand is specified as 2 or greater, the indicator byte is bypassed and is not included as part of the punched record.

| Since the DATASET command prefixes a line number automatically to each record of a VISAM data set read from cards, any VISAM data set that is to be read from cards should not contain line numbers. Therefore, if an existing VISAM line data set is to be punched on cards and later
| recreated by reading those cards with a DATASET command, the user should be careful to punch out the stored VISAM data set without including line numbers.

Invalid characters appear as blanks when EBCDIC records are punched. If a read error occurs, the record in question is not punched, but is written in hexadecimal on SYSOUT.

Return Data: At completion of execution of the PU macro instruction, register 1 contains the address of the batch sequence number assigned to the nonconversational task established by this macro instruction; the low-order byte of register 15 contains one of the following codes:

Code	Significance
00	PU request was accepted.
All other codes	Register 15 contains a two-byte system message number.

Examples: In EX1, the operands are presented as a character string. In EX2, a symbolic address designates the operand string.

```
EX1     PU        'DSNAM2,,020,99,,ERASE'
EX2     PU        CDTAG
```

Since EX2 specifies an address, the user has provided the operand string at location CDTAG. When the macro instruction is executed, the necessary alphameric characters must be available in the string.

PUT -- Include a Record in an Output Data Set (R)

The PUT macro instruction (for VSAM, VISAM, and QSAM) can be specified in either locate mode or move mode. In locate mode, the PUT macro instruction places in register 1 the address of an output buffer. The user should subsequently construct at that address the next record to be incorporated in an output data set. In move mode, the PUT macro instruction moves a record from a user-specified area in virtual storage into an output buffer so that the system may include the record in the output data set.

Name	Operation	Operand
[symbol]	PUT	dcb address,record address

dcb address
: specifies the address of the data control block opened for the data set being created or extended.

| Specified as: Register notation (1 through 12), or an RX address.

record address (for move mode only)
: specifies the address of the logical record to be moved into the buffer.

| Specified as: Register notation (1 through 12), or an RX address.

Initialization: The address of a save area must be placed in register 13 before execution of this macro instruction.

CAUTIONS:

For VISAM and QSAM: Any exceptional condition resulting from the execution of a PUT macro instruction causes control to be passed to the user's synchronous error exit (SYNAD) routine. In this case, the general register and the exceptional condition fields in the data control block are set as shown in Appendixes B and F.

For VISAM: The PUT macro instruction may only be used with data sets that have been opened for OUTPUT. If more than one DCB has been opened for a non-shared data set within a task, the PUT will be ignored.

Return Data: In locate mode (for QSAM), the address of the next buffer segment large enough to hold the next logical record is returned in register 1 after execution of this macro instruction.

Programming Notes:

For VSAM: It is the user's responsibility to store the length of each format-U record in the DCBLRE field of the data control block before issuing the PUT. This length must be a multiple of 4096 bytes.

For VSAM and VISAM: For format-V records, each record must begin with a four-byte length field. The user must place the length of the record in the low-order three bytes of that four-byte field before issuing a PUT macro instruction. The high-order byte must contain binary zero. Rules for sharing data sets are given in Appendix K.

For QSAM: Before executing this macro instruction, the user must place the length of the record in the logical record length field (DCBLRECL) of the data control block according to the format of the logical records, as follows:

1. Format-F records: the logical record length is taken from DCBLRECL. This field should not be altered after the DCB is opened or an incorrect length block will be written, causing abnormal termination.

2. Format-U records: the actual record length must be known before the record is constructed, and must be placed in the DCBLRECL field. Abnormal termination occurs if DCBLRECL is greater than DCBBLKSI.

3. Format-V records: For locate mode, the actual record length must be placed in the DCBLRECL field, or an estimated record length (not less than the actual record length) must be placed in the DCBLRECL field. If the estimated record length in DCBLRECL is greater than DCBBLKSI, abnormal termination occurs.

   For move mode, the length of each logical record determines the amount of buffer space needed. If the length is greater than DCBBLKSI, abnormal termination occurs. The record address provided by the user is returned in register 1. Because QSAM does not support the substitute mode PUT, this feature (return of the area address) provides the compatibility that allows move mode to be used to execute programs originally written for OS (where substitute mode PUT may be used).

Examples: In the following example, the use of a move mode PUT is shown. The address of the next logical record to be processed is returned in register 1 following the locate mode GET macro instruction. The record is part of an input data set associated with the data control block INDCB. After the record is processed within the input buffer, the move mode PUT is used to move the record to an output buffer. Before the PUT is executed, the address of the record is placed in register 0. The branch instruction is used to reenter the processing loop.

```
INVEN    GET    INDCB
          .
          .
         LR     0,1
         PUT    OUTDCB,(0)
|        B      INVEN
```

| PUTSEG -- Put a Page into a Disconnected Segment Group (O)

| The PUTSEG macro instruction will put a page from a virtual storage
| buffer into an existing disconnected segment group.

| L-form

Name	Operation	Operation
symbol	PUTSEG	[DNAME=,] MF=L

E-form

Name	Operation	Operation
[symbol]	PUTSEG	[DNAME=,] ADDRESS=,BUFFER=,MF=(E,list)

Standard form

Name	Operation	Operation
[symbol]	PUTSEG	DNAME=,ADDRESS=,BUFFER=

Note: All operands are keyword.

DNAME=
specifies the eight character EBCDIC name of the disconnected segment group.

Specified as: name enclosed within apostrophies: in the E or standard form only, as the address of DNAME expressed as a relocatable expression, RX address, or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the disconnected segment group name.

Default: none.

CAUTION: any user specified DNAME beginning with SYS will be rejected by the system.

ADDRESS=
specifies the relative page address of the disconnected segment group page.

Specified as: in the E or standard form only, the address of a word containing the relative page address expressed as a relocatable expression, RX address, or register notation.

Default: none.

BUFFER=
specifies the page aligned virtual storage address of the page to be put into the disconnected segment group.

Specified as: in the E or standard form only, the address of a word containing the virtual storage address expressed as a relocatable expression, RX address, or register notation.

Default: none.

Return codes: upon return from execution of PUTSEG, register 15 will contain a return code as follows:

Code	Meaning
X'00'	successful
X'08'	DNAME invalid
X'12'	segment not available to user class
X'16'	invalid address
X'40'	system error

Register 1 will contain the address of the Nameseg Parameter List.

| Note: the DSECT CHANSG covers the Nameseg Parameter List.

| Example:

```
                •
                •
                •
            PUTSEG   DNAME=DOC1,ADDRESS=DOCPADDR,BUFFER=DOCPMOD
                •
                •
                •
| DOCPADDR  DC  F'4096'    address relative to start of disconnected group
| DOCMOD    DC  A(BUFPAG)     address of page aligned buffer
| DOC1      DC  CL8'DOCCHNG'  name of disconnected segment group
```

| In the example above, the contents of the second page in the discon-
| nected group 'DOCCHNG' will be replaced with the contents of BUFPAG.

PUTX -- (VSAM) Replace a Sequential Logical Record (R)

The PUTX macro instruction (for VSAM) allows the user to return an updated logical record to an input data set. (See below for the QSAM PUTX description.)

Name	Operation	Operand
[symbol]	PUTX	dcb address

dcb address
specifies the address of the data control block opened for the data set being processed.

Specified as: Register notation (1 through 12), or an RX address.

Initialization: The address of a save area must be placed in register 13 before execution of this macro instruction.

Programming Notes: The PUTX macro instruction can only replace a record that was located by a locate-mode GET macro instruction. The data control block must be opened for the UPDAT mode while using PUTX. The user must not change the length of the record during the replacement process.

Rules for sharing VSAM data sets are given in Appendix K.

PUTX -- (QSAM) Include a Logical Record in an Output or Updated Data Set (R)

The PUTX macro instruction (for QSAM) causes the next logical record contained in a buffer area of an input data set to be written as the next sequential logical record of an output or updated data set. This macro instruction may be specified in either update mode or output mode. In update mode only, the output and input data sets are one and the same, and only the first operand is required. In output mode, two different data sets are used, requiring that both operands be specified. (See above for the VISAM PUTX discription.)

Name	Operation	Operand
[symbol]	PUTX	output dcb address [,input dcb address]

output dcb address
specifies the address of the data control block opened for the out-

put data set. The output data must be opened for UPDAT if the update mode is used or it must be opened for OUTPUT if the output mode is used.

Specified as: Register notation (1 through 12), or an RX address.

input dcb address
specifies the address of the data control block opened for the input data set. This operand is required in the output mode.

Specified as: Register notation (0 or 2 through 12), or an RX address.

CAUTIONS: The following cautions apply:

- The data set must reside on a direct access device.
- For blocked-format records, if any logical record in a block has been returned by a PUTX macro instruction, the control program will not write the entire block back to the data set until all the logical records in that block have been processed.
- The length of the block and the length of each logical record cannot be altered.
- Additional logical records cannot be inserted in the block, nor can existing logical records be deleted from the block.

Programming Notes: Any exceptional condition resulting from the execution of a PUTX macro instruction causes control to be passed to the user's synchronous error exit (SYNAD) routine.

The PUTX macro instruction must always be preceded by a locate mode GET macro instruction. This GET macro instruction must specify the same data set as specified by an update mode PUTX macro instruction, or it must specify the data set that is used as input by an output mode PUTX macro instruction.

Since the update mode uses only a single data set, the user need only issue a PUTX for those logical records that are to be updated. Those records that have not changed can be bypassed, and thereby remain unchanged, simply by issuing two successive GET macro instructions (see the example below.)

In output mode two distinct data sets are used and a PUTX is required for each logical record that is to be included in the output data set being created. Abnormal termination occurs if these requirements are violated.

Compatible Record Formats And Buffering Techniques: Normally, when the PUTX macro instruction is used, data sets with the same record formats and buffering techniques are processed together. However, the control program supports certain variations from this procedure. Figure 15 indicates which combinations of input and output record formats are acceptable.

	Output data set (move mode)				
Input data set (locate mode)	to U[1]	to F[2]	to FB[2]	to V[3]	to VB[3]
from U	S	---	---	---	---
from F	S	S	S	---	---
from FB	S	S	S	---	---
from V	S	---	---	S	S
from VB	S	---	---	S	S

where:

---	indicates unacceptable record format combination
S	indicates acceptable record format combinations (only simple buffering is supported by TSS)
U	indicates format-U records
F	indicates format-F records
FB	indicates format-F blocked records
V	indicates format-V records
VB	indicates format-V blocked records

[1]The block size for the format-U output data set must be as large as the largest logical record size of the input data set.
[2]The logical record size for format-F and -FB records must be the same for both data sets.
[3]The maximum logical record for format-V and -VB records must correspond.

Figure 15. Acceptable record formats for QSAM and the PUTX macro instruction

Example: The following example shows the use of a PUTX macro instruction when records are being updated. The locate-mode GET macro instruction provides the address of the next record to be updated. The PUTX macro instruction, after processing the record, returns it to the data set. The conditional branch instruction tests the condition code. If the record is to be updated, the next sequential instruction is executed; if it is not to be updated, another GET macro instruction is issued to locate the next record. The unconditional branch following the PUT macro instruction is used to reenter the processing loop. When all input records have been processed, the EODAD routine is given control.

```
    .
    .
    .
LLS GET DCBA
    .
    .
    .
    BH LLS
    .
    .
    .
    PUTX DCBA
    .
    .
    .
    B LLS
```

RAE -- Restore and Enable (O)

The RAE macro instruction restores the prior inhibit state of the task monitor and sets the problem program in the enabled state.

Name	Operation	Operand
[symbol]	RAE	status byte address

status byte address
    specifies the address of a one-byte area previously used by an SAI macro instruction for saving the prior task monitor inhibit status.

    Specified as: Register notation (2 through 12), or a relocatable expression.

    Default: The restore function is not executed but the problem program is set in the enabled state.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: There are separate enable indicators for privileged and nonprivileged programs. The RAE macro instruction sets the appropriate indicator, depending on the attributes of the program being assembled. A nonprivileged program cannot inhibit dispatching to privileged programs.

READ -- (VISAM) Read a Selected Logical Record (S)

The READ macro instruction (for VISAM) moves a selected logical record from an input data set to a user-specified area. The user selects the record by providing either the record key or the retrieval address. (See below for the BSAM READ description.)

Standard form:

Name	Operation	Operand
[symbol]	READ	decb name,type,dcb address,work area address, key field address

L- and E-forms:

Name	Operation	Operand
[symbol]	READ	decb name,type,[dcb address],[work area address] [,key field address],MF={L|E}

Note:: A symbol is required in the name field of the L-form. If an operand is not specified with the L-form, it must be provided with the E-form.

decb name
: specifies the name to be assigned to the data event control block (DECB) constructed as part of the expansion of this macro instruction. The DECB is illustrated in Appendix B, Figure 18.

  Specified as: One to eight alphameric characters, the first of which must be alphabetic; in the E-form only, register 1 may also be specified.

type
: specifies the type of READ operation.

  Specified as:
  KX - read according to specified key ("Read Exclusive"), permitting no other user sharing the data set to gain access to the record until the current user has released the record. The record must be released by the RELEX macro instruction or by a subsequent WRITE macro instruction referring to the same data control block.

  KY - read according to specified key.

  KZ - read according to specified retrieval address.

dcb address
: specifies the address of the data control block opened for the data set being processed.

  Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also as register notation (2 | through 12); in the E-form only, also as an RX address.

work area address
: specifies the address of an area in virtual storage into which the | record is to be placed.

Note: The area must be large enough to contain the largest expected record.

Specified as: Same as the dcb address operand.

key field address
: specifies the address of the field containing either the record key for a READ (type KY or KX) or the retrieval address for a READ (type KZ). The retrieval address is a four-byte field, beginning on a word boundary, that is in the data control block and may be accessed using the DCBD macro instruction and the name DCBLPA.

  Specified as: Same as the dcb address operand.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix F). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: A READ by retrieval address (type KZ) must not be used with a shared data set.

Exceptional conditions resulting from the execution of a READ macro instruction cause control to be passed to the user's synchronous error exit (SYNAD) routine; these conditions include "key not found", "key greater than last key on data set," and "invalid retrieval address." For these conditions, the general registers and the exceptional condition fields in the data control block are set as shown in Appendixes B and F.

Programming Notes: READ (type KY) imposes a page-level read interlock on the pages containing the record to be read, whereas READ (type KX) imposes a page-level write interlock and releases a page-level read interlock. As the record pointed to by the data control block shifts within the data set, page-level interlocks are released from pages no longer being used.

Rules for sharing VISAM data sets are given in Appendix K.

L- and E-Form Use: The L-form macro instruction results in a macro expansion consisting of only a parameter list (DECB). The format of the DECB is described in Appendix B.

The E-form macro instruction results in a macro expansion consisting of only executable instructions. The E-form macro instruction uses the DECB built for it by the L-form macro instruction. Only MF=E should be specified for the MF operand of the E-form, because it is the DECB symbol that names the parameter list of the L-form.

Any E-form parameter replaces the corresponding parameter in the DECB.

READ -- (BSAM) Read a Block (S)

The READ macro instruction (for BSAM) transmits a block of data from an input data set to a user-specified virtual storage area. To allow overlap of the I/O operation with processing, the READ macro instruction returns control to the user's program before the input operation is complete. (See above for the VISAM READ description.)

The READ macro instruction may be used to read backwards from magnetic tape.

Standard form:

Name	Operation	Operand
[symbol]	READ	decb name,type,dcb address,work area address [,length]

L- and E-form:

Name	Operation	Operand
[symbol]	READ	decb name,type,[dcb address], [work area address][,length],MF={L|E}

Note: The name field is required with MF=L. If an operand is not specified with the L-form, it must be specified with the corresponding E-form of the macro instruction.

decb name
: specifies the name to be assigned to the data event control block (DECB) constructed as part of the expansion of the macro instruction. The DECB is illustrated in Appendix B, Table 18.

  Specified as: One to eight alphameric characters, the first of which must be alphabetic; in the E-form only, register 1 may also be specified.

type
: specifies the type of READ operation.

  Specified as:
  SF - sequential forward reading of a physical sequential data set.

  SB - sequential backward reading from a magnetic tape.

dcb address
: specifies the address of the data control block opened for the data set being processed.

  Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also in register notation (2 through 12); in the E-form only, as an RX address.

work area address
: specifies the address of an area in virtual storage into which the block of data is to be read. If SF is written in the type field, this operand specifies the address of the first byte of the area; if SB is written, the address of the last byte is specified.

  Specified as: Same as the dcb address operand.

length
: specifies, for format-U records, the number of bytes to be transmitted. If this parameter is specified for format-F or format-V records, it is ignored. For format-F and -V blocks, the length is obtained from the BLKSIZE field of the data control block. This operand is required for format-U records.

  Specified as: 'S' (Note the apostrophes), in which case the program attempts to read the maximum size specified in the data control block (the largest block size that can be specified is 32,767 bytes); as a relocatable expression; in the standard and E-form, also in register notation (2 through 12).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Abnormal termination occurs if:

1. The specified data control block was not validly opened.

2. The specified DECB is already in use by a previous READ or WRITE macro instruction; that is, it has not been checked by a CHECK macro instruction.

3. An attempt is made to issue a READ macro instruction that causes the number of unchecked READ and WRITE macro instructions to exceed the DCBNCP parameter specified in the data control block.

4. An attempt is made to read an OUTPUT data set.

Programming Notes: The READ macro instruction returns control to the user's program before the transmission of data has been completed. To determine whether the read operation is completed, it is necessary to issue the CHECK macro instruction before using the data transferred into the specified area. The user may determine when he has reached the end of the last file on the last volume by checking if DCBMW=X'02'. A DCBD macro instruction must be included in the program and the DSECT generated by the DCBD must be linked to the data control block, either by loading a base register with the address of the DCB and issuing a USING statement for the DSECT CHADCB, or by issuing:

```
CLI DCBMW-CHADCB+DCB1,X'02'
```

This adds the displacement of DCBMW into CHADCB to the address of the data control block DCB1.

After an I/O error, exit is made to the SYNAD exit, if one is specified in the DCB, and the 8 sense bytes used to store information pertaining to disk or tape devices are saved in DECB sense bytes 0 through 7. DECB byte 1 should be set to X'02' to have these bytes put into the DECB. (See Appendix B, Table 3.) If no SYNAD exit is specified, abnormal termination occurs.

The DECB employed for a read operation must not be reused or modified until the CHECK macro instruction is issued.

After a READ operation has been checked, the length of a format-U block or a truncated block in a fixed-length blocked data set can be determined from the residual count field of the Channel Status Field in the DECB. The residual count is subtracted from the block length to determine the length of block still to be read (see example below). The number of READ operations may not exceed that specified in the DCBNCP field in the data control block without using a CHECK macro instruction.

Example: If the user specifies SYNAD=ERR1 in his DCB and if his DECB is named DECB0, the length of the block yet to be read may be determined as follows:

```
ERR1    CLI     DECB0+37,X'0'    IS INCORRECT LENGTH INDICATOR SET
        BNE     CONTIN           NO, CONTINUE
        LH      11,DCB1+48       YES, GET BLOCK LENGTH
        LH      7,DECB0+38       GET RESIDUAL COUNT
        SR      11,7             GET BLOCK LENGTH TO BE READ
        .
        .
        .
```

A data set written on a direct access device with track overflow specified must have track overflow specified for all read operations referring to that data set. If a track selected by a READ macro instruction is flagged as defective, the alternate track is automatically selected. For any device, the operator is notified if intervention is required to complete the operation.

If a READ (type SB) macro instruction is issued for a format-V record, the address of the first byte of the record can be calculated by subtracting the count field in the channel status word from the maximum block size and subtracting the result from the work area address.

If the length specified in the READ macro instruction for format-U records is less than the length of the actual physical record, the extra bytes of data are not transmitted.

The first four bytes on format-V blocks contain control information that is passed with the record when read. The area specified by the work area address operand must be large enough to accommodate the maximum record size.

L- and E-Form Use: The L-form macro instruction results in a macro expansion consisting of only a parameter list (DECB). The format of the DECB is described in Appendix B.

The E-form macro instruction uses the DECB built for it by the L-form macro instruction. Any E-form parameter replaces the corresponding parameter in the DECB.

Example: In EX1, a DECB, with the symbolic name ADECB, is produced as part of the in-line expansion. It indicates that forward reading of the next block in the data set associated with data control block INDCB should be performed, using area INAREA.

```
EX1     READ     ADECB,SF,INDCBA,INAREA
```

In example EX2, the type operand indicates backward reading of a block of records from the data set associated with the data control block INDCB. For format-U records, 100 bytes are transmitted; after the operation, the bytes extend from INAREA+99 to INAREA. For records other than format-U, the length parameter is ignored.

```
EX2     READ     ADECB,SB,INDCB,INAREA+99,100
```

| ## REDTIM -- Read Elapsed Real Time (O)

| The REDTIM macro instruction provides the system time in
| microseconds.

Name	Operation	Operand
[symbol]	REDTIM	

| Note: There are no operands.

| Execution: the System/370 (hardware) real time clock is read (and
| adjusted) to give the present time in microseconds.

| Return Data: The resulting double-precision fixed-point number is re-
| turned in registers 0 and 1.

| Example: Suppose you want to find the date and time. You might write:

```
|       NAME   REDTIM
```

REL -- Release Data Set or Remove Job Library From Program Library List (S)

The REL macro instruction may be used to cancel a preceding definition for either a public or private data set, or to release the input/output devices associated with a private data set. It may also be used to release one or all data sets of a given concatenation, and to remove JOBLIB from the user's program library list.

Standard form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	REL	{address of operand string| 'data definition name,[data set name], [{SCRATCH|HOLD}][,{SCRATCH|HOLD}]'}

L-form (see "Operand Strings" in Part II, Section I):

Name	Operation	Operand
symbol	REL	'data definition name,[data set name,] [{SCRATCH|HOLD}][,{SCRATCH|HOLD}]',MF=L

Note: A symbol is required in the name field of the L-form.

E-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	REL	{address of operand string | 'data definition name,[data set name], [{SCRATCH|HOLD}],[{SCRATCH|HOLD}]'} ,MF=(E,list)

Note: Two positions are supplied for HOLD and SCRATCH so that the user can both scratch the volume and hold the device.

address of operand string

specifies the address of the first operand in an operand string.

Specified as: A relocatable expression, or register notation (2 through 12). Note that the operand string can also be specified in the macro operand as a character string enclosed in apostrophes, as shown.

data definition name

specifies a data definition name previously issued by a DDEF macro instruction or command. This name identifies the data set to be released. The name may specify a job library and may also specify that the library data set name is to be removed from the program library list.

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic).

data set name

specifies the name of one sequential data set in a concatenated series.

Specified as: The fully qualified name of a nonpartitioned data set or of a nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number). See "Data Set Name" in Part II, Section 1.

Default: All data sets concatenated with the named data set are released.

SCRATCH

specifies that the volume(s) will not be re-used by the current task and may be dismounted or made available to other tasks. This option is relevant only to private volumes.

182.2

Specified as: SCRATCH

Default: See HOLD operand.

HOLD
specifies that the device(s) will be re-used by the current task, and that reservation(s) made by a non-conversational SECURE command must be retained.

Specified as: HOLD.

Default: Defaults are indicated by the action chart below.

Action chart for REL macro:

	SCRATCH	HOLD	volume disposition	device disposition
	-------	----	default according to task mode	default according to task mode
(1)	SCRATCH	----	logically dismounted	released, reservation dropped
(2)	-------	HOLD	remains mounted and reserved	remains available; reservation retained
	SCRATCH	HOLD	logically dismounted	remains available reservation retained

(1)=default for conversational task
(2)=default for non-conversational task

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: When a data set has been released, it cannot be referred to again until another DDEF macro instruction or command defining that data set is issued.

Return Data: After execution of the REL macro instruction, the low-order byte of register 15 contains one of these hexadecimal codes:

Code	Significance
00	Successful completion
04	Defaulted or invalid ddname
08	Attention interruption occurred
0C	Reserved data definition name specified - not permitted
10	Undefined data definition name
14	Uncataloged on public storage
18	Undefined data set name
20	Invalid input
24	Unable to unload all modules loaded from library

Examples: In EX1, a character constant is given for the data definition name DD1. In EX2, the address of the same name is given.

```
EX1      REL       'DD1'
EX2      REL       RELTAG
```

RELEX -- Release Read Exclusive Record (R)

The RELEX macro instruction (for VISAM) makes a record of a shared data set available to other users after the record has been read with a READ exclusive (type KX) macro instruction.

Name	Operation	Operand
[symbol]	RELEX	dcb address

dcb address
  specifies the address of the data control block opened for the data set being processed.

| Specified as: Register notation (1 through 12), or an RX address.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Exceptional conditions resulting from the execution of a RELEX macro instruction cause control to be passed to the user's synchronous error exit (SYNAD) routine. In this case, the general registers and the exceptional condition fields in the data control block are set as shown in Appendixes B and F.

Programming Note: Rules for sharing VISAM data sets are given in Appendix K.

RELSE -- Release an Input Buffer (P)

The RELSE macro instruction (for QSAM) causes the remaining contents of the current input buffer to be ignored. The next GET macro instruction will retrieve the first logical record from the next input block.

Name	Operation	Operand
[symbol]	RELSE	dcb address

dcb address
  specifies the address of the data control block opened for the input data set.

| Specified as: Register notation (1 through 12), or an RX address.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction is the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: A RELSE macro instruction is ignored if used with unblocked records, or if all records in a buffer have been processed, or if it immediately follows another RELSE macro instruction.

A RELSE issued before the first GET of the data set is ignored.

Programming Notes: If a data set is being read backwards, the RELSE causes the same results as in forward reading.

RELSEG -- Release Reserved Segment Group (O)

The RELSEG macro instruction releases a reserved segment group. The name and length are forgotten by the system. Space allocated within the reserved segment group remains addressable.

Macro-form

Name	Operation	Operand
[symbol]	RELSEG	RNAME=reserved segment name

RNAME=
specifies the eight character EBCDIC name of an existing reserved segment group.

Specified as: Reserved segment name enclosed within apostrophies, the address of RNAME is expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the reserved segment name.

Default: None.

Return Data: On return from RELSEG, register 15 will contain a return code describing the success of the operation.

Return Codes

00	Successful
04	RNAME invalid
12	Segment group not available to user class
40	System error

CAUTION: Any user specified RNAME beginning with SYS will be rejected by the system.

Examples:

```
1.  RL1      RELSEG      RNAME='RNAME1'

2.  RL2      RELSEG      RNAME=RNM
    RNM      DC          CL8'RNAME'

3.  RL3      RELSEG      RNAME=(3)
```

Execution of example 3 assumes that the reserved segment name is contained in registers 3 and 4.

RETURN -- Return to a Program (O)

The RETURN macro instruction, when issued in a called program, returns control to the calling program. The function of this macro instruction depends upon how it is used:

1. If the first program to receive control from the system issues a RETURN macro instruction to return control to the system, the effect is the same as if an EXIT macro instruction had been issued.

2. A program that follows Type I linkage conventions and is given control by the CALL macro instruction, can return control to the program that called it by issuing a RETURN macro instruction.

The RETURN macro instruction may also be used to restore the contents of the registers of the calling program that were saved by the SAVE macro instruction issued in the called program.

Name	Operation	Operand
[symbol]	RETURN	[(first register[,last register])][,T][,RC=code]

first register, last register
: specifies the range of registers whose contents are to be restored from the save area in the calling program.

  Specified as: The operands are written as decimal numbers such that, when inserted in an LM instruction, they cause the contents of the desired registers in the range from 14 through 12 (that is, 14, 15, and 0 through 12) to be restored.

  The contents of registers 14 and 15, if specified, are restored from words 4 and 5 of the save area: the contents of registers 0 through 12, if specified, are restored from words 6 through 18.

  Default: If the last register operand is omitted, only the contents of the register specified by the first register operand are restored. If both operands are omitted, no register contents are restored.

  The address of the save area defined by the calling program must be loaded into register 13 before execution of this macro instruction.

T
: specifies that a 1 is to be set in the low-order bit of the forward link, word 3, in the save area defined by the calling program. (See the MARKRTRN macro description.) This action occurs after completion of the register reloading specified by the first operand. The bit is set to stop the forward chain.

  This parameter is supplied to facilitate tracing; that is, checking program flow. There is no tracing in TSS; this parameter is provided for compatibility with the IBM OS or OS/VS System.

  Specified as: T

RC
: specifies a return code that is to be placed in the 12 low-order bits of register 15, the return-code register.

  Specified as: Register notation (15 only), or an absolute expression whose value lies between 0 and 4095 inclusive.

Initialization: The address of the save area defined by the calling program must be loaded into register 13 before execution of this macro instruction.

Programming Notes: The contents of register 14, the return register, must be restored by means of the first operand of the macro instruction, or the register must be correctly loaded before the macro instruction is executed.

The register operands must not specify that the contents of register 13, the save area register, are to be restored. If the contents of register 13 are to be saved and restored, it should be done according to the linkage conventions described in Appendix E. The RETURN macro instruction assumes that the save area register (register 13) is correctly positioned to the save area defined by the calling program.

If no return code is specified, the contents of the return code register (register 15) will not be changed unless the register operands span the return code register. When a RETURN macro instruction terminates a program, the return code in register 15 can be interrogated by the calling program.

Examples: EX1 is a RETURN macro instruction that restores the contents of registers 2 through 10. A 1 is placed in the low-order bit of word 3 in the save area. EX2 restores the contents of registers 14 through 5 and places a return code of 12 in register 15. EX3 does not restore the contents of any register; however, control is returned to the calling program.

```
EX1      RETURN      (2,10),T
EX2      RETURN      (14,5),RC=12
EX3      RETURN
```

RSVSEG -- Reserve Segment Group (0)

The RSVSEG macro instruction associates an eight character EBCDIC name with a contiguous set of virtual storage segments. A virtual storage segment is defined as a contiguous set of sixteen virtual storage 4096 byte pages aligned on a sixteen page boundary.

| Note: this macro instruction has one or more operands that can be used only by a systems programmer; these operand(s) are defined and specified in the System Programmer's Guide manual.

L-form

Name	Operation	Operand
Symbol	RSVSEG	[RNAME=,LENGTH=,]MF=L

E-form

Name	Operation	Operand
[symbol]	RSVSEG	[RNAME=,LENGTH=,ADDRESS=,]MF=(E,list)

Standard-form

Name	Operation	Operand
[symbol]	RSVSEG	[RNAME=,LENGTH=,ADDRESS=]

Note: All operands are keyword.

RNAME=
specifies the eight character EBCDIC name to be assigned to the segment group.

Specified as: Name enclosed within apostrophies; in E or standard form only, as the address of RNAME expressed as a relocatable expression, RX address or register notation. If register notation is used, the register specified must be the first of a set of paired registers containing the reserved segment name.

Default: If this operand is omitted, the system will assign a unique eight character EBCDIC name to the reserved segment group in the form $$$XXXXX, where XXXXX is from 00000 to 99999.

CAUTION: RNAMES beginning with $$$ or SYS are reserved for system use and cannot be specified by the user.

LENGTH=
specifies the number of contiguous virtual storage segments to be assigned to this segment group.

Specified as: An absolute expression; in the E or standard form only, the address of a halfword expressed as a relocatable expression, RX address or register notation. If register notation is used, the value must be given as a binary number placed in the low order two bytes of the register, right-adjusted. If an expression or RX address is used, the address pointed to must be two bytes long with the length right-adjusted in the field.

Default: If this operand is omitted, the system will assign a length of one virtual storage segment.

ADDRESS=
Specifies the starting address of the virtual storage segment group. This address must be segment aligned.

Specified as: In the E or standard form only, the address of ADDRESS, expressed as a relocatable expression, RX address, or register notation.

Default: If this operand is omitted, the address of the segment group will be returned by the system.

Return Data: On return from RSVSEG, all defaulted operands will have their system assigned values placed in the NAMESEG parameter list and register 15 will contain a return code describing the success of the operation.

Return Codes

00	Successful
04	RNAME invalid
12	Segment not available to user class
16	Invalid address
20	Segment group overlap
24	Invalid length
32	Insufficient space available
36	User generated system reserved name
40	System error or system limit reached

Register 1 contains the address of the Nameseg Parameter List.

Note: The DSECT, CHANSG covers the Nameseg Parameter List.

Programming Notes: The return code in register 15 may be used to construct a branch table to handle the varying results from execution of the RSVSEG macro.

Upon expansion of this macro a set of input flags is constructed in the Nameseg Parameter List. They are:

```
X'40'      RNAME specified
X'20'      ADDRESS specified
X'08'      LENGTH specified
X'02'      RSTRCT=Y specified (system programmers only)
```

Upon execution of RSVSEG, a set of output flags will be constructed with the above values.

Subsequent GETMAIN requests which specify the RNAME (user or system generated), will allocate space within the Reserved Segment Group. If sufficient space is unavailable within the Reserved Segment Group, no attempt will be made to allocate space outside of the Reserved Segment Group. Also, GETMAIN requests which do not specify an RNAME, will not allocate space within a Reserved Segment Group.

L- and E-form Use Example 1:

```
RLIST      RSVSEG     LENGTH=2,MF=L
           RSVSEG RNAME=R1,ADDRESS=RADD,MF=(E,RLIST)
R1         DC  CL8'MYNAME'
RADD       DC  A(X'00400000')  Segment aligned address
```

In the expansion of the L-form, a Nameseg Parameter List will be created in the following format:

```
RLIST--->     +0  .0 A.B 4.  .  .
              +4  .  .  .  .  .  RNAME to be filled
              +8  .  .  .  .  .  in by E-form
             +12  .  .  .  .  .  Not
             +16  .  .  .  .  .  used
             +20  .  .  .  .  .  Address filled in by E-form
             +24  .0 0.0 2.0 8.  .
             +28  .  .  .  .  .  Not used
```

Upon successful execution of the E-form, the Nameseg Parameter List will be as follows:

```
RLIST--->     +0  .0 A.B 4.  .  .
              +4  . M . Y . N . A .
              +8  . M . E .  .  .
             +12  .  .  .  .  .  Register 15=0
             +16  .  .  .  .  .  Register 1=Address
             +20  .0 0.0 4.0 0.0 0.  of RLIST
             +24  .0 0.0 2.6 8.6 8.
             +28  .  .  .  .  .
```

L- and E-form Use Example 2:

```
RLIST2        RSVSEG      MF=L
              RSVSEG      MF=(E,RLIST2)
```

The expansion of the L-form will produce a Nameseg Parameter List as follows:

```
    RLIST2-->       +0   .0 A.B 4.    .    .
                    +4   .    .    .    .    .
                    +8   .    .    .    .    .
                   +12   .    .    .    .    .
                   +16   .    .    .    .    .
                   +20   .    .    .    .    .
                   +24   .    .   .0 0.    .
                   +28   .    .    .    .    .
```

Upon successful execution of the E-form, the Nameseg Parameter List will be as follows:

```
    RLIST2 -->      +0   .0 A.B 4.    .    .
                    +4   . $ . $ . $ . X .   Where 0 ≤ XXXXX ≤ 99999
                    +8   . X . X . X . X .
                   +12   .    .    .    .    .
                   +16   .    .    .    .    .
                   +20   .0 0.S S.0 0.0 0.   Where X'00' ≤ SS ≤ X'FF'
                   +24   .0 0.0 1.0 0.6 8.   0 ≤ SS ≤ 255
                   +28   .    .    .    .    .

                           Register 15 = 0
                           Register 1 = Address of RLIST2
```

Standard-form Use Example 1:

```
     RS      RSVSEG
```

The expansion of the standard-form will produce a Nameseg Parameter List as follows:

```
    CHDXXXXX-->     +0   .0 A.B 4.    .    .   Where CHDXXXXX is an
                    +4   .    .    .    .    .   assembler generated
                    +8   .    .    .    .    .   symbol
                      .
                      .
                      .
                   +28   .    .    .    .    .
```

Upon successful execution of the standard-form, the Nameseg Parameter List will be as follows:

```
    CHDXXXXX-->     +0   .0 A.B 4.    .    .
                    +4   . $ . $ . $ . Y .
                    +8   . Y . Y . Y . Y .   Where 0 ≤ YYYYY ≤ 99999
                   +12   .    .    .    .    .
                   +16   .    .    .    .    .
                   +20   .0 0.S S.0 0.0 0.   Where X'00' ≤ SS ≤ X'FF'
                   +24   .0 0.0 1.0 0.6 8.   0 ≤ SS ≤ 255
                   +28   .    .    .    .    .

                         Register 15 = 0

                         Register 1 = Address of CHDXXXXX
```

If this example were re-entered, the RNAME field would contain a new RNAME of the form $$$ZZZZZ where YYYYY < ZZZZZ ≤ 99999 and the address field would contain a segment address different from that assigned by the previous execution. All other fields would remain unchanged.

Standard-form Use Example 2:

```
    RSVSEG     RNAME='SRNM',LENGTH=LEN1,ADDRESS=ADD1
    ADD1       DS      F
    LEN1       DS      H
```

The expansion of the standard-form will produce a Nameseg Parameter List as follows:

```
    CHDXXYXX-->   +0   .0 A.B 4.   .   .
                  +4   . S . R . N . M .
                  +8   . b . b . b . b .
                   .
                   .
                   .
                  +24  .   .   .6 8.   .
```

Upon successful execution of the standard-form, the Nameseg Parameter List will be as follows:

```
    CHDXXXXX-->   +0   .0 A.B 4.   .   .
                  +4   . S . R . N . M .
                  +8   . b . b . b . b .   Where aa0000 is a
                 +12   .   .   .   .   .   segment address
                 +16   .   .   .   .   .   previously placed in
                 +20   .0 0.a a.0 0.0 0.   field ADD1 & 1111 is
                 +24   .1 1.1 1.6 8.6 8.   a length previously
                 +28   .   .   .   .   .   placed in field LEN1
```

CAUTION: Subsequent executions of this type of example will be unsuccessful if reserved segment name 'SRNM' has not been released.

Standard-form Use Example 3:

```
    RLIST3        RSVSEG      RNAME='RNM1',MF=L
                  RSVSEG      LENGTH=2,MF=(E,RLIST3)
                  RSVSEG      RNAME=NEWNAME,LENGTH=LEN3,MF=(E,RLIST3)
    NEWRNAME      DC    CL3'MYNAME2'
    LEN3          DC    H'16'
```

The expansion of the L-form will produce a Nameseg Parameter List as follows:

```
    RLIST3-->     +0   .0 A.B 4.   .   .
                  +4   . R . N . M . 1 .
                  +8   . b . b . b . b .
                   .
                   .
                   .
                  +24  .   .   .4 0.   .
                  +28  .   .   .   .   .
```

Upon successful execution of the first E-form, the list will be as follows:

```
    RLIST3-->     +0   .0 A.B 4.   .   .
                  +4   . R . N . M . 1 .
                  +8   . b . b . b . b .
                 +12   .   .   .   .   .
                 +16   .   .   .   .   .
                 +20   .0 0.S S.0 0.0 0.   X'00' ≤ SS ≤ x'ff'
                 +24   .0 0.0 2.4 8.6 8.
                 +28   .   .   .   .   .
```

Upon successful execution of the second E-form, the list will be as follows:

```
RLIST3-->       +0   .0 A.B 4.   .   .
                +4   . M . Y . N . A .
                +8   . M . E . 2 . b .
               +12   .   .   .   .   .
               +16   .   .   .   .   .
               +20   .0 0.t t.0 0.0 0.   Where tt ≠ SS and
               +24   .0 0.1 0.4 8.6 8.   X'00' ≤ tt ≤ X'FF'
               +28   .   .   .   .   .
```

Note: If the second E-form had not specified a new RNAME, the system would have passed a return code of four in register 15 and no changes would have been made to the Nameseq Parameter List.

SAEC -- Specify Asynchronous Entry Conditions (S)

The SAEC macro instruction creates an interrupt control block (ICB) to service asynchronous interruptions, and specifies the address of a communication area, the data control block associated with the device the user desires to service, and the interruption handling routine's entry point.

L-form:

Name	Operation	Operand
[symbol]	SAEC	[EP={entry point address|0}]
		[,{DCB=dcb address|SDA=sda address}]
		[,COMAREA=area address]
		[,{INTTYP|ATTNTYP}={{({A|S|R},code[,...]) |NULL|SAVE|RESTORE}]
		[,PFKMSK={({A|S|R},{integer|integer-integer}) |NULL|SAVE|RESTORE}]
		[,MF={L|(E,icb address)}]

Note: There is no standard form of this macro instruction. The name field becomes the ICB address used by a SIR or DIR macro instruction.

EP=

specifies the entry point of the interruption routine to which control is to be transferred when an interruption of the type specified by INTTYP occurs. The routine must be aligned on a fullword boundary.

Specified as: 0, or a symbol (one to eight alphameric characters, the first of which must be alphabetic).

Default: 0. If this operand is omitted, or if EP=0 is specified, the user must place the entry point name of the interruption routine in the interrupt control block generated by this macro instruction.

DCB=

specifies the address of a previously opened data control block associated with the device or unit for which the routine is to service interruptions. If the SYSIN terminal is specified, the DCB parameter must be specified as SYSINDCB, and the USATT macro instruction parameter must be specified to disable the higher priority system-supplied attention interruption routine for the SYSIN device.

Specified as: In the L-form, as a relocatable expression; in the E-form, as an RX address or as register notation (2 through 12).

SDA=
specifies the address of a halfword containing the SDA (Symbolic Device Address) of the device whose interruptions are to be serviced by this routine. If a SYSIN terminal is specified, the SDA parameter must be specified as SYSINSDA, and the USATT macro instruction must be specified to disable the higher priority system-supplied attention interruption routine for the SYSIN device.

| <u>Specified as</u>: In the L-form, as a relocatable expression; in the E-form, as an RX address or as register notation (2 through 12).

COMAREA=
specifies the address of 16-byte area in main storage, aligned on a fullword boundary, that is to be used by the control program to pass interruption information to the interruption routine.

| <u>Specified as</u>: In the L-form, as a relocatable expression: in the E-form, as an RX address or as register notation (2 through 12).

| INTTYP=/ATTNTYP=
| specifies types of interruptions that will cause entry to the interruption routine, the treatment of the interruption information specified by the code operand, or any action to be taken. INTTYP is the preferred keyword to use for new code; ATTNTYP is accepted so that older programs using this keyword need not be recoded.

<u>Specified as</u>:
| A - specifies that the interruption information (code) is to be added to the existing INTTYP/ATTNTYP field of the ICB.

| S - specifies that the interruption information (code) is to be subtracted from the existing INTTYP/ATTNTYP field of the ICB.

| R - specifies that the interruption information (code) is to replace the existing INTTYP/ATTNTYP field of the ICB.

| code - specifies the type or types of interruptions to be added to, subtracted from, or replaced in the INTTYP/ATTNTYP field of the ICB, and can be written as one or more of the following:

ATTN	- indicates an attention interruption.
CANCEL	- indicates that the routine is to service interruptions from the CANCEL key on the alphameric keyboard. The CANCEL key should be reserved to request control program intervention.
ALL	- indicates that the routine is to service interruptions from all sources.
EOS	- indicates that the routine is to service interruptions caused by execution of end-of-order-sequence orders.
AE	- indicates that the routine is to service interruptions caused by asynchronous errors.
END	- indicates that the routine is to service interruptions from the END key on a 2250.
LP	- indicates that the routine is to service interrutions from the light pen on a 3270 or 2250.
NULL	- indicates that none of the types of interruptions covered by INTTYP/ATTNTYP are to be serviced.
SAVE	- specifies that the contents of the INTTYP/ATTNTYP field of the ICB are to be saved. If this or NULL or RESTORE is written, the A, S, or R and interruption-type codes are not written.

RESTORE - specifies that the contents of the INTTYP/ATTNTYP field of the ICB are to be replaced with the mask saved by an INTTYP=SAVE operand in a previous SAEC macro instruction.

PFKMSK= (3270, 3066, 2250 devices only)
specifies the program function key(s) from which asynchronous interrupts are to be serviced by the interrupt routine, or changes the program function keys to be so serviced, or permits the overriding of presently defined program function keys with those defined in a previous SAEC macro.

Specified as:

{A|S|R} - specifies that the program function key(s) that follow (integer) are to be added to, subtracted from, or replace those already specified in the PFKMSK field of the ICB.

integer - identifies the program key numbers to be added, subtracted, or replaced; one or more decimal digits separated by commas in the range 0 to 31; to specify a consecutive range of program function keys, code the first and last key numbers separated by a hyphen; for example, 7-11.

NULL - specifies that the interrupt routine is not to service interrupts from program function keys.

SAVE - specifies that the current PFKMSK field of the ICB is to be saved.

RESTORE - specifies that the current PFKMSK field of the ICB is to be replaced with that of the last SAEC macro instruction having a PFKMSK=SAVE operand.

Note: if NULL, SAVE, or RESTORE is coded, A, S, or R and integer are not coded.

icb address
specifies the address of the interrupt control block.

Specified as: Register notation (1 through 12), or an RX address.

CAUTION: If an interruption routine is to serve multiple units, a separate ICB must be defined (SAEC macro instruction) and specified (SIR macro instruction) for each unit; also, the routine must be reenterable.

Programming Notes: The data control block address or SDA address in an ICB should not be changed while the associated routine is active (currently processing or interrupted before completion of its processing) without first deleting the interruption routine with a DIR macro instruction. After changing the DCB/SDA address, the routine must be reestablished with a SIR macro instruction.

The format of the first four words of the interrupt control block is:

ICB	+0	COMAREA ADDR
	+4	DCB or SDA ADDR
	+8	PFKMSK
	+12	INTTYP/ATTNTYP

Upon entry to an interruption routine, register 1 contains the ICB address and the COMAREA contains the information relating to the interruption to be serviced. The format of the communication area is:

COMAREA	+0	X'03'	OVERLAY	PFK KEY NO.	INTTYP/ATTNTYP
	+4	SENSE DATA			
	+8	X POSITION		Y POSITION	
	+12	RESERVED			

INTTYP/ATTNTYP
indicates the type of interruption that occurred, as follows:

Code	Type
01	END key (2250)
02	Program function key (2250, 3270, 3066)
03	Light pen (2250, 3270)
04	EOS (end-of-sequence) (2250)
05	CANCEL or ATTEN (ALL)
06	AE (asynchronous error) (2250)
07	Sense operation failed (2250)
08	RMI operation failed (2250)

L- and E-Form Use: If neither L- nor E-form is specified, L is assumed. The in-line code for the E-form alters the contents of an ICB. Therefore the MF operand with no other operands is meaningless and produces an assembly error message.

The A, S, R, SAVE, and RESTORE operands are specified only in the E-form of this macro instruction.

Examples: the ICB may be referred to by the symbolic name ICBX1. Conditions are defined for an interruption routine whose initial entry point is the location specified by the symbolic name AR1. All interruptions are processed by this interruption routine. When an interruption on the device specified by the DCB operand GRAPHD1 causes entry to AR1, the interruption data is present in the first four words of AREA1; the address of the ICB is in register 1.

```
ICBX1    SAEC       EP=AR1,DCB=GRAPHD1,INTTYP=(ALL),
                    COMAREA=AREA1,MF=L
         DS         0F
AREA1    DS         CL16
```

If the user wishes to have a routine called whenever PF key 3 on the terminal is pressed, the user could code:

```
SAEC  EP=RTN1,SDA=SYSINSDA,COMAREA=COM,PFKMSK=3
```

Then, whenever PF key 3 is pressed and the user has released the key from the access method (see Device Screen Commands in the Terminal User's Guide), the routine will be called asynchronously at entry RTN1.

SAI -- Save and Inhibit (O)

The SAI macro instruction saves the inhibit status of the task monitor and sets the problem program in the inhibit state. SAI does not handle the inhibit status for program or SVC interruptions for user-defined interruption handling routines.

Name	Operation	Operand
[symbol]	SAI	[address of status area]

address of status area
specifies a one-byte area for saving the prior inhibit status of the task monitor.

Specified as: Register notation (2 through 12), or a relocatable expression.

Default: The problem program is set in the inhibit state.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: The task monitor always dispatches privileged routines in the inhibit state, unless the interruption program was made available to the system by a SIR macro instruction with the operand INHIBIT=NO.

The task monitor dispatches to nonprivileged interruption programs in the enabled state, unless the interruption program was made available to the system by a SIR macro instruction with the operand INHIBIT=YES.

| There are separate inhibit indicators for privileged and nonprivileged programs. The SAI macro instruction sets the appropriate indicator according to the attributes of the program being assembled. A nonprivileged program cannot inhibit dispatching of privileged programs.

SAVE -- Save Register Contents (O)

The SAVE macro instruction is normally written at each entry point of a called program. Upon entry to the program, SAVE stores the contents of specified registers in a save area provided by the calling program. The saved register contents may then be restored by a RETURN macro instruction, an LM instruction, or other programming technique.

Name	Operation	Operand
[symbol]	SAVE	(first register[,last register])[,T][,identifier]

first register,last register
specify the range of registers whose contents are to be stored in the save area defined by the calling program that is pointed to by register 13.

Specified as: The operands are written as unsigned decimal numbers or absolute expressions so that, when inserted in an assembler language STM instruction, they cause the contents of the desired registers in the range of 14 through 12 (that is, 14, 15, and 0 through 12) to be stored. The contents of register 14 and 15, if specified, are saved in words 4 and 5 of the save area; the contents of registers 0 through 12, if specified, are saved in words 6 through 18. The contents of a given register are always saved in a particular word in the save area. For example, the contents of register 3 are always saved in word 9 of the save area, even if the contents of register 2 are not saved. The register operands must not request saving the contents of register 13, which is the pointer to the save area.

<u>Default</u>: If the last register operand is omitted, only the contents of the register specified by the first register operand are saved.

<u>Note</u>: T and identifier are parameters used to facilitate tracing; that is, checking program flow. There is no tracing in TSS; these parameters are provided for compatibility with the IBM OS and OS/VS Systems.

T
specifies that the contents of registers 14 and 15 are to be saved in words 4 and 5 of the save area, if not already saved by the register operands. If the T and last register operands are present and the the first register operand is 14, 15, 0, 1, or 2, the contents of all registers from 14 through the last register value are saved.

identifier
specifies the identifier of the entry point at which the SAVE macro instruction is located.

<u>Specified as</u>: A character string or an asterisk (*). If a character string is specified, it may consist of as many as 255 characters; it must contain no blanks or commas. Because it can have a length greater than eight characters, the operand can be a combination of a data set name and a program name, or some other complex name.

If this operand is written as an asterisk, the entry-point identifier is the same as the symbol in the name field of this macro instruction. If the name field is blank, the name of the control section containing the SAVE macro instruction is used.

Programming Notes: If the called routine is to use register 13, it must save the contents of register 13 and, before termination, restore it. The SAVE macro instruction must not be used for this saving.

When the macro is expanded, both the entry-point identifier and the count of the number of bytes in the identifier, in that order, are placed in front of the actual entry point to the SAVE routine. The entry point identifier is assembled starting at the nearest possible halfword boundary preceding the actual entry point. Because the count byte always immediately precedes the entry point, an extra byte is sometimes needed to achieve the required halfword alignment for the identifier string. When the extra byte is needed, a character blank is inserted at the end of the entry point identifier, immediately preceding the count byte. The count byte contains a count equal to the number of characters in the identifier plus the blank (if used). The count byte itself is not included in the count.

A symbol in the name field of a SAVE macro instruction is an entry-point name. The entry-point name and the entry-point identifier are the same only if the last operand of the macro instruction is an asterisk. The entry-point name is used in passing control to the entry point. If a program in another assembly module is to branch to the entry point, the entry-point name should be an operand of an assembler-language ENTRY statement provided by the user in the current assembly module.

Examples: EX1 saves the contents of registers 14 through 10. The contents of registers 14 and 15 (and registers 0 and 1) are saved because the T operand is written. The entry point identifier is F4RTNA7B99. EX2 saves registers 3 and 4. The entry point identifier is EX2.

Examples		Macro Expansions		
EX1	SAVE(2,10),T,F4RTNA7B99		DC	0H
			DC	CL11'F4RTNA7B99',FL1'11'
		EX1	STM	14,10,12(13)
EX2	SAVE(3,4),,*		DC	0H
			DC	CL3'EX2',FL1'3'
		EX2	STM	3,4,32(13)

SEEC -- Specify External Entry Conditions (S)

The SEEC macro instruction creates an interrupt control block (ICB) to service external interruptions, and specifies the address of a communication area and the interruption handling routine's entry point.

L- and E-form:

Name	Operation	Operand
[symbol]	SPEC	[EP=entry point][,COMAREA=area address] [,INTTYP=message number][,MSGAREA=buffer address] [,MSGLTH=length of message][MSGHDR={Y|N}] [,MF={L|(E,icb address)}]

Note: A symbol is required in the name field with the L-form and becomes the ICB address used by a SIR or DIR macro instruction. If the MF operand is omitted, MF=L is assumed. There is no standard form of this macro instruction.

EP=
specifies the entry point of the interruption routine to which control is to be transferred when an interruption occurs for the message specified by INTTYP.

Specified as: 0, or a symbol (one to eight alphameric characters, the first of which is alphabetic).

Default: 0

If this operand is omitted or if EP=0 is specified, the user must place the entry-point name of the interruption routine in the interrupt control block generated by this macro instruction.

COMAREA=
specifies the address of an area in main storage, aligned on a fullword boundary, that is to be used by the control program to pass interruption information to the interruption routine.

Specified as: A relocatable expression; in the E-form only, as an
| RX address, or register notation (2 through 12).

INTTYP=
specifies the message number (0-255) that will cause entry to the interruption routine.

Specified as: A number, maximum 255. Message numbers 0 to 127 are reserved for nonprivileged programs, message numbers 128 to 236 are reserved for IBM privileged programs, and message numbers 237 to 255 are reserved for installation privileged programs.

MSGAREA=
specifies the address of an area into which the message is to be moved. The message header precedes the message text if MSGHDR=Y is specified.

Specified as: A relocatable expression; in the E-form only, as an
| RX address, or register notation (2 through 12).

MSGHDR=
specifies whether the entire Message Control Block (MCB), the message header and the text, is to be moved into the message area, or just the message text. If the user wants header information, he specifies MSGHDR=Y and must make the message area (specified by the MSGAREA operand) two doublewords longer than the length for the message specified by MSGLTH. The header will be placed in the message area ahead of the text.

Specified as:
Y - the message header is to be placed in the message area as well as the text. If interruptions are received via INTINQ only

(program is in disabled state), Y should be specified.
N - only the message text is placed in the message area.

<u>Default</u>: N

MSGLTH=
specifies the length in doublewords of the message.

<u>Specified as</u>: A number, maximum 255.

icb address
specifies the address of the interrupt control block.

<u>Specified as</u>: An RX address, or register notation (1 through 12).

<u>CAUTION</u>: If an interruption routine is to serve multiple messages, a separate ICB must be defined (SEEC macro instruction) and specified (SIR macro instruction) for each message, and the routine must be reenterable.

<u>Programming Notes</u>: The message number (INTTYP) in an ICB should not be changed while the associated routine is active (currently processing or interrupted before completion of its processing) without first deleting the interruption routine with a DIR macro instruction. After changing the INTTYP, the routine must be reestablished with a SIR macro instruction.

If an external interruption occurs and no routine has been created by a SIR macro instruction to handle the interruption, a message is sent to the terminal indicating that the interruption has occurred but no routine is available to handle the interruption. This occurs for all message numbers except 127, for which no indication is made.

The format of the first three words of the interrupt control block is:

ICB	+0	COMAREA ADDRESS
	+4	RESERVED
	+8	RESERVED
	+16	ENTRY POINT ADDRESS

Upon entry to an interruption routine, register 1 contains the ICB address and the COMAREA contains the information relating to the interruption to be serviced. The format of the communication area is:

COMAREA	+0	X'02'	MESSAGE LENGTH	MESSAGE NUMBER
	+4	MESSAGE AREA ADDRESS		
	+8	RESERVED		
	+12	RESERVED		

<u>Return Data</u>: When MSGHDR=Y is specified, the message (header and text) is placed in the message area by the SEEC macro instruction, the address of the first byte of the message header is placed in the MSGAREA address field of the COMAREA, and the length of the message text in doublewords

is placed in the MSGLTH field of the COMAREA. When MSGHDR=Y is specified, the message area contains:

<u>Offset</u>	<u>Contents</u>
0	Length of message text in doublewords (1 byte)
3	Message Code (1 byte)
4	VSEND SVC (2 bytes)
8	Identification of sending task (2 bytes)
10	Identification of receiving task (2 bytes)
16	Message text starts.

When MSGHDR=N is specified, only the message text is placed in the message area, starting at byte 16.

If MSGHDR=Y is specified and the message is truncated, the truncated length is placed in the COMAREA length field; the header is always passed.

<u>L- and E-Form Use</u>: There is no standard form of this macro instruction, since no linkage is performed. The in-line code for the E-form alters the contents of an ICB. Therefore, the MF operand with no other operands is meaningless and produces an assembly error message.

<u>Examples</u>: 1) The ICB may be referred to by the symbolic name ICBE1. Conditions are defined for an interruption routine whose initial entry point is the location specified by the symbolic name PROG1. When an interruption for message #4 (as specified by the INTTYP operand) causes entry to PROG1, the interruption data is in the first four words of AREA1, and the address of the ICB is in register 1.

```
ICBE1    SEEC       EP=PROG1,INTTYP=4,COMAREA=AREA1,MSGAREA=AREA2,
                    MSGLTH=72,MF=L
         DS         0F
AREA1    DS         CL16
AREA2    DS         72D
```

2) This macro instruction will, when executed, cause the ICB defined in example 1 to be modified, allowing interruptions for message #6 (but no longer for message #6) to be processed by the routine with entry point at PROG1.

```
               SEEC     INTTYP=6,MF=(E,ICBE1)
```

SETDV -- Set Dictionary Value (O)

The SETDV macro instruction allows a user application program to define, manipulate, and delete TSS Dictionary values (entries).

<u>Note</u>: The SETDV macro plus the GETDV macro give the programmer capability of creating, updating and deleting entries in the TSS Dictionary.

Name	Operation	Operand
	SETDV	{symbol,type,value},...

symbol
name of the dictionary entry or value to be set.

<u>Specified as</u>: one-to-eight characters in single quotes, the first character of which must be alphabetic, or an RX address.

Note: if given as an RX address, the address must be preceded by a one-byte length.

type

indicates the type of entry.

Specified as: any one of the following:

'SYN'	- synonym
'DEF'	- default
'INTG'	- integer command symbol word
'CHAR'	- character command symbol word
'FLT'	- floating point number
'LOG'	- logical command symbol word
'HEX'	- hexadecimal command symbol word

Rules concerning type are as follows:

(1) 'CHAR' and 'HEX' must be less than 256 bytes in length.

(2) 'INTG' and 'HEX' must be on a fullword boundary; a length of four is assumed.

Default: none

value

the new value to be given to the dictionary entry defined by the symbol and type parameters.

Specified as: an RX address, a character string, or register notation (2 through 12).

Default: if value is not given, the dictionary entry defined by symbol and type will be deleted.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: There is no validity checking by the SETDV processor to make certain that the value given conforms to the 'type' code specified.

Example 1:

```
          LA    R3,SYMBOL1
       SETDV    ((R3),'INTG',RC4)     set return code for user
          .
          .
          .
          DC    AL1 (L'SYMBOL1)
 SYMBOL1  DC    C'SYSRC'
 RC4      DC    F'4'
```

Example 2:

```
          LA    R3,RC0
       SETDV    ('SYSRC','INTG',(R3))
          .
          .
          .
 RC0      DC    F'0'
```

SETL -- Specify Start of Sequential Processing (R)

The SETL macro instruction (for VSAM, VISAM, and QSAM) positions a data set to the beginning, end, previous record, or any point within the data set.

Name	Operation	Operand
[symbol]	SETL	dcb address,processing type[,record key]

dcb address
specifies the address of the data control block opened for the data set being processed.

| Specified as: Register notation (1 through 12), or an RX address.

| processing type
| specifies the starting point for processing and any optional serv-
| ices requested.

Return Data: Register 15 is set to 08, meaning normal return

SETTU — Set User Timer (R)

The SETTU macro instruction sets the user timer field in the XTSI, thereby limiting your task's execution time.

Name	Operation	Operand
[symbol]	SETTU	[time]

time
    specifies the time duration in milliseconds that you want placed in the user timer field.

Specified as: A decimal number from 0 to 55364812 or, if the number is first placed in a register, in register notation (1 through 12).

Default: It is assumed that the issuer has placed the time duration in register 1.

Initialization: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding SETTU. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding SETTU must be issued with the PRIVILEGED option.

Execution: The quantity contained in register 1 is converted to microseconds and stored in the extended task status index field called user timer value (XTSUTI).

Example: Assume that register 5 contains the number of milliseconds to which you'd like to set the user timer. You might write:

```
NAME   SETTU    (5)
```

SETUP — Set Up Task Status Index Field (R)

The SETUP macro instruction permits you to alter or set the contents of a selected field in the TSI.

Name	Operation	Operand
[symbol]	SETUP	[field][,register]

field
    specifies the field you want to set or alter.

Specified as: One of the codes described below, or, if a value corresponding to one of the codes (also shown below) is first placed in register 15, as (15).

USERID - set the user identification field
SYSIN - set the input data set location field
SYSOUT - set the output data set location field
SOPRIV - operator/(combined with privilege class-E)
         system programmer privilege
SPPRIV - system programmer, nonprivileged
UPRIV - user

CONV - set the conversational task flag
ITMFLG - set the intertask message flag
XPR - set the external priority flag
AUTH - set the privilege field
MAV - set the maximum auxiliary storage field

Field	Value
USERID	1
SYSIN	3
SYSOUT	4
SOPRIV	6
SPPRIV	7
UPRIV	9
CONV	10
ITMFLG	12
XPR	13
AUTH	14
MAV	16

register
designates the even-odd register pair in which you have placed the information you want put into the specified TSI field.

Specified as: The odd register, expressed as an absolute expression or register notation.

Initialization: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding SETUP. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding SETUP must be issued with the PRIVILEGED option.

Execution: From one to eight bytes of registers 0 and 1 are inserted into the task status index field specified by the low-order byte of register 15. The number of bytes to be inserted depends on the field specified.

Field	Code	Implied length (bytes)
USERID	1	8
SYSIN	3	2
SYSOUT	4	2
SOPRIV	6	1
SPPRIV	7	1
UPRIV	9	1
CONV	10	1
ITMFLG	12	1
XPR	13	2
AUTH	14	1
MAV	16	2

Example: Assume that registers 12 and 13 contain an eight-character user identification. You might write:

```
TEST    SETUP    USERID,(13)
```

SETUR -- Set Up Unit Record Device (R)

The SETUR macro instruction specifies the configuration for online printers and card punches.

Name	Operation	Operand
[symbol]	SETUR	{dcb address[,setup]|parameter

dcb address
specifies the address of the data control block opened for processing a data set on a printer or card punch.

<u>Specified as</u>: A relocatable expression or register notation.

<u>Default</u>: None

setup
specifies the address of the desired form number for the punch. For printers, it specifies the name of the default region in the SYSUCS data set from which all printer defaults (FCB, CHAIN/TRAIN, etc.) may be found.

<u>Specified as</u>: one to six alphameric characters

<u>Default</u>: PAPER.

parameter pointer
specifies the address of a parameter list (which is defined by the CHASUR DSECT) which contains the exact specifications for a printer setup. This parameter list is in the following format:

```
SURORG   DS   0F
SURDCB   DS   A         ADDR OF MSAM DCB
SURCHARS DS   0C        START OF CHARS TO LOAD
SURCHAR1 DS   CL6       REG NAME OF 1ST CAT ENTRY
SURCHAR2 DS   CL6       REG NAME OF 2ND CAT ENTRY
SURCHAR3 DS   CL6       REG NAME OF 3RD CAT ENTRY
SURCHAR4 DS   CL6       REG NAME OF 4TH CAT ENTRY
SURFCB   DS   CL6       FCB REG NAME TO LOAD BY
SURDSN   DS   CL44      FQN OF SYSUCS DS TO USE FOR LOAD
SURCPDSN DS   CL44      FQN OF COPY MOD DS
SURBURST DS   CL1       Y! FOR BTS
SURPAPER DS   CL10      PAPER TO USE      ** NET **
SURCOPYG DS   X         NO COPIES OF DS
SURCOPY  DS   XL8       NO COPIES OF PAGE (ONLY 1ST BYTE USED)
SURFLASH DS   CL8       NAME OF FLASH IMAGE
SURFCNT  DS   X         COUNT OF COPIES TO FLASH
SURFORM  DS   CL6       NAME OF DEFAULT REGION IN UCS
SURVID   DS   CL6       VID THIS JOB
SURFLG   DS   X         FLAG BYTE
SURFLSH  EQU  SURFLG    0=FLASH MAY OR MAY NOT BE REQD
SURFLSHM EQU  X'80'     1=DO NOT FLASH
SURDFLT  EQU  SURFLG    0=CHARS PARM FILLED IN BY PRINT CMND
SURDFLTM EQU  X'40'     1=CHARS FILLED IN VIA DEFAULT
SURVID2  DS   CL6       VERSION ID OF COPY MOD DS
SURLEND  EQU  *
SURL     EQU  *-CHASUR  CURRENTLY USED LENGTH
         DS   2X        USED FOR ALIGNMENT (SPARE)
         DS   21F       USED FOR ALIGNMENT (SPARE)
SURLEN   EQU  *-CHASUR  LEN OF TABLE
```

<u>Programming Notes</u>: To ensure a valid setup, the SETUR macro instruction should be issued before any I/O operations are directed to a printer or punch. This is done by issuing SETUR immediately after opening a data set or after the FINISH macro instruction is executed and the I/O operation completed.

<u>Card Punch</u>: The setup for a card punch is described by the form number of the card that the operator is to load into the punch-feed hopper of the punch. This form number is an installation-defined constant. When the macro instruction is executed, the SETUR routine determines which form is mounted in the punch (the currently mounted form number -- or zeros if the DCB was just opened -- is stored for each device in the SDAT). If the desired form is already mounted, control is returned to the invoking routine with a return code of 0. If the form is not mounted, a message is written to the operator (WTO) to mount the desired form number (6-byte parameter), and to ready the punch. A return code of 4 is provided to the calling task. When the operator indicates that the punch is properly loaded, by causing a not-ready to ready interruption, the SDAT is changed to reflect the new form number. On the next call to SETUR, control is returned to the invoking routine with a return code of 0.

<u>Printer</u>: if the 'dcb address,setup' form of this macro is used, the value specified for the setup parameter is used as the index into the SYSUCS dataset from which printer setup defaults are obtained. The default region of the SYSUCS dataset must specify FCB, PAPER, and print train requirements.

When the 'parameter pointer' form of the macro is used, SETUR will fill in any missing defaults based upon the value specified in the SURFORM value in the parameter list. In either case, should a required parameter not be filled in, SETUR will issue an appropriate return code.

<u>Execution</u>: The SETUR macro instruction returns a code in register 15 indicating the manner in which the SETUR call was completed. All return codes are defined in Figure 35.

Return Code	Meaning
0	Operation completed successfully.
4	Operation not complete; SETUR macro instruction should be reissued.
8	Unrecoverable I/O error occurred while attempting to load the device.
12	User software error.
16	System software error.
20	RJE disconnect error.
24	Job cancelled.
28	Page backup requested.
	Note: for return codes 12 and 16, register 1 will point to a prompt parameter list indicating the exact cause of the error.

Figure 35. Return codes for the SETUR macro instruction

When the SETUR macro instruction is executed, the routine determines if the present configuration of the printer, specified in the SUR TABLE, pointed to by the SDAT, is the configuration requested for this SETUR call. If the form, carriage tape (FCB) chain/train, etc., are present, control is returned to the invoking routine. If the desired configuration is not present, the system acts to achieve the desired configuration.

SETUR uses the SYSUCS data set to build the necessary blocks to load a printer configuration. The SYSUCS data set used may or may not be user specified. If defaulted, SETUR uses the system owned SYSUCS data set TSS*****.SYSUCS(0); this data set contains all the information needed to load the 1403, 3211, and 3800 printers.

SYSUCS: this data set is a region data set consisting of 4 basic regions. Each region name is 8 bytes long, the first two bytes of which are predefined by the system. They are as follows:

1. CTXXXXX -- character arrangement table region. This region contains the information needed to load the UCSB in the 3211 and 1403 printers, and the translate tables and WCGMs in the 3800 printer. For the 3800 printer, it may also contain the name(s) of graphic modification regions. A maximum of 12 names may be specified.

2. FBXXXXXX — format control buffer region. This region contains the information needed to indicate which density and carriage control tape are needed for the 1403 printer, and the FCB specification and density settings for the 3211 and 3800 printers.

3. GFXXXXXX — graphic modification region. This region the picture images needed to built graphic modifications for the 3800 printer. All the standard IBM graphic modifications are in TSS*****.SYSGRAPH(0).

4. standard setup region. This region contains the default information needed for a standard printer setup. It is also used to backfill any setup information required but not specified.

Example 1: the example that follows contains the information needed to load the 3211 and 3280 printers with the P11 chain/train configuration. This is indicated by the DEVICE=3211/3800,NAME=P11 statements. The load information immediately follows this statement(s). In the case of the 3211 it is the chain/train image. For the 3800 it is the translate table followed by the WCGM ID. This example does not define the P11 train image for the 1403. However, it does indicate where this information may be found. The statement DEVICE=1403,SEE=(PN,1403) indicates the P11 compatible 1403 chain/train image may be found in the region CTPN of the SYSUCS data set.

```
CTP11    0000100 DEVICE=1403,SEE=(PN,1403)
CTP11    0000200 DEVICE=3211,NAME=P11
CTP11    0000300 1'BDJL-5K*C(NA@=E0?)S¬#R>V92"68<XYT|GF%H._U07/P3WMIQ,4
CTP11    0000350 1'BDJL-5K*C(NO$=E:#)SA&RZV9+G682;YT<XF%H._U07/P3WMIQ,4
CTP11    0000400 1'BDJL-5K*C(NA@=E0?)S¬#R>V92"68<XYT|GF%H._U07/P3WMIQ,4
CTP11    0000450 1'BDJL-5K*C(NO$=E:#)SA&RZV9+G682;YT<XF%H._U07/P3WMIQ,4
CTP11    0000500 1'BDJL-5K*C(NA@=E0?)S¬#R>V92"68<XYT|GF%H._U07/P3WMIQ,4
CTP11    0000550 1'BDJL-5K*C(NO$=E:#)SA&RZV9+G682;YT<XF%H._U07/P3WMIQ,4
CTP11    0000600 1'BDJL-5K*C(NA@=E0?)S¬#R>V92"68<XYT|GF%H._U07/P3WMIQ,4
CTP11    0000650 1'BDJL-5K*C(NO$=E:#)SA&RZV9+G682;YT<XF%H._U07/P3WMIQ,4
CTP11    0000700 END
CTP11    0001000 DEVICE=3800,NAME=P11
CTP11    0001100
CTP11    0001200     TRANSLATE TABLE
CTP11    0001300
CTP11    0001400 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0001500 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0001600 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0001700 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0001800 00FFFFFFFFFFFFFFFFFFFF0B0C0D0E0F
CTP11    0001900 10FFFFFFFFFFFFFFFFFFFF1B1C1D1E1F
CTP11    0002000 2021FFFFFFFFFFFFFFFFFF2B2C2D2E2F
CTP11    0002100 FFFFFFFFFFFFFFFFFFFF3A3B3C3D3E3F
CTP11    0002200 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0002300 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0002400 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0002500 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
CTP11    0002600 FF010203040506070809FFFFFFFFFFFF
CTP11    0002700 FF111213141516171819FFFFFFFFFFFF
CTP11    0002800 FFFF2223242526272829FFFFFFFFFFFF
CTP11    0002900 30313233343536373839FFFFFFFFFFFF
CTP11    0003000
CTP11    0003100     WCGM PAIRS
CTP11    0003200
CTP11    0003300 (82,00)
CTP11    0003400
CTP11    0003500      GRAPHIC NAMES MAX 12
CTP11    0003600
CTP11    0003700 END
```

Example 2:

```
FBSTD6   0000100 DEVICE=1403
FBSTD6   0000200 FORMAT=STANDARD,6
FBSTD6   0000300 END
FBSTD6   0000400 DEVICE=3211
FBSTD6   0000500 FORMAT=1(6,1),62(6,12),66(6,9)
FBSTD6   0000600 END
FBSTD6   0001300 DEVICE=3800
FBSTD6   0001400 FORMAT=1(6,1),62(6,12),66(6,9)
FBSTD6   0001500 END
```

The "DEVICE=" keyword signals the start of the device dependent information. For the 1403 the operator will be requested to mount the carriage tape 'STANDARD' and set the printer density to 6 lines per inch (LPI). For the 3211 and 3800 an FCB image setting the density to 6 with channel code 1 at line 1, channel code 12 at line 62, and channel code 9 at line 66 will be built.

Example 3:

```
STPAPER   0000200 PAPER=1PLY
STPAPER   0000300 FORMAT=STD6
STPAPER   0000400 CHARS=P11,H11
```

The default region of the SYSUCS data set is used by both the SETUR process and the print command. SETUR uses this region to backfill defaulted values in the SETUP request. The print command uses it to fill in defaulted values in the batch work queue. This information is used by the batch monitor to schedule print jobs on the correct printer. At print request time the 'CHARS=' keyword indicates that either a P11 or H11 train image can satisfy the print request. At SETUR time the 'CHARS=' keyword indicates that the P11 train image should be loaded in the printer. PAPER type is 1PLY regardless of printer type. The region FBSTD6 will be used to fulfill the FCB requirements based upon device type.

<u>Example 4</u>:

In the example that follows two picture images have been defined. Both pictures will have a pitch value of 10 as indicated by the 'PITCH=' keyword. The keyword 'CODE=' defines the displacement into the translate table where the graphic modification is to be placed. A maximum of 24 picture images may be specified in a graphic modification region. The first line of each picture image must specify the code and pitch value. The second line, in the above example, is optional and is used for reference purposes only. Each picture image lmust have 24 lines. The system will accept a maximum of 18 characters per line. Short lines will be padded to the right with blanks, long lines will be truncated.

```
| GFGRF1  0000100 CODE=5B  PITCH 10
| GFGRF1  0000200 123456789012345678
| GFGRF1  0000300
| GFGRF1  0000400
| GFGRF1  0000500
| GFGRF1  0000600
| GFGRF1  0000700     ***     ***
| GFGRF1  0000800    *****   *****
| GFGRF1  0000900     ***     ***
| GFGRF1  0001000
| GFGRF1  0001100    ***       ***
| GFGRF1  0001200    ***       ***
| GFGRF1  0001300    ***       ***
| GFGRF1  0001400    ***       ***
| GFGRF1  0001500    ***       ***
| GFGRF1  0001600    ***       ***
| GFGRF1  0001700    ***       ***
| GFGRF1  0001800    ***       ***
| GFGRF1  0001900    ****     ****
| GFGRF1  0002000     ***********
| GFGRF1  0002100      *********
| GFGRF1  0002200
| GFGRF1  0002300
| GFGRF1  0002400
| GFGRF1  0002500
| GFGRF1  0002600
| GFGRF1  0002700 CODE=7B  PITCH 10
| GFGRF1  0002800 123456789012345678
| GFGRF1  0002900
| GFGRF1  0003000
| GFGRF1  0003100
| GFGRF1  0003200
| GFGRF1  0003300     ***     ***
| GFGRF1  0003400    *****   *****
| GFGRF1  0003500     ***     ***
| GFGRF1  0003600         ***
| GFGRF1  0003700         ***
| GFGRF1  0003800        *****
| GFGRF1  0003900       *** ***
| GFGRF1  0004000       *** ***
| GFGRF1  0004100       *** ***
| GFGRF1  0004200      ***   ***
| GFGRF1  0004300      ***   ***
| GFGRF1  0004400     ***********
| GFGRF1  0004500     ***********
| GFGRF1  0004600    ***       ***
| GFGRF1  0004700    ***       ***
| GFGRF1  0004800
| GFGRF1  0004900
| GFGRF1  0005000
```

SETVLOCK -- Set VM Lock (O)

SETVLOCK is used to set a VM Lock.

Name	Operation	Operand
[symbol]	SETVLOCK	lock,log [,SET=set]

lock
    specifies the VM Lock to be set.

Specified as: an RX address.

log
specifies the VM Lock Anchor to be used to record the status of the specified lock.

Specified as: the symbol naming a LOGVLOCK macro.

set
specifies an address in the current module to be branched to if the specified lock is already marked "set".

Specified as: an RX address.

Default: The status of the lock will not be checked.

Execution: If the branch address is specified and if the VM Lock Anchor indicates "set" the branch will be performed. Otherwise, the specified VM Lock will be set and the VM Lock Count (ISAVLKCT) in the task's Interrupt Storage Area (CHAISA) WILL BE INCREMENTED. The address of the lock will be saved in the VM Lock Anchor for use by OPNVLOCK, etc.

CAUTION: This macro must be protected from task interrupts by ITI/PTI.

Programming Note: Refer to VM Locking in Section 3.

SETXP — Set External Page Table Entries (R)

The SETXP macro instruction allows a range of virtual storage to be associated with a set of external storage addresses. It also flags pages as "unprocessed by dynamic loader." The first reference to the page or pages will then cause control to be given to the dynamic loader.

Name	Operation	Operand
[symbol]	SETXP	

Note: There are no operands.

Initialization: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding SETXP. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding SETXP must be issued with the PRIVILEGED option.

Execution: The first bit of the halfword immediately following the SVC is interpreted as a flag. If this bit is 1, the high-order bit of the SDA indicates which entry has been processed by the loader. The maximum page count is 1022. The low-order 10 bits of the halfword following the SVC are interpreted as a page count. The first fullword following the SVC contains the virtual storage address at which the external page table entries are to be set. After this word — and depending on the page count — are a number of words; each word contains an external storage address that is to be associated with a page in the virtual storage range. If the unprocessed-by-loader flag is set for a page, the first reference to that page by a program causes control to be given to the dynamic loader via a task-program interruption type 16.

The external page table entries supplied in the parameter list are set as indicated. The unprocessed-by-loader bit is set for each page whose bit string flag is a 1 and the high-order bit of the SDA is zero. This allows a mixed list to be processed.

__Return Data__: None.

__Example__: Suppose that you want to set external page table entries for three pages beginning at location NEW. You might write:

```
SAMPL   EX      0,SET
        B       SOMEPLACE
SET     DS      0F            SVC MUST BE ON FULL WORD BOUNDARY
        SETXP
        DC      H'3'          NO BIT STRING, THREE PAGES
        DC      A(NEW)        ADD EXTERNAL PAGE TABLE ENT AT NEW
        DC      H'12'         SYMBOLIC DEVICE NUMBER
        DC      H'115'        RELATIVE PAGE ON DEVICE
        DC      H'35'         SYMBOLIC DEVICE NUMBER
        DC      H'51'         RELATIVE PAGE ON DEVICE
        DC      H'12'         SYMBOLIC DEVICE NUMBER
        DC      H'34'.        RELATIVE PAGE ON DEVICE
```

__SETXTS -- Set Up Extended Task Status Index Field (R)__

The SETXTS macro instruction enables you to set the estimated run time of your task in the XTSI.

Name	Operation	Operand
[symbol]	SETXTS	[field]

field
specifies the XTSI field to be set.

Specified as: ESTIM, which indicates that the estimated run-time field of the XTSI is to be set; SET24, which indicates 24-bit machine addressing is to be used; or, if the decimal value of 3 (for ESTIM) or 12 (SET24) is first loaded into register 15, as: (15).

Default: It is assumed that the issued has placed a value in register 15.

Initialization: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding SETXTS. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding SETXTS must be issued with the PRIVILEGED option.

Execution: The value in registers 0 and 1 when SETXTS is issued is stored in the extended task status index field indicated by the code in register 15.

Example: Suppose you want to set the estimated run-time field of the extended task status index. You could write:

```
            SR        0,0
            L         1,=F'runtime'
      NAME  SETXTS    ESTIM
```

SIPEHOOK -- System Performance Evaluation (O)

The SIPEHOOK macro instruction is assembled into various resident supervisor modules so that system data may be collected by the System Internal Performance Evaluation Module (SIPE)

Name	Operation	Operand
[symbol	SIPEHOOK	number-value,hookcode-value

number
specifies a unique number for this SIPEHOOK within this assembly module.

Specified as: a two digit decimal number.

hookcode
specifies which SIPE collector is to be activated because of this hook.

Specified as: a three digit decimal number.

Execution: The action that occurs when a hook is reached is actually determined by the setting of an instruction switch located in the prefixed storage area (PSA) of main storage. (PSA is the term used to describe main storage locations 0-4095, which can be addressed without a base register.). When TSS startup is completed, this instruction switch contains a NOPR instruction (actually, a two-byte BCR isntruction, with condition code 0).

When control arrives at a hook, this central switch is the subject of an EXECUTE instruction. If SIPE is not being used, the NOPR instruction is executed, and control flows through the hook. However, if SIPE is active, the initialization phase of SIPE has reset this central switch to an SVC. This SVC is executed by the hook and results in a transfer of control to SIPE, which recognizes the SVC code as denoting hook execution. Basically, the following events occur for a selected hook:

1. The hook is entered, executing the switch in the PSA region (SVC).

2. The hardware-stored SVC old PSW contains the current machine status and the instruction counter.

3. The SVC new PSW becomes active.

   (a) SIPE saves all machine registers.

   (b) SIPE locates the hook via the SVC old PSW (instruction counter) and inspects the hook identity code (a constant included in the hook).

   (c) A collector is given control to abstract the appropriate data for this hook and file it in the output buffer.

   (d) The I/O buffer is output if necessary.

5. The machine registers are restored.

6. The SVC old PSW is loaded, returning control to the host module at a point just past the hook.

<u>Example</u>: Suppose SIPE collector 145 is to be activated in a supervisor module. The macro instruction might be written:

```
         SIPEHOOK  01,145
```

This would generate:

```
         EX  0,PSASIP
         NOP *-*
         ORG *-2
         DC  AL1(145)
         DC AL1(255)
```

<u>STORE -- Store Register Contents (O)</u>

The STORE macro instruction stores the contents of one or more registers.

Name	Operation	Operand
[symbol]	STORE	area,(first register[,last register])

area
    specifies the address of the storage area in which the register or registers are to be saved.

<u>Specified as</u>: An RX address, or register notation. If register notation is used, the address must first be loaded into the specified register.

first register
    specifies the first in a range of registers whose contents are to be saved, or the only register whose contents are to be saved.

    Specified as: A decimal number from 8 through 15.

last register
    specifies the last register in a range of registers.

    Specified as: A decimal number not greater than 15.

    Default: Only the register specified in the first register operand is saved.

Programming Notes: The area must be large enough to contain the specified range of registers.

STXTR -- SET and XTRCT Table

The STXTR is a macro used for generating internal tables for use by the three SET/XTRCT routines -- CEAH2, CEAS2, and CEAS4.

Name	Operation	Operand
[symbol]	STXTR	table,field,type

table
    specifies the name of the dsect which is used in each particular routine.

    Specified as: CHATSI for CEAH2
    CHASYS for CEAS2
    CHAXTS for CEAS4

field
    specifies the field within the dsect which is to be SETup or XTRCTed.

    Specified as: any field within the particular dsect used.

type
    specifies whether the field can only be XTRCTed or also SETup.

    Specified as: SETUP - setup or extracted
    XTRCT - extracted only

Programming Notes: The table generated is in a standard form that the SET/XTRCT modules interpret to perform the correct movement of data from virtual memory to the corresponding supervisor tables.

SYSER -- Indicate Nonresident-Program-Detected Error (O)

The SYSER macro instruction is the means by which a nonresident program reports errors it has detected.

Name	Operation	Operand
[symbol]	SYSER	error type,fillin,id1,id2,id3,call

error type
specifies the type of error detected.

Specified as: One of the codes shown in Figure 23 under the ERROR macro instruction.

fillin
must be included for compatibility.

Specified as: Any two-digit decimal number in the range 00 through 27.

id1
is the first of three unique identifiers for the message to be issued when SYSER is invoked.

Specified as: A decimal number in the range 1 through 83.

id2
the second of three unique identifiers for the message to be issued when SYSER is invoked.

Specified as: A decimal number in the range 1 through 99.

id3
the third of three unique identifiers for the message to be issued when SYSER is invoked.

Specified as: A decimal number in the range 1 through 999.

call
is used to identify one of several calls in a module.

Specified as: A decimal number from 1 through 99.

Initialization: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding SYSER. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding SYSER must be issued with the PRIVILEGED option.

Execution: The processing unit receiving the SYSER SVC stops all other processing units in the system. A message (see "SYSER DUMP" in Section 5) is issued at the operator's terminal, the system enters the wait state, and, at the installation's discretion, a dump is taken.

If the error type is 2 (major software), a program interruption 202 is queued on the calling task; this ultimately results in its abnormal termination. If the error type is 3 (hardware failure), the SVC 228 routine transfers control to the recovery nucleus. If the error type is 1 (minor software), or if the recovery nucleus returns control to the SVC 228 routine, all other processing units in the system are restarted; control is then returned to the instruction following the SYSER parameter list.

Programming Note: Part of the message issued at the operator's terminal is a nine-digit SYSER code; this code is formatted from the id1 (aa), id2 (bb), id3 (ccc), and call (nn) operands of the SYSER macro instruction and has the form aabbcccnn. This construction permits you to identify calls to the system error processor from privileged virtual storage modules to facilitate debugging. You might, for example, assign a particular id1 code to a group of related modules, assign a particular id2 code to a subset of this group, and a particular id3 code to a module or group of modules within this subset; such an arrangement would identify the source of the call to the system error processor. You could then, using the call operand, assign sequential numbers to the SYSER calls issued by that module or group of molules to aid recognition of particular errors resulting in calls within the sequence. For example, you might write:

```
                           ┌──────────id1
                           │ ┌────────id2
                           │ │  ┌─────id3
                           │ │  │ ┌───call
             SYSER  1,00,13,6,99,1
```

and the resulting SYSER code, 130609901, would identify the error which resulted in the call to the system error processor.

To avoid the possibility of issuing different SYSER calls with the same SYSER code (thus duplicating the messages issued at the operator's terminal and creating confusion as to the reason for the call), see <u>System Messages</u> for those SYSER codes already in use in the system.

<u>Example</u>: Suppose your task detects a minor software error and you want to get just the basic SYSER output. You might write:

```
       BUG    SYSER    1,00,2,0,23,01
```

<u>TSEND -- Force Time Slice End (R)</u>

The TSEND macro instruction forces on your task an early time slice end.

Name	Operation	Operand
[symbol]	TSEND	

<u>Note</u>: There are no operands.

<u>Initialization</u>: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding TSEND. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding TSEND must be issued with the PRIVILEGED option.

<u>Execution</u>: The current time slice of the task issuing the SVC is terminated.

<u>Example</u>: If you want to cause your current time slice to come to an end, you might write:

```
       XYZ       TSEND
```

<u>TSTVLOCK -- Test VM Lock (O)</u>

The TSTVLOCK macro is used to test the recorded status of a VM lock.

Name	Operation	Operand
[symbol]	TSTVLOCK	log,[set],[open]

log
    specifies the VM Lock to be tested.

Specified as: the symbol naming a LOGVLOCK macro.

set,open
specify addresses in the current module to be branched to if the lock is marked "set" or "open", respectively.

Specified as: RX addresses.

Execution: The specified VM Lock Anchor is tested, and the appropriate branch is executed.

Programming: Refer to VM Locking in Section 3.

TWAIT -- Wait for Terminal I/O Interruption (R)

The TWAIT macro instruction checks for a response to a message you have sent and, pending its arrival, puts your task in the delay state, which causes your task's pages to be moved to auxiliary storage.

Name	Operation	Operand
[symbol]	TWAIT	

Note: There are no operands.

Execution: The SVC must be the subject of an Execute instruction and must occupy the second halfword of a fullword control block called an event control block (ECB). The resident supervisor checks the second bit of the halfword preceding the supervisor call and interprets this bit as the event complete bit. If this bit is 1, the supervisor returns control and the SVC is in effect a NOP (no operation). If the bit is 0, and there are any unmasked interruptions queued on the task, a NOP is also affected. Otherwise, the supervisor sets the TWAIT flag in the task's TSI to 1 and puts the task in the delay state; this causes time slice end to occur for the task. The task is removed from the delay state when any task-interruption -- if the task is enabled -- occurs.

Example: Suppose you send a message to some terminal and are waiting for a response. The posting routine associated with the IOCAL (see the IOCAL macro instruction) used to transmit the message to the terminal is responsible for setting the event-complete bit of an event control block to 1. You have reached a point in your program beyond which you do not wish to continue until the IOCAL posting routine has been entered. You might write:

```
           EX      0,TEST+2
           B       IOCOMPLETE
     TEST  DS      0F            ALIGN
           DC      H'0'          POSTING FLAGS
           TWAIT
```

UFLOW -- User Flow for TSS and MTT (R)

The UFLOW macro instruction is used (for example, by the FLOW command processor) to modify or obtain either the conversational task limit and the number of current TSS users, or the multiterminal task (MTT) application user limits and the number of current users for each application.

Name	Operation	Operand
[symbol]	UFLOW	

Note: There are no operands (see Initialization).

Initialization: Before executing this macro instruction, registers 1 and 0 must be loaded with the following parameters:

Register 1
An action code, specifying the action to be taken.

Code	Meaning
1	Set the conversational task limit to the value specified in the low-order two bytes of register 0.
2	Obtain the number of conversational tasks currently in execution and the conversational task limit (see Return Data below).
3	Set MTT user limits for each of the application names specified in the input buffer that is pointed to by register 0. The input buffer must contain:

0 7	8 9	10 11	12 13	14 15
Application name (1)		MTT user limit (1)		
Application name (2)		MTT user limit (2)		
etc.				

The application name must be left-aligned and consist of up to eight alphameric characters; the first of which must be alphabetic. Setting the application name to X'FF' indicates the end of the buffer list, for both action codes 3 and 4. The MTT user limit must be specified in binary. The low-order bytes that are not used must contain blanks.

Code	Meaning
4	Obtain the number of MTT users in execution for each application name that is currently active in the system. Register 0 must contain the address of a virtual storage buffer into which the application names and the statistics pertaining to each can be recorded when the requested processing is performed.

Register 0
Either the conversational task limit, representing the total number of TSS users that may be logged on concurrently, if action code 1 is specified, or a virtual storage buffer address, if action code 3 or 4 is specified.

The task limit that is specified in the two low-order bytes does not affect users that are already logged on. If set to zero, no additional users may logon to TSS; the maximum must always be less than or equal to the system maximum. Any virtual storage address that is specified should point to a buffer that starts on a page boundary and does not exceed a page length. This address can point to either an input buffer (see action code 3) or to an output buffer (see action code 4). Two system DSECTs, CHAOFL and CHAUFN, respectively, are available within TSS for use in referring to fields in those buffers.

A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to coding UFLOW. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued, prior to coding UFLOW, must be issued with the PRIVILEGED option. To ensure (for action codes 3 and 4) that the buffer will be in main storage when UFLOW is executed, an MVC instruction must immediately precede the UFLOW macro instruction.

CAUTIONS: Use of UFLOW (which produces an SVC 187) is restricted to tasks having system programmer authority (O or P). Any virtual storage buffer that is provided must not go over a page boundary.

Execution: The privilege specified by the DCLASS macro instruction is verified. If acceptable, the action code is validated. For action codes 1 and 2, the appropriate limit field, conversational TSS task limit (action code 1), or the MTT application user limit (action code 3) is set in the multiterminal status control block (MTSCB). For action codes 3 and 4, the requested statistics (current number of conversational TSS tasks or current number of MTT users on a specified MTT application) and the system maximums for such limits (see Programming Notes) are recorded in the buffer. All error conditions are identified by return codes or, for action code 3, in the original input buffer (see Return Data below).

Return Data:

Register 0
For successful execution of action codes 2 and 4, contains the requested TSS or MTT statistics in the form:

For action code 2:

0 15	16 31
Current number of conversational tasks	Conversational task limit

For action code 4:

0 7	8 9	10 11	12 13	14 15
Application name (1)	Current MTT users	User limit	Maximum number of users	
Application name (2)	Current MTT users	User limit	Maximum number of users	
etc.				

The application name must be left-aligned and must consist of up to eight alphameric characters, the first of which must be alphabetic. When the application name field contains hexadecimal Fs, it indicates the end of the output buffer list. The low-order bytes that are not used contain blanks. The current MTT user value, the MTT user limit, and the maximum number of MTT users are all binary values.

For action code 3: register 0 points to the input buffer and may contain error indications:

1. If an application name is nonexistent, the two halfwords starting at byte 10 in the input buffer are set to X'FFFF'.

2. If the maximum allowable user limit (recorded in MTSMAX in the MTSCB) is exceeded, the two halfwords starting at byte 10 in the input buffer are set to C'**', and the maximum value is placed in the next halfword (at byte 14).

Register 15
Contains a return code:

Code	Meaning
0	Normal completion
4	Conversational task limit is larger than maximum value (MTSMAX) for action code 1
8	Action code specification error
12	Buffer exceeded page boundary

Programming Notes: The initial TSS conversational task limit is established during system generation with the TSKLMT macro instruction (see System Generation and Maintenance); the number of MTT administrators (or MTT tasks) is included in the count of conversational tasks.

The user limit specified for each MTT application program with UFLOW can never exceed the maximum value originally established by the MTT administrator when he issued an MTT command.

Before UFLOW is first issued, a GETMAIN macro instruction can be issued to get the buffer, which can be retained for the duration of the task.

If a conversational TSS task ends abnormally (completion code 2), a new task is created regardless of the conversational limit.

A command, FLOW, available only to system managers, administrators (see Managers and Administrator's Guide), and operators (see Operator's Guide), can be used dynamically to modify the number of conversational or batch tasks.

UPDTUSER -- Update User Tables (O)

The UPDTUSER macro instruction causes the data pertaining to external storage that is currently in each user table in the SYSUSE data set to be updated with information from the various user catalogs and DSCBs.

Name	Operation	Operand
[symbol]	UPDTUSER	[mode]

mode
specifies whether all or select user entries in the SYSUSE data set are to be updated. Select users are those users with currently active tasks and those users owning shared data sets that are currently being accessed.

Specified as: A - all
S - select

Default: A

Initialization: A DCLASS macro instruction with the PRIVILEGED option must be coded in a CSECT prior to issuing UPDTUSER. If more than one DCLASS macro instruction is issued in a module, the last DCLASS issued prior to coding UPDTUSER must be issued with the PRIVILEGED option.

Execution: UPDTUSER updates the cumulative page count fields in the user table data set (SYSUSE) by extracting the information from each user's catalog and each referenced data set's format-E DSCB. Temporary public data sets are erased, the total number of pages assigned to each user table is changed to reflect the values indicated by their DSCBs, and the temporary and external storage allocation fields are updated.

If mode S is specified, only those entries whose users were active at the time of issuing UPDTUSER (or, if the system failed and was restarted, users who were active at the time of system failure) are updated. ("Active" here means active task or with a shared data set that was being read. The flag USEADC in the user entry indicates an active user.)

Return Data: A message signifying the completion of the update is written to SYSOUT, a return code is placed in register 15, and control is returned to the issuing program.

Return Codes	Meaning
00	Normal return
04	DSCB error or improper authority code

Example: A privileged system programmer has previously issued an RPS or CVV command, or has decided that many user tables have become obsolete.

User: UPDTUSER

System: Returns the following message to SYSOUT: "nnnn USER TABLE STORAGE ALLOCATIONS UPDATED AGAINST DSCBS."

Programming Notes: Following an RPS or CVV command, an UPDTUSER command or macro instruction should always be issued. UPDTUSER may be issued without a preceding RPS or CVV.

UPDTUSER facilitates the conversion from an old user table entry DSECT to a new one.

If the user table is suspected or known to be in error, issuing UPDTUSER causes the current catalog and DSCB information to be placed in the user table.

If the user table is up to date except for active users, which may be true following a system failure and restart, the use of mode S speeds the updating.

Any temporary public data sets are deleted by issuing UPDTUSER.

When the user table of the task issuing UPDTUSER is itself updated, the shared virtual storage of that user table is updated to correspond to the updated SYSUSE record.

USAGE -- Display Resource Usage (S)

This macro instruction obtains accounting data that has been accumulated for a user.

USAGE is described in Assembler User Macro Instructions, except for the following information that is applicable only to system programmers (authority codes O or P).

userid
: specifies the address of a location containing the userid of the user for whom the accounting data is requested. (A nonprivileged user may obtain only his own accounting data; a system programmer may obtain the accounting data of any user.) The userid at the specified location must contain one to eight alphameric characters, the first character must be alphabetic, and the userid must be delimited by X'27'.

    Specified as: A relocatable expression, or, if the address is first loaded into the specified register, in register notation.

    Default: The issuer's userid will be used.

USELOCK -- Lock User Table Entry (O)

The USELOCK macro instruction is used to lock the virtual memory copy of a user table entry.

Name	Operation	Operand
[symbol]	USELOCK	

Note: There are no operands.

Initialization: The program issuing the USELOCK macro must have previously set up base registers for task common (CHATCM) and the user table entry (CHAUSE).

Execution: The USELKCNT is loaded into general register 15 to control the number of time slice ends that will be issued. The lock byte is then tested with a TS instruction. If successful the task id is moved from task common to the user table entry and processing continues. If the TS instruction was unsuccessful, a time slice end is issued, the count in general register 15 is decremented, and the lock byte is tested again. When the count goes to zero, processing continues as if the TS instruction had been successful.

VDMER -- VAM Data Management Error Recovery (S)

The VDMER macro instruction provides an error exit for attempting recovery or issuing diagnostic messages when error conditions arise while processing VAM data sets. If used conversationally, VDMER issues diagnostic messages and returns to the user's terminal without terminating his task. If executed nonconversationally, diagnostic messages are written to the SYSOUT device, and the task is terminated.

Standard form:

Name	Operation	Operand
[symbol]	VDMER	dcb address,message id,flags

L- and E-forms:

Name	Operation	Operand
[symbol]	VDMER	dcb address,message id,flags,MF={L|(E,list)}

<u>Note</u>: A symbol is required in the name field of the L-form. An operand omitted from the L-form must be specified in the E-form; an operand specified in the L-form is overlaid by the same operand in the E-form.

dcb address
specifies the address of the data control block (DCB) for the data set in error.

<u>Specified as</u>: A relocatable expression or register notation (2 through 12). If register notation is used, the address must first be loaded into the specified register.

message id
specifies the address of the second word of a parameter list that contains the identification number of the diagnostic message for the error condition. If there is variable data to be supplied for the message, pointers to the variable inserts follow the message ID, and a one-byte count of these pointers will precede the message ID (see below).

C	AAAA	AAAA	$P^1P^1P^1P^1$	$P^2P^2P^2P^2$	P^0...

C = One-byte count of pointers (may be zero).

A = Eight-character message ID. This doubleword is addressed by word 2 of the parameter list.
$P^1, P^2, \ldots P^0$ = Four-byte pointers to variable data, if any.

<u>Specified as</u>: A relocatable expression or register notation (2 through 12). If register notation is used, the address must first be loaded into the specified register.

flags
specifies the address of a two-byte field where: byte 1 indicates the type of error that occurred, and byte 2 indicates additional information about the error. The flags and their meanings are:

<u>Byte 1</u>	<u>Meaning</u>
'10'	EODAD or SYNAD condition
'20'	Clear last operation flag

<u>Byte 2</u>	<u>Meaning</u>
'0A'	Called by one of the OPEN modules CZCOA, CZCPZ, CZCOP
'0C'	SDST error in CZCOA
'0E'	Non-VAM data set in CZCOZ

<u>Specified as</u>: A relocatable expression or register notation (2 through 12). If register notation is used, the address must first be loaded into the specified register.

<u>Execution</u>: VDMER closes the data set that is causing the error. If it was open, a temporary close (CLOSE, type T) is issued on all the data sets associated with that task. Diagnostic messages are written to the

NA	Add one or more new aliases.
NAR	Add one or more new aliases and return duplicate aliases.
R	Replace the user data associated with a member and close the member.
U	Replace the user data associated with a member but do not close the member.
D	Delete a member from the data set; the directory entries for the member and all of its aliases are deleted and the space occupied by the member is made available for subsequent use.
DA	Delete one or more aliases.
C	Change the name of a member.
CA	Change the name of an alias.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: A member may not be subsequently referred to by the same data control block after a type-N or -R STOW until a FIND of that member is again requested, since these types of STOW close the member.

STOW abnormally terminates the task if any conditions are discovered that make continuation impossible.

Programming Notes: Only type-R STOW is permitted on a shared data set opened for input. The format of the user area used by the STOW macro instruction depends on the type of STOW requested. It is the user's responsibility to construct the area and pass the address of the area to STOW in the user area operand of this macro instruction. The area requirements are:

Types N and U: The area must be at least 12 bytes long and begin on a fullword boundary.

bytes	8	4	N
	Name	N	User Data

Name - Eight-character member name

N - Number of bytes of user data ($0 \leq N \leq 510$)

User Data - Contains the variable data supplied by the user. The data is stored in the POD and can be retrieved by means of the FIND macro instruction.

Types NA, NAR, and DA: The area must be at least 20 bytes long and begin on a fullword boundary.

bytes	8	4	8	8		8
	Member Name	M	Alias 1	Alias 2	....	Alias M

Member Name - Name of the member to which the aliases are linked or are to be linked.

M - Number of aliases to be added or deleted.

Aliases - The aliases to be added or deleted.

Note: Type-NAR STOW causes duplicate aliases to be stored in a page provided by a GETMAIN macro instruction. The first word of this page contains the count of duplicate aliases. The STOW macro instruction places the address of this page in register 0 before exiting.

Type D: The specified area must contain the member name that is to be deleted. It is eight bytes long. When a member name is deleted, all of its aliases are also deleted.

bytes	8
	Member Name

Type C: The name of the member and the name to which it is to be changed are in this area (16 bytes).

bytes	8	8
	Member Name	New Member Name

Type CA: The area specified must be 24 bytes long.

bytes	8	8	8
	Member	Old Alias	New Alias

Member - The eight-character name of the member with which the old alias is associated.

Old Alias - The eight-character alias being changed.

New Alias - The eight-character alias being used for the replacement.

Type R: If any user data is specified, the length must be four bytes longer than the length of the data and the area must begin on a full-word boundary. The additional four bytes are required to specify the length of the specified data.

bytes	4	N
	N	User Data

N - Number of bytes of user's data to be placed in the POD (a number from 1 to 510).

User Data - Contains the variable data supplied by the user. The data is stored in the POD, and can be retrieved by means of the FIND macro instruction.

The user must have exclusive access to a member in order to issue type-C or type-D STOW; that is, he must have opened the data set with an OPEN option that causes the member to be write-interlocked.

Member interlocks are released by CLOSE (referring to the same DCB that caused the interlock to be set), type-R STOW, or a subsequent FIND.

Rules for sharing VPAM data sets are also given in Appendix K.

Return Data: After execution of the STOW macro instruction, bits 24 through 31 of register 15 contain one of the following hexadecimal codes indicating the status of the operation. The user should examine this code to determine the course of action.

Code	Meaning
00	Successful completion of STOW
04	New name or alias is already in use (N, NA, NAP, C, or CA)
08	Member name is not in POD (U, D, DA, or CA)
10	Old member name is not in POD (C); alias is not in POD (DA); old alias is not in POD (CA)
14	Invalid type STOW requested (STOW out of range, member name FFFFFFFF specified, input area not on a fullword boundary, or STOW NA and alias count 0)
18	User data exceeds maximum length of 510 bytes.

SYSIN -- Obtain Input Line From SYSIN or the Source List (S)

The SYSIN macro instruction either prompts the user's SYSIN device for an input line, or it reads a line from the Source List. The input line may consist of a message, a command, or data.

Standard form:

Name	Operation	Operand
[symbol]	SYSIN	input line area,length of input line area, [source code],[prompt character][,exit address]

L- and E-form:

Name	Operation	Operand
[symbol]	SYSIN	[input line area],[length of input line area], [source code],[prompt characters][,exit address], MF={L|(E,list)}

Note: A symbol is required in the name field of the L-form. The parameters specified in the E-form will overlay those specified in the L-form of the macro instruction. The E-form may not specify more operands than are specified in the L-form.

input line area
: specifies the address of a user storage area in which the SYSIN macro instruction is to store the requested input line. No boundary requirements exist for this operand.

  Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, in register notation (2 through 12); in the E-form only, as an RX address.

length of input line
: specifies the address of a fullword that specifies the number of bytes in the user's input line area. If the requested input message is too long for the specified storage area, it is truncated on the right. The four low-order bits of register 15 contain a return code of X'04' if truncation occurs. If the requested input line length is less than the number of bytes in the user's area, the contents of this fullword are replaced by the actual number of bytes transmitted.

  Specified as: Same as the first operand.

source code
: specifies the source from which the input line is to be obtained and the location to which it should be transmitted.

  Specified as: A one- or two-byte code. The first byte serves as the source code and the second byte, if present, indicates that commands are to be transmitted to the user's input line area. If only the source code is specified (that is, the second byte is left blank), the transmittal location is as indicated under the various source codes. The source and location codes are:

Source Code	Meaning
L	Obtain the input line from the Source List (created by the system Command Analyzer routine) and return normally if the line contains a message or data. If the input line is a command, a return is made to the specified exit address without obtaining the command. If the input line is a command but no exit address is specified, a return is made to the caller with a return code of X'0C'. If there is no data to be processed, the return code is X'40'.
G	Obtain the input line from SYSIN and, if it contains an input line or data, return normally. If the message is a command, transfer the command to the Source List, but do not transmit it to the user-specified storage area. If the input line is a command but no exit address is specified, transfer the command to the Source List and terminate the program normally.

E	Obtain the input line from either SYSIN or the Source List, depending on the setting of the SYSINX parameter established in the user's profile by previous issuance of a DEFAULT command. The value of the SYSINX parameter in the user profile might have been previously established as either G, L, or E. If the source operand is defaulted when issuing the SYSIN macro instruction, the source code existing in the user profile establishes the actual default code. SYSINX is initially set to G by the system.
Location Code	
S	This code may be used as a suffix to any of the first three codes, but may not be used by itself. It modifies the action of the code to which it is suffixed by causing commands to be transmitted to the user's input line area just as ordinary data or message input would be.

Default: E

prompt characters
specifies a special command prompt character or string that is to be issued at the user's terminal to prompt the user to enter an input line. The indicated prompt string should be preceded by one byte containing the string length.

Specified as: Same as the first operand.

exit address
specifies the address that is to receive control if the requested message is a command and the source operand is not specified with the S as a suffix. This operand is not valid if the source operand is specified as LS, GS, or ES. This operand must be specified if the source code operand is not specified with the S suffix.

Specified as: Same as the first operand.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: When the input line is read, it is examined to determine if it contains commands, an input message, or data. If an input message or data was read from SYSIN or the Source List, the input line is transferred to the user-specified input line area and execution of the user's program continues. However, a user may have entered a command or command statement in response to the prompting produced by execution of the SYSIN macro instruction. If the reply from SYSIN or the Source List is a command or command statement, it is not transmitted to the user-specified input area unless the suffix S appears in the code. Instead, the SYSIN routine passes control to the user-indicated exit address. At his exit address, the user can then examine the commands by searching the Source List, and either execute them immediately and continue processing, or execute them further on in his program.

A user can alter the action of the SYSIN routine by entering the system prompt character (an underscore) following the SYSIN macro instruction routine's prompt string when it is written out at the terminal. If commands are entered in this manner, they are executed immediately and the SYSIN routine returns a code of X'0C' in register 15 to the user.

If the suffix S is used, commands are transmitted to the user's input line area, just as data normally is.

If a line is requested from the Source List when the latter is empty, the message is obtained from SYSIN instead.

Return Data: The hexadecimal return codes placed in register 15 when control is returned to the user are shown in Figure 16.

When a normal return is made, the total number of bytes transmitted to the user area is passed to the user in the area in which he indicated the maximum message length. If source code L is specified and a null line (that is, a zero-length record) is detected as data, SYSIN completes successfully and returns an indication of the zero length in the maximum message length area.

Bits 16-23 Code	Significance
0	Input record contains no continuation code; record is therefore complete.
1	Input record contains a continuation code; issue another SYSIN to obtain next portion of record.
Bits 24-31 Code	
0	Task is nonconversational; normal return made.
4	Record truncated (exceeded maximum input length specified by the user).
8	Attention interruption occurred.
C	An immediate command (a command preceded by the control language prefix character, normally an underscore) was detected and executed. To resume execution, a nonpriviliged program must issue a GO command; if a privileged program issued SYSIN, the program cannot be resumed and it must be reinvoked.
10	Task is conversational; SYSIN received from terminal keyboard or source list; normal return made.
20	Task is conversational; SYSIN received from terminal card reader; normal return made.
40	SYSIN request not processed.

Figure 16. Return Codes from SYSIN macro instruction

Note: Various combinations of the above return codes may also be received. For example, a return code of X'14' indicates that the task is conversational, input is from the keyboard, and the record is truncated.

An example of L- and E-form use is:

```
              .
              .
              .
    SUE     SYSIN     INAREA,LENGTH,G,PMPT,EXITEND,MF=L
            SYSIN     LENGTHB,,PMPTB,,MF=(E,SUE)
              .
              .
              .
```

When the E-form of this macro instruction is executed, the length of the input line area and the prompt character operands (LENGTH,PMPT) specified in the L-form are replaced in the parameter list by the length and prompt character (LENGTHB,PMPTB) specified in the E-form.

Example: Execution of the following example causes the prompt characters 'ENTER ID' to be displayed at the user's terminal, and his reply to be read from the terminal and transmitted to the input line area labeled INAREA. The number of bytes transferred to INAREA is placed in the LENGTH field specified by the user. When the SYSIN routine returns control to the user's program, register 15 contains a return code of X'10'.

The example is:

```
                 .
                 .
                 .
         MVC     LENGTH,LCON
         SYSIN   INAREA,LENGTH,G,PROMPT,EXITADR
                 .
                 .
EXITADR  RETURN
                 .
                 .
         DC      AL1(L'PROMPT)                   LENGTH OF PROMPT STRING
PROMPT   DC      C'ENTER ID'
LCON     DC      A(L'INAREA)                     LENGTH OF INAREA
LENGTH   DC      F'0'
INAREA   DS      CL20
```

TRUNC -- Truncate an Output Buffer (R)

The TRUNC macro instruction (for QSAM) causes the current output buffer to be regarded as filled. The next PUT macro instruction will use the next block to hold a logical record.

Name	Operation	Operand
[symbol]	TRUNC	dcb address

dcb address
  specifies the address of the data control block opened for the output data set.

| Specified as: Register notation (1 through 12), or an RX address.

CAUTION: A TRUNC macro instruction will be ignored if used with unblocked records, or when a buffer is full, or if it immediately follows another TRUNC macro instruction.

The TRUNC macro instruction is meaningful only with format-F and -V blocked records. Its use with format-F blocked records means that the data set cannot be considered to contain standard blocks. When the data set is read, the RECFM operand of the DCB macro instruction must not contain an S.

Programming Notes: Any exceptional condition resulting from the execution of a TRUNC macro instruction causes control to be passed to the user's synchronous error exit (SYNAD) routine.

| If a TRUNC is issued on a data set opened for UPDAT (see the OPEN
| macro), the following GET retrieves the first logical record from the next block. The last block is written out, including all logical records read plus those not updated by a PUTX.

If a TRUNC is issued before the first PUT of a data set, the TRUNC macro instruction is ignored.

TTIMER -- Test Interval Timer (O)

The TTIMER macro instruction indicates the time remaining in the interval requested by a previous STIMER macro instruction and, optionally, cancels a previously specified timer interval.

Name	Operation	Operand
[symbol]	TTIMER	{TASK|REAL}[,CANCEL][,TNO={timer number}]

TASK
: specifies a TASK interval, as specified in the associated STIMER macro instruction and as identified in the exit list specified in the STIMER macro instruction.

  Specified as: TASK

REAL
: specifies a REAL interval, as specified in the associated STIMER macro instruction and as identified in the exit list specified in the STIMER macro instruction.

  Specified as: REAL

CANCEL
: specifies that the identified interval is to be canceled. If the interval expired before the TTIMER macro instruction was executed, the CANCEL operand has no effect.

  Specified as: CANCEL

  Default: If this operand is omitted, processing continues with the unexpired portion of the interval still in effect.

TNO=
: specifies the number of the programmed interval timer to be tested. Nonprivileged programs may test timers 0 to 15. Clocks 8-15 may be tested but they cannot be canceled; clock numbers over 15 are considered invalid.

  Specified as: A number.

  Default: If this operand is omitted or invalid, timer 0 will be assumed for nonprivileged programs.

Return Data: When control is returned to the user program, one of the following return codes is placed in register 15.

Code	Meaning
00	Successful completion.
04	Invalid clock number was specified.

The time remaining in this interval is returned in register 0, whether or not the interval is canceled.

The remaining time appears as a 32-bit unsigned binary number in which the least significant bit has a value of 1 millisecond. The interval is returned in this form even if the interval was originally specified in decimal digits. If the interval expired and the event has already been dispatched before the TTIMER macro instruction was issued, a zero is returned in register 0.

Initialization: This macro instruction cannot be assembled in a privileged module unless the most recently issued DCLASS macro instruction in the assembly specified USER (see Appendix M) or if the DCLASS option is USER by default.

USAGE -- Display Resource Usage (S)

The USAGE macro instruction causes the user's resource statistics, which are accumulated during his use of the system, to be displayed in an area defined in his program.

Standard form:

Name	Operation	Operand
	USAGE	user area address[,user identification]

L- and E-form:

Name	Operation	Operand
[symbol]	USAGE	[user area address][,user identification] ,MF={L|(E,list)}

Note: A symbol is required in the name field of the L-form. Any operand that is not specified in the L-form must be specified in the corresponding E-form of the macro instruction.

user area address
specifies the address of the area in a user program where accumulated accounting statistics can be recorded for subsequent user reference. The user area should be 400 bytes.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, as in register notation (2 through 12); in the E-form only, as an RX address.

user identification
specifies the address of the user's identification code. When specified, the user ID must be coded elsewhere as one to eight alphameric characters, with X'27' following the last character. The first character must be alphabetic. A nonprivileged user must always specify his own user ID. Privileged users may specify any user ID (see System Programmer's Guide, GC28-2008 for further information).

Specified as: See the first operand.

Default: The current user identification.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: If a user current task attempts to use more system resources than have been allowed him in his User Limit Table, the user task is abnormally terminated.

<u>Programming Notes</u>: The system maintains a master resource scheduling table (SYSULT) which controls the amounts of system resources any one user is allowed to employ. Each user has a unique User Limit Table made available to him when he is joined to the system. The user's resource usage statistics are maintained in this table by the system.

USAGE records statistics for device allocation, number of pages of permanent or private storage allocated, total number of active tasks, CPU time, connect time, and bulk input/output operations performed by the user. These statistics reflect both the current amounts of various system resources the user has assigned to him at any point in a task as well as the accumulated statistics relating his total usage of various system resources since he was joined to the system. These statistics could be useful to programmers in determining the efficiency of various sections of their programs and for recording budgeting statistics required by management.

Information recorded in the user specified area can be examined by his program and, if desired, printed out on the SYSOUT device, in the following format:

```
TEMP STOR=ration;current;accum/PERM STOR=ration:current:accum/DA DEV=
ration;current/MAG TAP=ration;current;accum/PRINTERS=ration/
current;accum/RD PUN=ration;current;accum/TSS TASKS=ration;current/
BULKIN=accum/BULKOUT=accum/CPU TIME=ration:current;accum/CONN
TIME=ration;current;accum
```

where the printed values represent:

PERM STOR TEMP STOR	ration	= number of pages available for user's data sets
	current	= number of pages currently occupied by user's data sets
	accum	= accumulated number of pages times number of seconds they have been held to date
DA DEV MAG TAP PRINTERS RD-PUN	ration	= number of devices of this type available to the user
	current	= number of devices currently assigned to user
	accum	= number of such devices multiplied by the number of seconds for which they were assigned to the user
BULKIN BULKOUT	accum	= accumulated number of bulk input/output operations since the user was joined to the system
CONN TIME	ration	= maximum amount of time that the specified user can be connected to the system from a terminal (hhh:mm:ss)
	current	= number of hours, minutes, and seconds since the user logged on (hhh:mm:ss)
	accum	= total of all connect time during accounting period (hhh:mm:ss)
CPU TIME	ration	= maximum amount of CPU execution time permitted to tasks associated with this user identification (hhh:mm:ss)
	current	= number of minutes, seconds, and milliseconds of CPU time used since the user logged on (mm:ss:ms)
	accum	= number of hours, minutes, and seconds of CPU time since the user was joined to the system (hhh:mm:ss)

Examples: The user wants to display his resource usage statistics at some point in his program. He records those statistics in 400 contiguous bytes starting at STATAREA and then prints them out on the SYSOUT device with two successive GATWR macro instructions. GATWR prints the message statistics out in the format indicated under "Programming Notes."

```
               LOGON JONES
                 .
                 .
                 .
                (process data)
                 .
                 .
                 .
               USAGE      STATAREA,MYID         Can default MYID to EJBID1
               GATWR      STATAREA,LENGTH
               GATWR      SECHALF,LNGTHSEC
                 .
                 .
                 .
MYID           DC         C'EJBID1'
               DC         X'27'                 End of user ID
LENGTH         DC         F'200'
STATAREA       DC         200X'00'              Up to 400 bytes of statistics
SECHALF        DC         200X'00'              may be recorded in the user area
LNGTHSEC       DC         F'200'
```

USATT -- Give User Control of Attention Interruptions (O)

The USATT macro instruction allows the user to have his own routine process attention interruptions from the SYSIN device.

Name	Operation	Operand
[symbol]	USATT	

Note: There are no operands.

Initialization: This macro instruction cannot be assembled in a privileged module unless the most recently issued DCLASS macro instruction in the assembly specified USER (see Appendix M) or if the DCLASS option is USER by default.

Programming Notes: The user must first issue the SAEC and SIR macro instructions to establish the routine that is to process attention interruptions. He then issues the USATT macro instruction to disable the system attention interruption-handling routine, and all subsequent attention interruptions are processed by the user-specified routine. However, if no routine has been established, the attention interruption is lost and the user may not be able to reenter his program.

Once the user gains control of attention interruptions by issuing a USATT macro instruction, control can be returned to the system by using a CLATT, EXIT, CLIC, CLIP, PAUSE, or COMMAND macro instructions. If the user program issues a CLIC, CLIP, PAUSE, or COMMAND macro instruction, the system regains control of attention interruptions until a RUN command (without an operand) is issued. When a CLATT or EXIT macro instruction gives control of attention interruptions to the system, issuing a RUN command does not automatically return control of interruptions to the user. In this case, he can only regain control by issuing another USATT macro instruction in this program.

If the SAEC macro instruction is used to set up user control of attention interruptions, the DCB parameter must be specified as SYSINDCB.

If a user is using USATT to handle attentions, the attention handling routine must include a TCNTRL TYPE=RESTART (see Appendix N) macro, or the user's default for the implicit operand ATTNMODE must be OLD; if not, then a terminal lockout will occur (see Appendix I).

VCCW -- Define a Virtual Channel Command Word (O)

The VCCW macro instruction (for the IOREQ facility) generates a doubleword, the virtual channel command word, that contains the proper information to inform the IOREQ macro instruction of the I/O activity requested.

Name	Operation	Operand
[symbol]	VCCW	command code,data address,count [,(flag,[SIL][,SKP])]

command code
: specifies the hexadecimal command code. This expression's value is placed, right-aligned, in byte 1 of the VCCW doubleword.

Specified as: An absolute expression that specifies the hexadecimal command code, or the code itself enclosed in apostrophes. The codes are:

Command Code	Hexadecimal Command Code
WRITE	01
READ	02
NOP	03
SENSE	04
TIC	08
READBK	0C

data address
: specifies the data address (see the Programming Notes) of the VCCW to be generated (one word).

Specified as: A relocatable expression.

count
: specifies the count (see the Programming Notes) of the VCCW to be generated (two bytes).

Specified as: An absolute expression.

flag
: specifies which flag is to be set in the VCCW to be generated

Specified as:
CD - Chain data flag
CC - Chain command flag
SCC - Software command chaining flag (see "Programming Notes" below)
IOC - IORCB chaining flag
NCC - Indicates no command chaining

Default: CC

SIL
: specifies an additional flag (the suppress length indicator flag) to be set in the VCCW.

Specified as: SIL

Default: No additional flag is set.

SKP
specifies an additional flag (the skip flag) tc be set in the VCCW.

Specified as: SKP

Default: No additional flag is set.

Programming Notes: A virtual channel command wcrd (VCCW) is a doutle-word located on a doubleword boundary with this forrat:

```
Byte 0 - channel command
Byte 1 - flag byte

          Bit 0 CD chain data flag
              1 CC chain command flag
              2 SIL suppress length indicator flag
              3 SKP skip flag
              4 SCC software command chaining flag
              5 IOC IORCB chaining flag[1]
              6 Reserved
              7 Reserved

Bytes 2-3 binary count field of instruction
Eytes 4-7 address in virtual storage
```

Generally, each START I/O instruction issued by a user causes one I/O operation to be executed. The one I/O operation can consist of one VCCW, or a list of VCCWs chained together by the chaining data or chaining command flag bits.

When chaining data, one START I/O instruction executes a list of VCCWs that are chained together by the CD flag bit. The channel command in the first VCCW is executed and the data being processed is placed in storage under control of all of the remaining VCCWs in the chained list. The command codes in the remaining VCCWs are ignored.

When chaining commands, one START I/O instruction is used to execute a list of VCCWs chained together by the CC flag bit. Each VCCW in the list has a command code that is used to control a different channel operation on the same device. The command codes that are chained need not be the same. For instance, it is possible to do a write-backspace-read combination with a magnetic tape unit by chaining commands with three VCCWs. A single START I/O instruction will execute all three commands as one I/O operation. During command chaining, an I/O interruption does not occur at the end of each VCCW executed. When the last command in the chain has been executed, an I/O interruption occurs.

Although I/O interruptions normally occur at the end of the I/O operation (that is, whether single or chained VCCW operations), the software command chain bit (SCC) can be set in a VCCW within a chained list of VCCWs to cause an I/O interruption prior to the end of the I/O operation. When a VCCW is fetched with its SCC bit set, the system receives an I/O interruption as soon as it can be accepted, regardless of whether or not the VCCW with the SCC bit set has completed its execution. If it has not yet been completed, execution of the VCCW list resumes with that VCCW; if it has been completed, execution of the list resumes with the next sequential VCCW. The software command chain provides a user with a

----
[1]See "Programming Notes", under "IOREQ."

convenient way of noting the progress of an I/O operation when command chaining is being employed.

An I/O request involving a small amount of data may use the data buffer in the IORCB; this is called nonbuffered IOREQ. I/O requests (such as to read a card deck) that require longer data areas obtain buffers to contain the data; this is called buffered IOREQ.

A list of VCCWs generated by use of the VCCW macro instruction may be used to inform the IOREQ macro instruction what I/O activity is requested.

The VCCW list of a program that enters IOREQ through the nonprivileged entry point must not refer to pages of different protection classes; if IORCB chaining is in effect (if the IOC bit is on in one of the VCCWs of each of the chained lists but the last), then all pages referred to by all of the lists must have the same protection class.

Restrictions: The VCCW list must conform to the following rules:

1. If any VCCW in the VCCW list has the SCC flag set,
   a. The last instruction to be executed must be the last instruction in the VCCW list. This is accomplished by having this instruction the only instruction in the list other than a TIC which does not have a CD, CC, or SCC flag set.
   b. The last instruction in the list must not be a TIC.
   c. Only the last instruction may have the IOC flag set.
2. If no VCCW in the VCCW list has the SCC flag set,
   a. An instruction executed in the VCCW list, other than a TIC, that does not have the CD or CC flag set is the last instruction executed.
   b. The last instruction in the list may have the IOC flag set only if it is the last instruction in the list to be executed.
3. The last instruction in the VCCW list must not have the CD, CC, or SCC flag set.
4. If a VCCW has the CD flag set, the following VCCW need not have the same command code.
5. No VCCW may have a count field of 0 unless it is a TIC.
6. The address of a VCCW incremented by the VCCW count field must not cross a page boundary.
7. The entire VCCW list must not refer to more than eight different pages of storage.
8. The VCCW list requests the supervisor to allocate space for executing a particular VCCW when an IOREQ macro instruction is issued.
   a. In buffered IOREQ, all commands and data must be contained in one IORCB.
   b. In nonbuffered IOREQ, all commands and page lists must be contained in the IORCB.

9. When IORCB chaining is requested, the IOC flag must be set on the last VCCW of the list (generally a NOP). This command must be the last command in the list to be executed.

If there is a question as to whether a VCCW list requires too large an area, an IOREQ macro instruction may be executed and the return code tested.

VSEND -- Send Message to Another Task (P)

The VSEND macro instruction sends information to another task.

Name	Operation	Operand
[symbol]	VSEND	

Note: There are no operands.

Execution: The SVC 240 resulting from a VSEND macro instruction must be embedded in a message control block (MCB) and be the subject of an EXECUTE instruction.

The receiving task is alerted to the message by a task-external interruption. When the external interruption is accepted, the resident supervisor moves the MCB into the recipient task's ISA. No more than 2040 bytes can be transmitted. If the receiving task's intertask message flag (TSIMB) is on, it does not wish to receive messages. If the sending task's identification indicates that it belongs to the system operator or the batch monitor, the receiving task gets the message (that is, the pending task-external interruption) in any event. If the sender is neither the batch monitor nor the system operator and the receiver's intertask message flag is 1, register 15 is set to 4, telling the sender that his message was not accepted. If the recipient task cannot be found, register 15 is set to 0. If the message is sent, register 15 is set to 3. If and when the message is accepted by the recipient task, and if the reply flag in the senders MCB is on, the complete bit in the message event control block pointed to by the sender's MCB is set to 1.

Example: suppose you want to send the message 'THIS IS A TEST' to a task whose task identification is 1273. You might write:

```
ANY      EX       0,MCB+4
         B        UPUPAWAY
MCB      DS       0D            DOUBLE WORD BOUNDARY
         DC       X'02'         NUMBER OF DOUBLEWORDS OF MESSAGE TEXT
         DC       X'00'         FLAG BYTE
         DC       X'00'         RETURN CODE FOR MECB
         DC       X'00'         MESSAGE CODE
                                nonprivileged programs = 0-127
                                IBM privileged programs = 128-236
                                installation privileged
                                programs = 237-255
         VSEND
         DC       H'0'
         DC       X'1234'       OUR TASK ID
         DC       X'1273'       TASK ID OF RECIPIENT
         DC       A(ECB)        ADDR OF MESSAGE EVENT CONTROL BLOCK;
                                IF NO MECB IS BUILT INTO THE ISSUING
                                PROGRAM, THIS FIELD WOULD BE A(0).
         DC       CL15'THIS IS A TEST.'
```

WRITE -- (VISAM) Write a Selected Record (S)

The WRITE macro instruction (for VISAM) moves a selected record from a user-specified area to an output buffer. The system then includes the record in the output data set, either by key or retrieval address. This macro instruction may be used to update a record or add to the data set. When the write operation is completed, processing of the user's program continues.

Standard form:

Name	Operation	Operand
[symbol]	WRITE	decb name,type,dcb address,work area address, record key

L- and E-form:

Name	Operation	Operand
[symbol]	WRITE	decb name,type,[dcb address],[work area address], [,record key],MF={L|E}

Note: A symbol is required in the name field of the L-form. If an operand is not specified in the L-form, it must be specified in the corresponding E-form of the macro instruction.

decb name
: specifies the name to be assigned to the data event control block (DECB) constructed as part of the expansion of this macro instruction. (Refer to Appendix B, Figure 18 for an illustration of the DECB.)

  Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic); in the E-form only, also in register notation (1 only).

type
: specifies the type of WRITE operation.

  Specified as: One of the following codes:

  KR - WRITE replace by retrieval address ⎫
  ⎬ -for updating
  KS - WRITE replace by key ⎭

  KT - WRITE a record with a new key -for adding a record

dcb address
: specifies the address of the data control block opened for the data set being processed.

  Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also in register notation; in the E-form only, also as an RX address.

work area address
: specifies the address of the user's work area from which the record is to be written. It is the user's responsibility to place the record key in the work area before issuing this macro instruction. The address of this record key is specified in the record key operand.

<u>Specified as</u>: See the dcb address operand.

record key
specifies the address of the field containing either:

the record key, the length of which is indicated in the data control block, or

a retrieval address, a four-byte field on a fullword boundary, originally obtained from DCBLPA field of the DCB.

<u>Specified as</u>: See the dcb address operand.

<u>Initialization</u>: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

<u>CAUTION</u>: A WRITE replace by retrieval address (type KR) must not be used with a shared data set.

Exceptional conditions resulting from the execution of a WRITE macro instruction cause control to be passed to the user's synchronous error exit (SYNAD) routine. In this case, the general registers and the exceptional condition fields in the data control block are set as shown in Appendixes B and F.

<u>Programming Notes</u>: WRITE releases any page-level interlocks set for the data set as a result of executing macro instructions referring to the same data control block. Rules for sharing VISAM data sets are given in Appendix K.

<u>L- and E-Form Use</u>: The L-form macro instruction results in a macro expansion consisting only of a parameter list (DECB). The format of the DECB is described in Appendix B.

The E-form macro instruction results in a macro expansion consisting of only executable instructions. The E-form macro instruction uses the DECB built for it by the L-form macro instruction. Only MF=E should be written for the MF operand in the E-form, because it is the DECB symbol which names the parameter list of the L-form. Any E-form parameter replaces the corresponding parameter in the DECB.

<u>WRITE -- (BSAM) Write a Block (S)</u>

The WRITE macro instruction (for BSAM) writes a block of data from virtual storage to a physical sequential data set. To allow the I/O operation to be overlapped with processing, the WRITE macro instruction returns control to the user's program before the output operation is complete.

Standard form:

Name	Operation	Operand
[symbol]	WRITE	decb name,SF,dcb address,work area address,length

L- and E-form:

Name	Operation	Operand
[symbol]	WRITE	decb name,SF,[dcb address],[work area address] [,length],MF={L|E}

Note: A symbol is required in the name field of the L-form. Any operand that is omitted from the L-form must be supplied with the E-form of the macro instruction.

decb
specifies the name to be assigned to the data event control block (DECB) constructed as a part of the expansion of this macro instruction. (Refer to Appendix B, Figure 18 for an illustration of the DECB.)

Specified as: A symbol (one to eight alphameric characters, the first of which must be alphabetic); in the E-form only, also in register notation (1 only).

SF
specifies sequential forward writing of the block as part of the data set.

Specified as: SF

dcb address
specifies the address of the data control block opened for the data set being processed.

Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also in register notation (2 through 12); in the E-form only, also as an RX address.

work area address
specifies the address of the area in virtual storage that contains the block of data to be written. The user must construct the record-length information in front of each block of format-V records.

Specified as: See the first operand.

length
specifies, for format-U records, the number of bytes to be transmitted. If this parameter is specified for format-F or format-V records, it is ignored. For format-F blocks, the length value is obtained from the DCBBLK field of the data control block. For format-V blocks the length value is obtained from the first two bytes of the output area (LL).

Specified as: 'S', in which case the maximum block length for the data set is used, as specified in the data control block; as an absolute expression; in the standard and E-form, in register notation (2 through 12).

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix F). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

CAUTION: Abnormal termination occurs if:

1. A WRITE macro instruction is issued with record length longer than a track, unless track overflow is specified in the DCB macro instruction.

2. The data control block specified is not validly opened.

3. The DECB specified is already in use by a previous READ or WRITE macro instruction; that is, it has not been checked.

4. An attempt is made to issue a WRITE macro instruction, causing the number of unchecked READ and WRITE macro instructions to exceed the DCBNCP parameter in the data control block.

5. An attempt is made to write on a data set opened for INPUT or RDBACK.

6. An attempt is made to write on a file-protected tape when the data set was opened with DISP=OLD.

Programming Notes: The WRITE macro instruction returns control before actual transmission of data is completed. To determine whether a write operation has been completed, the CHECK macro instruction must be issued for the DECB for the WRITE operation. The DECB employed for the write operation and the virtual storage the block occupies must not be altered or used until the CHECK macro instruction is issued for that DECB.

If a track selected by a WRITE macro instruction is flagged as defective, an alternate track is automatically used. For any device, the operator is notified automatically if any intervention is required to complete the operation.

| If the data set has been opened for UPDAT (see the OPEN macro), the following considerations apply.

* The WRITE macro instruction returns a block to a physical sequential data set residing on a direct access device. The data set must be opened with the UPDAT option. Only the most recently read block can be updated and returned.

* The update mode is provided only for data sets on direct access devices. Although it is not necessary to update and return each block, the sequence of operations for those blocks that are updated must be:

```
    .
    .
    .
READ                    Block A
    .
    .
    .
CHECK                   Await completion of read
    .
    .
    .
```

```
update block in storage
    .
    .
    .
WRITE                   Block A
    .
    .
    .
CHECK                   Await completion of write
```

Thus, only the block last read, or its replacement, can be returned to the data set. Two READ macro instructions can be issued without an intermediate WRITE; this causes the first block to remain unchanged on the device.

Return Data: After an error causing abnormal termination, the eight sense bytes used to store information pertaining to disk or tape devices are saved in DECB sense bytes 0 through 7. DECB byte 1 should be set to X'02' to have these bytes put into the DECB. (See Appendix B, Figure 18.)

L- and E-Form Use: The L-form macro instruction results in a macro expansion consisting of only a parameter list (DECB). The format of the DECB is described in Appendix B.

The E-form macro instruction uses the DECB built for it by the L-form macro instruction. Only MF=E should be written for the MF operand in the E-form, because it is the DECB symbol that names the parameter list of the L-form. Any E-form parameter replaces the corresponding specified optional or required parameter in the DECB.

Example: The proper use of a WRITE macro instruction for format-U records is shown. A data event control block is constructed as part of the in-line macro expansion. A WRITE operation is to be performed from AREA to the data set defined by DCBOUT. Eight-hundred data bytes are to be transmitted for a format-U record (for formats-V or -F, the length parameter would be ignored).

```
EX1     WRITE     ADECB,SF,DCBOUT,AREA,800
```

WT -- Write a Data Set on Tape for Off-Line Printing (S)

The WT macro instruction edits and writes the specified VSAM or VISAM data set on magnetic tape in nonconversational mode for subsequent off-line printing and, optionally, erases it from the catalog when writing is finished. The output is written on 9-track tape in odd parity with standard OS, OS/VS labels. Each input data set record is written on tape in proper format for off-line printing, as a logical record or as a print line; records are blocked, if requested. The maximum blocked record length is 32,767 bytes.

Standard form:

Name	Operation	Operand
[symbol]	WT	{ address of operand string 'data set name 1,data set name 2, [volume number],[blocking factor], [starting byte],[ending byte] [,{EDIT {skips,[h],[lines],[P]}][,{ERASE|Y|N}]' }

L-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
symbol	WT	'data set name 1,data set name 2, [volume number],[blocking factor], [starting byte],[ending byte] [,{EDIT|skips,[H],[lines],[P]}] [,{ERASE|Y|N}]',MF=L

E-form (see "Operand Strings" in Part II, Section 1):

Name	Operation	Operand
[symbol]	WT	address of operand string,MF=(E,list)

address of operand string
    specifies the address of the first operand in the operand string.

    Specified as: Register notation (2 through 12) or a relocatable expression. Note that the operand string can also be specified in the macro operand as a character string enclosed in parentheses, as shown.

data set name 1
    specifies the name of the data set to be written on tape in print format. The data set name either must previously have been defined by a DDEF macro instruction or command, or must be in the catalog.

    Specified as: The fully qualified name of a nonpartitioned data set or a nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number).

data set name 2
    specifies the data set name under which the data set is to be cataloged as it resides on the output tape. The user must specify the name or the task will be abnormally terminated.

    Specified as: The fully qualified name of a nonpartitioned data set or a nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number).

volume number
    specifies the volume ID number of the output tape.

    Specified as: One to six alphameric characters.

    Default: A scratch tape is used.

blocking factor
    specifies the blocking factor of the output tape.

    Specified as: A one- to three-digit number; the maximum blocking factor permitted is 246.

    Default: 30

starting byte
    specifies the byte number at which tape writing is to start for each data set logical record.

Specified as: A one- to six-digit number.

Default: Writing starts with the first block of each logical record.

ending byte
: specifies the byte number at which printing is to stop for each data set record. This end byte is written.

  Specified as: A one- to six-digit number.

  Default: Writing continues to the last byte of each logical record or until the printer line length is reached, whichever occurs first.

EDIT
: indicates that the line spacing is controlled by a control character in the first byte of each data set logical record. This control character is user-supplied and may be in ASCII or machine code, but must be in the same code throughout the data set (Refer to Appendix D.)

skips
: specifies the number of lines to be skipped between records.

  Specified as:
  1 - indicates skip 1 line
  2 - indicates skip 2 lines
  3 - indicates skip 3 lines

  Default: 1

H
: specifies that the first logical record of the data set is to be repeated on each print page as a header line. The first 132 bytes or the first record, whichever is smaller, is to be used as the header.

  Specified as: H

  Default: The first record is not repeated.

lines
: specifies the number of lines to be printed on a page. The maximum number of lines per page is determined by the printer form used for the off-line printing of the data set. If not specified, 54 lines are printed on each page.

  Specified as: A one- or two-digit number.

  Default: 54

P
: specifies that page numbering is to be performed.

  Specified as: P

  Default: No page numbering is performed.

ERASE|Y|N
: specifies the disposition of the cataloged data set after the tape-writing operation is complete.

Specified as:

ERASE or Y - erase the cataloged data set after the tape-writing operation is complete.

N - do not erase the cataloged data set.

Default: N

Note: If ERASE or Y is specified for a shared data set that is currently being used by another user, a diagnostic message is issued and the data set is not erased.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: When the user executes a WT macro instruction for a data set defined in his task, the data set is released and disconnected from the user's task. The WT macro instruction processes data sets that were created by using either the virtual sequential or virtual index sequential access method. The tape data set is created by using the basic sequential access method. This output tape is written in odd parity with standard OS, OS/VS Labels. The data set must be in the catalog. If it is not cataloged the user's task is terminated. The ERASE option can be used to erase the data set after writing is completed.

If a data set to be written on tape was created via the DATA command, the first byte of each record contains an indicator for the origin of the record. Unless the starting byte operand is specified, this byte is written as part of the record upon issuance of the WT macro instruction. In such a case, if the record was originally entered through a card reader, the indicator byte will be written as a C. If it was entered through a terminal, the byte will be written as a blank character. When the starting byte operand is specified as 2 or greater, the indicator byte is bypassed and is not included as part of the written record.

No more than one print line can be written from a single data set record. If a read error occurs, the record in question is written in hexadecimal form on SYSOUT.

Return Data: At completion of execution of the WT macro instruction, register 1 contains the address of the batch sequence number assigned to the nonconversational task established by this macro instruction; the low-order byte of register 15 contains one of the codes given below.

Code	Significance
00	WT request was accepted.
All other codes	Register 15 contains a two byte system message number.

Examples: In EX1, the operand string is presented as a character string. In EX2, a symbolic address designates the address of the operand string.

```
EX1      WT         'OLDNAME,NEWNAME'
EX2      WT         TAPTAG
```

Since EX2 is given as an address, the user has provided the operand string at location TAPTAG. When the macro instruction is executed, the necessary alphameric characters must be available in the string.

WTL -- Write to Log (S)

The WTL macro instruction writes a message in the system log. If specified in the system operator's user profile, the message is also written on the main operator's console.

Standard and L-form:

Name	Operation	Operand
[symbol]	WTL	'message text'[,MF=L]

Note: A symbol is required in the name field of the L-form. If the MF operand is omitted, MF=I is assumed.

E-form:

Name	Operation	Operand
[symbol]	WTL	MF=(E,list)

message text
: specifies the message to be inserted in the system log and, if specified in the system operator's profile, written on the main operator's console. The message can include commas, blanks, and apostrophes, as in a character constant. The maximum message length is 255 bytes, including the required enclosing apostrophes.

  Specified as: The text of the message itself, enclosed in apostrophes.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: Parameter list use by the WTL macro instruction is not standard: register 1 contains the address of the message (rather than of an area that contains the address of the message). Refer to the second example under the WTC macro instruction.

Return Data: At completion of execution of the WTL macro instruction, the low-order byte of register 15 contains one of the following codes:

Code	Significance
00	Successful completion
04	Attention interruption
0C	Invalid message length; no message sent.

Example: The message ALL ON is to be sent to the operator and entered in the system log.

```
     EX1      WTL        'ALL ON'
```

WTO -- Write to Operator (S)

The WTO macro instruction writes a message on the main operator's console.

Standard and L-form:

Name	Operation	Operand
[symbol]	WTO	'message text'[,MF=L]

Note: A symbol is required in the name field of the L-form. If the MF operand is omitted, the standard form is assumed.

E-form:

Name	Operation	Operand
[symbol]	WTO	MF=(E,list)

message text
: specifies the message to be written on the operator's console. The message can include commas, blanks, and apostrophes as in a character constant. The maximum message length is 255 bytes. The message includes the required enclosing apostrophes.

  Specified as: The message itself, enclosed in apostrophes.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: Parameter list use by the WTO macro instruction is not standard: register 1 contains the address of the message (rather than of an area that contains the address of the message). See the second example below.

Return Data: At completion of execution of the WTO macro instruction, the low-order byte of register 15 contains one of the following hexadecimal codes:

Code	Significance
00	Successful completion
04	Attention interruption
0C	Invalid message length; no message sent.

Examples: In the following example, the message NOW COMPLETE is to be sent to the operator.

```
    EX1     WTO              'NOW COMPLETE'
```

The following example illustrates the use of the WTO and WTL parameter lists necessitated by the nonstandard use of parameter lists involved in these macro instructions:

```
      .
      .
      .
    WTOLEN  DC               A(L'WTOTXT)
    WDOTXT  DC               C'MESSAGE TEXT'
      .
      .
      .
            WTO              MF=(E,WTOLEN)
      .
      .
            WTL              MF=(E,WTOLEN)
```

WTOA -- Write to Operator with Action Message (S)

The WTOA macro instruction writes Operator Action messages (messages requiring some type of action from the operator) on the main operator's console.

Standard and L-form:

Name	Operation	Operand
[symbol]	WTOA	'message text'[,MF=L]

Note: A symbol is required in the name field of the L-form. If the MF operand is omitted, the standard form is assumed.

E-form:

Name	Operation	Operand
[symbol]	WTOA	MF=(E,list)

message text
: specifies the message to be written on the operator's console. The message can include special characters such as commas, question marks, and apostrophes. The maximum message length, including the required enclosing apostrophes, is 255 bytes.

Specified as: The message itself, enclosed in apostrophes.

The message, when displayed on the operator's console, is preceded by a series of three pointers (minus sign followed by greater-than sign) which appear as three arrows pointing from the margin towards the message.

Initialization: If this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M). Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: Parameter list use by the WTOA macro instruction is not standard: register 1 contains the address of the message (rather than of an area that contains the address of the message).

Return Data: At completion of execution of the WTOA macro instruction, the low-order byte of register 15 contains one of the following hexadecimal codes:

Code	Significance
00	Successful completion
04	Attention interruption
0C	Invalid message length; no message sent

Example: In the following example, the message WHAT IS ADMINISTRATOR'S EXTENSION NUMBER? is to be sent to the operator.

```
     EX1     WTOA     'WHAT IS ADMINISTRATOR''S EXTENSION NUMBER?'
```

This example will appear on the operator's console as:

```
        ->->-> WHAT IS ADMINISTRATOR'S EXTENSION NUMBER?
```

Note: The pointers are provided by the WTOA macro instruction.

WTOR -- Write to Operator with Reply (S)

The WTOR macro instruction writes a message on the system operator console and enables the system operator's reply to be transmitted to the program issuing the macro instruction. No further processing of the program occurs until the operator replies.

Standard form:

Name	Operation	Operand
[symbol]	WTOR	'message text',reply area address,reply length

L-form:

Name	Operation	Operand
symbol	WTOR	'message text',[reply area address] [,reply length],MF=L

Note: A symbol is required in the name field of the L-form.

E-form:

Name	Operation	Operand
[symbol]	WTOR	[reply area address][,reply length],MF=(E,list)

Note: Any operand that is omitted from the L-form must be supplied with the E-form of the macro instruction.

message text
: specifies the message to be written on the console. The message can include commas, blanks, and apostrophes as in a character constant. The maximum message length is determined at system generation. This length must not exceed the physical line length on the console output device or 253 characters, whichever is less. The message appearing on the console does not include the enclosing apostrophes.

: Specified as: The text of the message itself, enclosed in apostrophes.

reply area address
: specifies the address of an area into which the message reply text should be placed.

: Specified as: In the standard and L-form, as a relocatable expression; in the standard and E-form, also in register notation (2 through 12); in the E-form only, also as an RX address. |

reply length
: specifies the length, in bytes, of the reply text. The value must not exceed 255.

: Specified as: An absolute expression; in standard and E-form, also in register notation (2 through 12). The value specified may not exceed 255.

Initialization: If this macro instruction is to be executed in a privileged module the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED (see Appendix M.) Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming Notes: At completion of execution of the WTOR macro instruction, the low-order byte of register 15 contains one of the following hexadecimal codes:

Code	Significance
00	Successful completion
04	Attention interruption
0C	Invalid message length; no message sent.
10	Reply length greater than specified maximum reply length: reply was received, but only the maximum number of characters is in the reply area.

Example: The message IS B. R. SMITH JOINED? is written on the operator's console. The expected reply is three bytes long ("yes" or "no") and will be stored at location ALPHA.

```
     EX1      WTOR      'IS B. R. SMITH JOINED?',ALPHA, 3
```

XTRTM -- Extract Accumulated CPU Time (O)

The XTRTM macro instruction enables you to extract and examine the total CPU time used by your task.

Name	Operation	Operand
[symbol]	XTRTM	

Note: There are no operands.

Execution: The address of the task's XTSI is obtained by the SVC processor. The accumulated time is computed by subtracting the current timer value from the last time slice value and adding the accumulated time value to the difference.

Return Data: The total accumulated CPU time (in milliseconds) of the issuing task is computed and returned to the task in register 1.

Example: If you want to extract your task's accumulated time, you might write:

```
          TIME   XTRTM
```

APPENDIX A: EXIT LIST (EXLST)

The EXLST= operand of the DCB macro instruction may be specified only when using BSAM or QSAM data organization.

The exit list consists of a series of codes and addresses that inform the system of the location of a user-supplied exit.

Each entry in the list consists of three contiguous words aligned on fullword boundaries.

```
                          1 byte
                        ┌─────────┬──────────────────────────────────────┐
                        │         │                                      │
Word 1                  │  code   │              Unused                  │
                        │         │                                      │
                        ├─────────┴──────────────────────────────────────┤
                        │                                                │
Word 2                  │  V-con of routine                              │
                        │                                                │
                        ├────────────────────────────────────────────────┤
                        │                                                │
Word 3                  │  R-con of routine                              │
                        │                                                │
                        └────────────────────────────────────────────────┘

                        <──────────────────4 bytes─────────────────────────>
```

Entries do not have to be in order by code. To indicate the last entry in the exit list, the high-order bit of the byte containing the code is turned on. Since there are only five codes, the list should contain a maximum of five entries.

If an exit routine is in the same assembly module as the exit list, two A-type address constants should be coded for words 2 and 3 of the entry.

Code (hex)	Meaning
00	Entry ignored, i.e., not active
01	User routine to process user header labels
02	User routine to create user header labels
04	User routine to create user trailer labels
05	User DCB exit routine
07	User routine to handle input end-of-volume
08	User routine to handle output end-of-volume
8X	Signal last entry in list, where X is 0-8 representing 00-08 codes above

	TRAILER LABELS[2]		HEADER LABELS[2]	
	Exit 03 INPUT	Exit 04 OUTPUT	Exit 01 INPUT	Exit 02 OUTPUT
<u>OPEN</u>				
INPUT			X	
OUTPUT				X
UPDAT			X	
INOUT			X	
OUTIN				X
RDBACK			X	
<u>end-of-volume</u>				
INPUT	X		X[1]	
OUTPUT		X		X[1]
UPDAT	X		X[1]	
INOUT (READ)	X		X[1]	
(WRITE)		X		X[1]
OUTIN (READ)	X		X[1]	
(WRITE)		X		X[1]
RDBACK	X		X[1]	
<u>FEOV - CLOSE</u>				
INPUT				X[1]
OUTPUT		X		X[1]
UPDAT				X[1]
INOUT (READ)				
(WRITE		X		X[1]
OUTIN (READ)				
(WRITE)		X		X[1]
RDBACK				X[1]

[1]Exit not taken if: (1) current volume in process is last volume, or (2) data set is to be closed

[2]If last I/O operation was backward, user header label routine is invoked for trailer label processing and user trailer label routine is invoked for header label processing

Figure 17. Conditions upon exit -- routine entries

COMMON CONDITIONS ON ENTRANCE TO EXIT ROUTINES

On entrance to any user exit routine, register 13 will contain the address of a save area which may be used in linking to other routines. The 19th word of the save area will contain the R-con specified in the exit list. The contents of the rest of the save area will be unpredictable. Register 14 will contain a return address; its contents should be restored before issuing the RETURN macro instruction. Register 15 will contain the address of the entry point to the exit routine and may be changed by the exit routine according to the individual exit routine instructions outlined below. The contents of the remaining registers (except for registers 0 and 1 as outlined below) will be unpredictable.

<u>CAUTION</u>: Type III linkage is used to link to the nonprivileged user's exit routine. The exit routine is restricted to type I linkage in linking to other subroutines, and thus cannot issue macro instructions that require other types of linkage, such as GATWR and OBEY.

REQUIREMENTS FOR INDIVIDUAL TYPES CF EXIT ROUTINE

Code 01: User Routine to Process User Header Labels

When this routine is entered, register 1 contains the address of the data control block being processed, and register 0 contains the address of an 80-byte buffer that contains a user header label. The first four bytes of the buffer contain the characters UHL1 to UHL8 or UTL1 tc UTL8 depending on which of the eight permissible user header labels or trailer labels is being processed. To obtain the next label,the user issues a RETURN (no operands except RC= are allowed) with a hexadecimal 04 in the low-order byte of register 15. When the last label is processed, the user issues a RETURN macro instruction with a hexadecimal 00 in the low-order byte of register 15. The user may not issue data management macro instructions for this data set in this routine.

Code 02: User Routine to Create User Header Iabels

When this routine is entered, register 1 contains the address of the data control block being processed; register 0 contains the address of an 80-byte buffer in which the user is to build a label. The first four bytes already contain the characters UHL1 to UHL8 depending on which of the eight permitted user header labels is being created. These four bytes must not be altered. The user places information in bytes 4-79. Issuing a RETURN macro instruction, with a hexadecimal 04 in the low-order byte of register 15, causes the label to be written, and requests control to be returned to this routine so another label may be created. Issuing a RETURN macro instruction, with a hexadecimal 00 in the low-order byte of register 15, causes the last label to be written, and control is not returned to this routine. The user may not issue data management macro instructions to this data set in this routine.

Code 03: User Routine to Process User Trailer Labels

Same characteristics as Code 01.

Code 04: User Routine to Create User Trailer Labels

Same characteristics as Code 02, except the characters UTL1 through UTL8 are substituted for UHL1 through UHL8.

Code 05: User DCB Exit Routine

When this routine is entered, register 1 contains the address cf the data control block being opened. The user may alter fields in the data control block, if desired. To return control to OPEN, the user issues a RETURN macro instruction (no operands except RC= are permitted) with a hexadecimal code of 00 in the low-order byte of register 15.

Code 07: User Routine to Handle Input End-of-Vclume

Same characteristics as Code 01.

Code 08: User Routine to Handle Output End-of-Volume

Same characteristics as Code 02, except the characters UTL1 through UTL8 are substituted for UHL1 through UHL8.

EXIT-LIST EXAMPLE

The following is an example of the coding of an exit list. The exit list must be in the same assembly module as the data control block (DCB macro instruction) which refers to it.

```
         DC       0F              ALIGN TO FULLWORD BOUNDARY
APPLE    DC       X'02'
         ADCON    IMPLICIT,EP=MHDRLAB
         DC       X'03'
         ADCON    IMPLICIT,EP=PHDRLB
         DC       X'01'
         ADCON    IMPLICIT,EP=PLABY
         DC       X'85'
         ADCON    IMPLICIT,EP=ALTER
```

The symbolic name of this exit list is APPLE. To use it, EXLST= APPLE must be written in the DCB macro instruction. Note that the high-order bit of the hexadecimal code for the last entry is on, indicating the last entry in the exit list.

APPENDIX B: SYNCHRONOUS ERROR EXIT ROUTINE (SYNAD)

When using BSAM, QSAM, VISAM (either for VISAM data sets or VISAM members of VPAM data sets), or IOREQ macro instructions, it is possible that errors may result from an attempt to process data; in many cases, certain remedial actions are available to the user.

If desired, a routine may be written for the purpose of receiving control from the system when an error occurs. The conditions that cause control to be given to the SYNAD routine are described under each macro description.

The user indicates to the system that a SYNAD routine is supplied by writing the keyword parameter SYNAD= in the DCB macro instruction. The task is terminated if an error occurs that would normally cause SYNAD to be entered and no SYNAD was supplied.

The following is a list of suggested actions to be taken in a SYNAD routine:

1. Issue a RETURN macro instruction, which causes a record to be accepted with error ignored (BSAM and QSAM only).

2. Set flags that are meaningful to the program.

3. Close the data set.

4. Resume processing at another point in the data set.

5. Call another routine.

6. Terminate the program.

Entry To SYNAD During BSAM or QSAM Operations

If BSAM or QSAM is being used, the contents of the general registers upon entry to the SYNAD routine are as follows:

Register	Bit	Usage
0	0 thru 31	Address of data event control block (DECB).
1	0	Set to 1, if error was caused by a READ macro instruction (for BSAM), or by GET or RELSE macro instructions (for QSAM)
	1	Set to 1, if error was caused by a WRITE macro instruction (for BSAM), or by PUT, PUTX or TRUNC macro instructions (for QSAM)
	2	Set to 1, if error was caused by a BSP, CNTRL or POINT macro instruction (for BSAM), or by a SETL macro instruction (for QSAM)
	3	Set to 1, if (1) error indicated by bit 0 did not prevent reading the block; or (2) if error indicated by bit 1 occurred during creation of a new block
	4	Set to 1, if request was illogical; e.g., a POINT macro instruction (for BSAM) or a SETL macro instruction (for QSAM) referred to a block not contained in the data set
	5 thru 31	Not used
2 thru 12		(contents that existed before the macro instruction was executed)
13		SYNAD R-con value (for BSAM), or address of service routines save area (for QSAM)*.
14		Return address
15		The address of the entered SYNAD routine
*For QSAM the nineteenth word of the save area pointed to by register 13 contains the PSECT address (R-con) for the SYNAD routine.		

If the BSAM user specifies a SYNAD routine in his DCB, the routine is invoked for all errors. BSAM error codes are returned in the one-byte field of the DECB, DECID. If the DECID code is less than X'40', as many retries as wanted may be attempted. If the code is greater than or equal to X'40', but less than X'80', one retry is allowed: if that retry results in a code of less than X'40', an unlimited number of retries can then be attempted; if the retry results in a code greater than or equal to X'80', no further retries are allowed. If the DECID code is greater than or equal to X'80', and a retry is attempted, an ABEND (compcode 1) occurs. All error conditions that cause I/O retry at an alternate path are posted again after they are retried until they are successful (DECID=X'00'), or until all paths fail (DECID=X'8X'). For some error conditions, the user has control over retry by using the IMSK in the DCB. The hexadecimal return codes are:

Code	Meaning
04	Unit exception during read
05	Unit exception during read backwards
06	Unit exception during write
07	Tape at load point
08	Disk end of cylinder
12	Data check
14	Chaining check
15	Overrun
20	Incorrect length for length specified by LRECL, BLKSIZE and RECFM of DCB
21	Incorrect length for length in physical record
24	Tape data converter check
30	Error not retried due to DCB IMSK
40	File protected tape
42	Device not ready -- intervention required
43	Tape unit non-existent -- intervention required
46	Control unit and/or drive cannot read NZRI tape
50	Channel control check
51	Interface control check
52	Channel data check
54	Bus out check
55	Equipment check
56	Seek check
57	Missing address marker
5C	SIO failure
5D	Sense failure
5E	Invalid sense information
60	Program check
61	Protection check
62	Command reject (not file protect)
63	Track condition check
64	Track overrun
65	Unexpected end of cylinder
66	No record found
67	File protected
6C	Invalid status
6D	CCW specification check
6E	SDA not in CHBSDT
70	Purged I/O
71	Reset macro not accepted
80	No path available
94	Chaining check -- device unavailable
95	Overrun -- device unavailable
C3	Intervention required -- device unavailable
D0	Channel control check -- device unavailable
D1	Interface control check -- device unavailable
D2	Channel data check -- device unavailable
D4	Bus out check -- device unavailable
D5	Equipment check -- device unavailable
D6	Seek check -- device unavailable
DC	SIO failure -- device unavailable
DD	Sense failure -- device unavailable
FF	Recursive SYNAD exit -- device unavailable

The EXPLAIN command may be used to obtain an explanation of any of the above messages; issue the following at the terminal:

```
EXPLAIN DECID=xx,CZCRC
```

where xx is the DECID code to be explained.

The DECB begins on a fullword boundary; its format is shown in Figure 18.

Byte	Bit	Usage
0	0	Always set to 0
	1	Completion flag; set to 1 when an I/O event is completed
	2 thru 31	(Used by the system)
1		BSAM flags
4 and 5		Type field
6 and 7		Length field
8 thru 11		Data control block address
12 thru 15		Area addresses
16 thru 19		Pointer to status indicators
20 thru 25		(Used by the system)
26		Sense byte 0
27		Sense byte 1
28 and 29		(Used by the system)
30	1	Permanent error flag
31		(Used by the system)
32 thru 39		Channel status word
40 thru 47		Sense bytes 0 through 7

Figure 18. Data event control block (DECB)

BSAM flags
indicates whether more than 2 sense bytes are needed. A X'02' indicates the need for 8 extra sense bytes, which will be used for information about disk or tape devices.

Caution: If the user is building his own E-form DECB and specifies X'02' in byte 1, an 8-byte field must be added to the original 40 bytes of the DECB.

Type field
contains a numeric value representing SF (for READ or WRITE) or SB (for READ).

Length field
contains a binary number that represents the number of bytes in a block, or an indicator that the maximum block size specified in the data control block was used.

Data contrcl block address
contains the address of the data control block.

Area address
specifies the I/O area address. For BSAM it contains the address of the high-order byte of an area in virtual storage that is the object of a forward READ or WRITE; or the address of the low-order

byte of an area in storage that is the object of a backward READ operation.

Pointer to status indicators
: contains the address of the status indicators, which are two bytes in the channel status word. If control is passed to the SYNAD routine status information (i.e., sense byte 1, sense byte 2, and the channel status word) is arranged as indicated above. Each of these status indicators is described in detail below.

Channel status word
: is a doubleword illustrated below:

STORAGE PROTECTION KEY	COMMAND ADDRESS	STATUS BYTE 1 (unit)	STATUS BYTE 2 (channel)	COUNT-FIELD

The first six bits of sense byte 1 and all bits of status bytes 1 and 2 are device-independent. Their meaning is as follows:

Bit	Sense Byte 0	Bit	Status Byte 1	Bit	Status byte 2
0	Command reject	0	Attention	0	Program-controlled interruption
1	Intervention required	1	Status modifier	1	Incorrect length
2	Bus out check	2	Control unit end	2	Program check
3	Equipment check	3	Busy	3	Protection check
4	Data check	4	Channel end	4	Channel data check
5	Over run	5	Device end	5	Channel control check
		6	Unit check	6	Interface control check
		7	Unit exception	7	Chaining check

Bits 6 and 7 of sense byte 0 and all bits of sense byte 1 are device-dependent. Refer to individual publications on specific devices for interpretation of these bits.

sense bytes 0 through 7
: contain information pertaining to READ or WRITE macro instructions involving tape or disk devices. Refer to individual publications for the specific devices for the number of bytes that are used, and for details of the contents of these bytes.

<u>CAUTION</u>: If any of the bits 4-7 of status byte 2 are on, the system cannot recover. Any subsequent I/O operations on the data set result in abnormal termination of the task.

If the permanent error flag in the data event control block is on, the program must not issue any further I/O operations to the data set.

If SYNAD is invoked because of SETL only, the DCB address and status information in the DECB may be valid. All other fields may contain undefined information.

<u>Entry to SYNAD During VISAM Operations</u>

The SYNAD routine may be entered during VISAM operations (either processing of a VISAM data set or a VISAM member of a VPAM data set). The contents of the general registers upon entry to the SYNAD routine are as follows.

Register	Usage
0	Address of DECB, if error was caused by a READ or WRITE macro instruction. See Table 8
1	Address of the data control block
2 thru 13	(Contents that existed before the macro instruction was executed)
14	Return address
15	Address of the entered SYNAD routine

Additional information concerning the error that may prove useful to a SYNAD routine is found in the data control block fields, DCBEX1 and DCBEX2 (Appendix F).

APPENDIX C: END OF DATA ADDRESS (EODAD)

When using data management services for input data sets, the exact number of records in the input data set need not be known. When the last record of a data set being sequentially processed is accessed, a subsequent attempt to access a record causes the system to transfer control to a specified point in the user's program. For BSAM the transfer is made upon checking the READ macro instruction that requests a block after the last block is accessed. For QSAM the transfer is made when the user issues a GET macro instruction after all the records in the data set have been processed.

The user indicates to the system where control is desired upon the end-of-data condition by writing the keyword parameter EODAD= in the DCB macro instruction. The task is terminated if an EODAD routine is not supplied and an attempt is made to access a record after the last record in the data set. For BSAM, the termination occurs upon issuing the CHECK macro instruction for a READ macro instruction issued after the last block of a data set is accessed.

When the end-of-data routine is entered, the general registers are set as follows:

Register	Usage
0	(Not defined)
1	Address of data control block
2 thru 12	(Same as before the routine was entered)
13	EODAD R-con value (for BSAM), or address of service routines save area (for QSAM)*
15	Address of EODAD routine

*The nineteenth word of the save area pointed to by register 13 will contain the PSECT address (RCON) for the EODAD routine.

All record formats may optionally include a carriage control character in each logical record. This control character is recognized and processed if a data set is being written to a printer or punch. For format-F and -U records this character is the first byte of the logical record. For format-V records it must be the fifth byte of the logical record, immediately following the logical record length field.

Two alternatives are available; i.e., the carriage control character may be in machine code or FORTRAN code. If either option is specified in the data control block, the character must appear in every record.

MACHINE CODE

The user may specify in the data control block that the machine code control character is placed in each logical record. The user-supplied byte must contain the bit configuration specifying a write and the desired carriage or stacker-select operation. Only those commands that include a write are permitted; the independent carriage and stacker select operations are excluded.

The machine code control characters are:

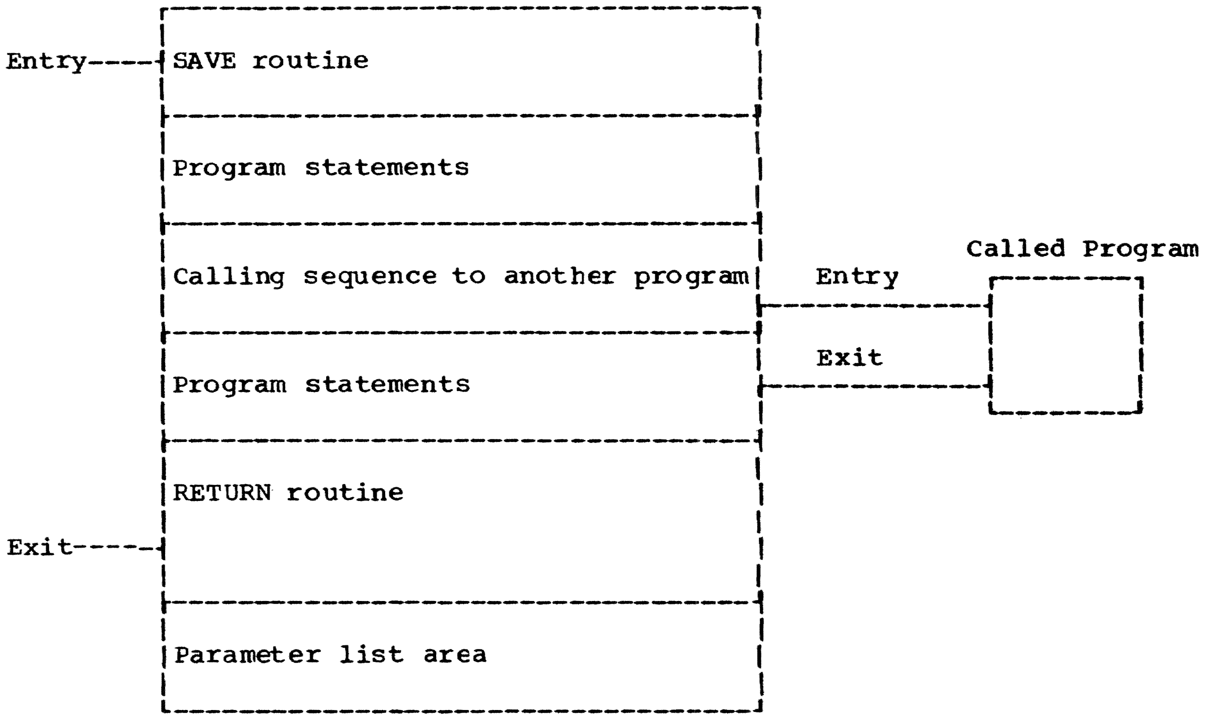
Function	Byte Value (hexadecimal)
Write (no automatic space)	01
Write and space 1 line after printing	09
Write and space 2 lines after printing	11
Write and space 3 lines after printing	19
Write and skip to channel 1 after printing	89
Write and skip to channel 2 after printing	91
Write and skip to channel 3 after printing	99
Write and skip to channel 4 after printing	A1
Write and skip to channel 5 after printing	A9
Write and skip to channel 6 after printing	B1
Write and skip to channel 7 after printing	B9
Write and skip to channel 8 after printing	C1
Write and skip to channel 9 after printing	C9
Write and skip to channel 10 after printing	D1
Write and skip to channel 11 after printing	D9
Write and skip to channel 12 after printing	E1
Note: To obtain the corresponding carriage-control operations (space or skip to channel N) without printing, increase the value of the low-order digit by hexadecimal 2. Example: space two lines - 13; skip to channel 5 - AB; skip to channel 9 - CB.	

FORTRAN CODE

The user may choose to specify FORTRAN (formerly called ASA or USASCII) code rather than machine code. FORTRAN uses the same control characters as the American National Standard Code for Information Exchange, ANSI X3.4-1968, sometimes referred to as ASCII. The code byte must appear in each logical record if this option is chosen, in the position defined above. The FORTRAN control character codes are:

FUNCTION	CHARACTER
Skip no line before printing	+
Skip 1 line before printing	blank
Skip 2 lines before printing	0
Skip 3 lines before printing	-
Skip to channel 1 before printing	1
Skip to channel 2 before printing	2
Skip to channel 3 before printing	3
Skip to channel 4 before printing	4
Skip to channel 5 before printing	5
Skip to channel 6 before printing	6
Skip to channel 7 before printing	7
Skip to channel 8 before printing	8
Skip to channel 9 before printing	9
Skip to channel 10 before printing	A
Skip to channel 11 before printing	B
Skip to channel 12 before printing	C

Linkage conventions govern communication among programs by establishing a standard that permits easy, efficient, error-free branching and linking to a desired program. The following chart summarizes the elements required by an assembler program to be both a calling and a called program:



In TSS, all linkage among programs residing in virtual storage conforms to one of the following three convention types:

- Type I -- Between two nonprivileged or between twc privileged programs.
- Type II -- From a nonprivileged to a privileged program.
- Type III -- From a privileged to a nonprivileged program.

Only the Type I convention is presented in this appendix; Types II and III conventions are described in System Programmer's Guide.

Type I linkage conventions include three basic standards to which the assembler user must adhere:

1. Utilizing the proper registers in establishing a linkage.
2. Reserving a save area in the calling program in which the called program may save the contents of the calling program registers.
3. Reserving a parameter area in the calling program, to which the called program may refer.

Proper Register Use

TSS has assigned roles to certain registers used in generating a linkage. The function of each linkage register is illustrated below. Note that registers 2 through 12 are not used.

General Register	Usage
0,1	Parameter list registers
13	Save area register
14	Return register
15	Entry point register, return code register

It is the responsibility of the called program to maintain the integrity of registers 2-12 so that their contents are the same at exit as they were at entry to the called program. It is the calling program's responsibility to maintain the floating point registers around a call. Registers 0, 1, and 13-15 must conform to the indicated conventions; when using system services (for example, interruption handling), these registers should not be used by the calling program, because their contents may be destroyed.

Reserving a Save Area

Every calling program must reserve an area of storage (save area) in which certain registers (that is, those used in the called program and those used in the linkage to the called program) are saved by the called program.

The minimum amount of storage needed for the save area of a program that is both calling and called is 19 words. Figure 19 shows the layout of the save area and the contents of each word.

A called program that does not call another program need not establish a save area. However, if registers 13 or 14 are used by the called program, that called program should save their contents and restore them before returning control to the calling program.

Reserving a Parameter Area

If a called program requires a parameter list, every program calling it must reserve an area of storage (parameter area) in which the parameter list used by the called program is located. Each entry in the parameter area occupies four bytes at a fullword boundary. If the parameter list is of variable length, the word preceding the first entry contains the length (in words) of the parameter list. Each entry contains the address of an argument to be passed to the called program. The CALL macro instruction may be used to generate the parameter list as well as to link to the called program.

There are two types of linkage available to users of TSS: implicit linkage and explicit linkage. When an explicitly loaded module is no longer needed, it can be deleted explicitly.

SAREA --> (word 1)	Length, in bytes, of the save area and any appendages to it
SAREA + 4 --> (word 2)	Address of the calling program's save area. This field is set by the called program in its own save area
SAREA + 8 --> (word 3)	Address of the next save area; that is, the save area of called program. This field is set by the called program
SAREA + 12 --> (word 4,)	Contents of register 14, containing the address to which return from the called program is made. This field is set by the called program in the calling program's save area
SAREA + 16 --> (word 5)	Contents of register 15, containing the address to which entry into the called program is made. This field is the called program in the calling program's save area
SAREA + 20 --> (word 6)	Contents of register 0
SAREA + 24 --> (word 7)	Contents of register 1
SAREA + 28 --> (word 8)	Contents of register 2
SAREA + 32 --> (word 9)	Contents of register 3
	Eight words containing the contents of registers 4 through 11
SAREA + 68 --> (word 18)	Contents of register 12
SAREA + 72 --> (word 19)	Address of the PSECT for the called program belonging to calling program. This field must be set by the calling program, by storing in it the R-con value of the called program

Figure 19. Save area layout and word contents

Implicit Linkage

Program reference to a V- or R-type address constant (adcon) of an external symbol constitutes a request for implicit linkage. When an undefined external symbol is referred to in this manner, the loader is called to make available, in the user's virtual storage, those modules required to satisfy this external reference. This automatic action requires only specification of external symbols and adcon types as required by the assembler.

Explicit Linkage

Within a given program there may be several references to different subprograms; however, for a given execution of that program, only one of those subprograms might be required. Since dependence on normal implicit linkage would require, in the calling program, the presence of adcons for all such subprograms, some unnecessary overhead would be experienced in preparing the unused adcons for linking.

It is also possible to develop, during program execution, the external name of the module, entry point, or CSECT which is to be explicitly linked. In this case, it may not be possible to specify the modules to be linked during assembly.

To allow for these situations, two explicit functions are provided that retrieve the desired subprogram at object time. The LOAD macro instruction loads the desired program; the explicit CALL macro instruction, in addition to loading the program, establishes the necessary linkage to it.

Explicit Deletion

The DELETE macro instruction makes virtual storage available by dispensing with a program that was explicitly loaded previously but is no longer available. Further, any other programs that are no longer required as a result of the deletion of the specified program are also deleted.

This appendix contains descriptions of the contents of the fields of a data control block and the priority of the various sources for filling those fields, for those who desire to alter data control blocks or interrogate fields for the information contained therein.

Sources for Providing Data Set Attributes

In general, a user writes a source program to create or process data. This data is considered to be a data set. In TSS, the system requires that certain attributes and identification information pertaining to data sets must be available to the system before a user can make use of the special programs and data management facilities comprising TSS.

These attributes can be furnished to the system from two to six different sources depending on whether the data set being processed is a new data set or a data set that has been previously defined to the system. The combined information provided by these sources must provide the system with all the information it requires to begin processing a particular data set. The six possible sources which provide the system with the attributes of a data set are listed below in the order of their priority. Figure 20 indicates the DCB operands applicable to each access method and their valid alternate sources prior to opening that data control block.

Source 1 - The User's Program

The user may alter or fill data control block fields any time after the block has been created by a DCB macro instruction. A DCB macro instruction with no operands merely reserves virtual storage for a data control block, with all its fields containing binary zero. The user has the opportunity to alter fields at OPEN time by specifying the address of a user routine that is to alter the DCB at open time; the routine is specified as the EXLST parameter of the DCB macro instruction or lower priority DDEF macro instruction or command. Any user-coded data control block modification routine will find the DCBD macro instruction very convenient for referencing the fields of the control block.

Source 2 - The DCB Macro Instruction

Information may be supplied to the data control block by specifying operands in the DCB macro instruction. In this case, the DCB macro instruction, in addition to creating a data control block, also fills the specified fields with the attributes indicated via the operands.

Source 3 - The Catalog

At the time a data set is cataloged certain attributes (data set organization, data set disposition, device class, and data set affinity) are recorded in the catalog. When a user desires to re-open that data set for additional processing, information previously recorded in the catalog need not be specified again by another attribute source. If such recorded information is specified again by another attribute source, the previously recorded attribute information will take precedence.

DCB Operand	Specifies	Applicable Access Method						Valid Alternate Sources			
		VSAM	VISAM	VPAM	BSAM	QSAM	IOREQ	User's Program	DEFINE DATA Command	Data Set Label	System Service Routines
DDNAME	Symbolic name identical to that used in ddname operand of DDEF command associated with data set	X	X	X	X	X	X	X			
DSORG	Data set organization	X	X	X	X	X	X	X	X		
RECFM	Record format information	X	X	X	X	X		X	X	X	X
LRECL	Logical record length	X	X	X	X	X		X	X	X	
EODAD	Address of user's end-of-data routine for input data sets	X	X	X	X	X		X			
OPTCD	Optional service desired, write with validity check (for direct access devices only)	X	X	X	X	X		X	X	X	
SYNAD	Address of user's synchronous error exit routine (entered when an uncorrectable error occurs in I/O operation)		X	X*	X	X	X	X			
KEYLEN	Key length		X	X*	X			X	X		
RKP	Displacement of key from first byte of logical record		X	X				X	X		
PAD	Space to be left on each page of virtual index sequential data set (to allow subsequent insertions)		X	X*				X	X		
MACRF	Types of macro instructions used in processing data set (GET, PUT, READ, WRITE, etc.)				X	X		X	X		
DEVD	Device on which data set resides plus, for some device types, device-dependent information (data code, tape density, etc.)				X	X		X	some device dependent information	some device dependent information	
BLKSIZE	Maximum block length				X	X		X	X	X	
IMSK	Number code indicating what system error recovery procedures (if any) are to be invoked				X	X		X	X		X
EXLST	Address of user's exit list				X	X		X			

*(only for VISAM members)

Figure 20. DCB operands, their specifications, access methods, and alternate sources (part 1 of 2)

		Applicable Access Method						Valid Alternate Source			
DCB Operand	Specifies	VSAM	VISAM	VPAM	BSAM	QSAM	IOREQ	User's Program	DEFINE DATA Command	Data Set Label	System Service Routines
NCP	Number of consecutive READ, WRITE, or IOREQ macro instructions issued before, CHECK macro instructions.				X		X	X	X		X
BUFNO	Number of buffers				X			X	X		
BUFOFF	Buffer offset field				X				X		
BFALN	Buffer alignment				X						
BUFL	Buffer length				X			X	X		
EROPT						X		X	X		

Figure 20. DCB operands, their specifications, access methods, and alternate sources (part 2 of 2)

Source 4 and 5 - The DDEF Macro Instruction (4) Or Command (5)

The DDEF macro instruction or command can supply the same information to all fields in the DCB as can be specified via the DCB macro instruction, except for the EODAD, SYNAD, and EXLST parameters. The DDEF macro instruction or command must be used for each data set to be processed because it is the only source of DSNAME, the data set name. The primary difference between the DDEF macro and command is the ability of the DDEF command to provide attribute information from the terminal at execution time rather than at assembly time.

Source 6 - Data Set Labels or Data Set Control Blocks (DSCBs)

At the time a data set is recorded on a storage device, a data set label or DSCB is created. The label or DSCB of an existing data set contains some data control block information. If fields in the data control block are still unspecified at open time, the information is taken from the data set label or DSCB and placed into the Data Control Block.

Priority of Sources

Many of the attributes of a data set that are required by the system can be furnished from more than one of the six possible sources. In such cases, each of the sources providing this information is assigned a priority and the system will use the information from the source with the highest priority. When two or more of the sources have corresponding entries, the attributes in the lower priority sources will be ignored.

This priority scheme provides great flexibility since information omitted in a higher priority source can be supplied by a lower priority source. Thus, if attribute parameters such as DSORG are not specified in the higher level DCB and DDEF macro instructions they may be supplied dynamically, at the terminal, by the lower priority DDEF command, or by the DSCB or tape label.

If a field has been specified in the higher priority DCB or DDEF macro instructions at assembly time or by the user's program prior to OPEN, it will not be possible to modify that field dynamically from the terminal (e.g., if there were a LRECL parameter specification in the DDEF command at the terminal and the DCB also contained an LRECL specification at assembly time, the LRECL specification of the DDEF command would be ignored. In many cases, if a lower priority source provides the same attribute data as a higher priority source but the data provided differs in each source, the system will issue diagnostics indicating this. The system will either assume the higher priority source contains the valid data and continue processing based on that source or it will require the user to issue the proper matching attribute data in the lower priority source. Thus, if a user specifies a data set's organization (DSORG) in a DDEF command for a data set that is already cataloged, it must agree with the DSORG recorded in the catalog or diagnostics will be issued asking the user to reenter the correct data set organization parameter or to default to the system default value. In the latter case, if the user fails to provide the proper information and does not use the system default option the system will abend the user's task.

The fields are presented in alphabetical order and are described in the following format:

```
NAME    (length)                          (name,name)
        specification of contents
```

NAME
    is the keyword parameter name if the field may be supplied by keyword parameter in a DCB macro instruction. If the field is not supplied by keyword parameter, a meaningful name or phrase is given; for example, retrieval address.

length
    specifies the length of the field in bytes.

name
    specifies the symbolic name or names which, when used in conjunction with the DCBD macro instruction, will address the data control block field.

    An X in a bit position means that bit is not tested.

BLKSIZE (2-byte field) (DCBBLKSI,DCBBLK)
    specifies a binary value for the maximum block length in bytes. The maximum value is 32,760.

BUFL (2-byte field) (DCBBUFL, DCBBUF)
    contains a binary number that represents the length, in bytes, of each buffer obtained for a buffer pool. The maximum is 32,760.

BUFNO (1-byte field) (DCBBUFNO, DCBBUN)
    contains a binary number that represents the number of buffers assigned to a data control block. The maximum is 255.

BUFOFF (1 byte field) (DCBBOF, DCBCPT)
    specifies the length of the buffer offset field in bytes for ASCII tapes (may be specified only as a DCB subparameter of the DDEF command). An integer from 0 to 99 may be specified. Acceptable buffer offset values for various record formats and data set dispositions are:

Disposition \ Format	Undefined	Fixed	Variable
NEW (OUTPUT)	0	0	0 or 4
OLD (INPUT)	0 - 99	0 - 99	0 - 99

DDNAME (8-byte field) (DCBDDNAM, DCBDDN)
contains a name of up to eight characters.

DEVD (1-byte field) (DCBDEVD, DCBDEV)

Code	Bit Pattern
DA	11000100
PT	11100111
TA	11100011
PR	11010111
RD	11011001
PC	11010101
no device specified	11010110

The additional keyword operands that are optionally used with DEVD= cause information to be inserted in device-dependent parameters 1 and 2.

Device Dependent Parameter 1 (1-byte field) (DCBDD1)

This byte is used to contain information from the KEYLEN operand that is subordinate to the DEVD= operand of the DCB macro instruction.

KEYLEN (DCBKEYLE, DCBKEY)
contains a binary number that represents the length, in bytes, of the key associated with a physical record. The maximum is 255.

Device Dependent Parameter 2 (1-byte field) (DCBDD2)

This byte is used to contain information from the TRTCH operand that is subordinate to the DEVD= operand of the DCB macro instruction.

TRTCH (DCBTRT)

Code	Bit Pattern
C	00100011
E	00111011
T	00010011
ET	00101011
Odd parity, no translation	00110011

DSORG (2-byte field) (DCBDSORG, DCBDSO)

Code	Bit Pattern
PS	0100000X 00000000
PSU	01000001 00000000
VI	01110001 00000000
VS	01110010 00000000
VIP	01110011 00000000
VSP	01110100 00000000
VP	01110101 00000000

EODAD (8-byte field) (DCBEODVD, DCBEOV) for V-con
(DCBEODRD, DCBEOR) for R-con
contains the address of the user's EODAD routine. The first word contains the entry point address. The second word contains the address of the PSECT for the EODAD routine. If the EODAD routine

has no PSECT, the second word contains the address of the CSECT containing the EODAD routine.

EROPT (1-byte field) (DCBEROPT, DCBERO)

Code	Bit Pattern
ACC	10000000
SKP	01000000
ABE	00100000

Exceptional Condition Field 1 (1-byte field) (DCBEX1)

Error Caused By	Bit Pattern
GET	00000000
PUT	00000100
SETL	00001000
READ	00001100
WRITE	00001111
DELREC	00010100

Exceptional Condition Field 2 (1-byte field) (DCBEX2)

Type of Error	Bit Pattern
Keys equal - sequence error	00000100
Key not found	00001000
Keys out of sequence	00001100
Keys do not coincide	00001111
Keys coincide	00010100
Invalid retrieval address	00011000
Invalid record length	00011100
Position past end of data set	00011111
Position before beginning of data set	00100100
Exceed maximum number of overflow pages	00101000
Exceed maximum size of shared data set	00101100

EXLST (4-byte field) (DCBEXLST, DCBEXL)
contains the address of a user-supplied exit list. The exit list must be in the same CSECT as the data control block.

IMSK (4-byte field) (DCBIMSK,DCBIMK)
contains the system error mask. The bit pattern is as specified under IMSK in the DCB macro instruction.

LRECL (4-byte field) (DCBLRECL, DCBLRE)
specifies for format-F records the length in bytes of a logical record. For BSAM or QSAM the maximum value is 32,760 bytes; for VSAM, 1,048,576 bytes; for VISAM, 4,000 bytes.

MACRF (2-byte field) (DCBMACRF, DCBMAC)

Code	Bit Pattern
G	01000000 00000000
GS	01000001 00000000
P	00000000 01000000
PS	00000000 01000001
R	00100000 00000000
RC	00100010 00000000
RP	00100100 00000000
W	00000000 00100000
WC	00000000 00100010
WP	00000000 00100100

Note: For G[S], P[S] the bit pattern becomes the appropriate combination of the above bit patterns.

For R[C|P], W[|P] the bit pattern becomes the apprcpriate combination of the above bit patterns.

NCP (1-byte field) (DCBNCP)
contains a binary number that represents the number of consecutive READ or WRITE macro instructions that are to be issued before a CHECK macro instruction is given. The maximum is 99.

OPTCD (1-byte field) (DCBOPTCD, DCBOPT)

	Bit Pattern
A - ASCII tape request	XX100000
W - write validity check	10000000
Default: no service is performed	00000000

OPTIONS (1-byte field) (DCBOPI)
contains the bit patterns specified by option parameters in the OPEN and CLOSE macro instructions.

OPEN OPT1	OPEN OPT2	CLOSE OPT	Bit Pattern
INPUT			XX0000XX
OUTPUT			XX1111XX
INOUT			XX0011XX
OUTIN			XX0111XX
RDBACK			XX0001XX
UPDAT			XX0100XX
	REREAD		01XXXXXX
	LEAVE		11XXXXXX
		REREAD	XXXXXX01
		LEAVE	XXXXXX11

PAD (1-byte field) (DCBPAD)
contains a binary number that represents the space, as percentage, left available within the pages of a VISAM data set, providing for insertions within the pages of a VISAM data set. The maximum is 50 (50 percent).

RECFM (1-byte field) (DCBRECFM,DCBREC)

Code		Bit Pattern
U	- undefined	11X00XXX
V	- variable	01XX0XXX
VB	- variable blocked	01X10XXX
D	- variable (ASCII tapes)	1X1XXXXX
DB	- variable (ASCII tapes) blocked	1X11XXXX
F	- fixed	10XXXXXX
FB	- fixed blocked	10X1XXXX
FS	- fixed standard	10XX1XXX
FBS	- fixed blocked standard	10X11XXX
A	- Extended ANSI FORTRAN control characters	XXXXX10X
M	- machine code control characters	XXXXX01X
	- no control characters	XXXXX00X
	- KEYLEN specified in data control block	XXXXXXX1

EXAMPLE: If this byte contains 10010100, the record format is fixed length, blocked records with an ANSI FORTRAN control character.

Retrieval Address for Virtual Access Method (4-byte field) (DCBLPA)

This field contains a retrieval address that is used for recording and repositioning to specified records of a data set.

Retrieval Address for QSAM (6-byte field) (DCBLPDQ)

This field contains a retrieval address that is used for recording and repositioning to specified records of a data set.

RKP (2-byte field)(DCBRKP)
contains a binary number that represents the displacement of the key field of a record from the first byte of the record.

SYNAD (8-byte field) (DCBSYNVD, DCBSYV) for V-con
(DCBSYNRD, DCBSYR) for R-con
contains the address of the user's SYNAD routine. The first four bytes contain the entry point address. The second four bytes contain the address of the PSECT for the SYNAD routine. If the SYNAD routine has no PSECT, the second word contains the address of the CSECT containing the routine.

This appendix describes how the DDEF macro instruction is used to define a private data set or a nonstandard public data set. For information on the definition of standard data sets, refer to the description of the DDEF macro instruction in the main body of this manual. (Standard data sets have virtual sequential organization, are on direct access public storage, and are arranged in units of pages.) Figure 21 lists required and optional operand fields of the DDEF macro instruction. The format of the DDEF macro instruction is as follows:

Standard form:

Name	Operation	Operand
[symbol]	DDEF	{address of operand string|'operand string'}

L-form:

Name	Operation	Operand
symbol	DDEF	'operand string',MF=L

E-form:

Name	Operation	Operand
[symbol]	DDEF	MF=(E,list)

address of operand string
 specifies the address of the first operand of the operand string (see "Operand Strings" in Part II, Section 1).

 Specified as: Register notation (2 through 12), or a relocatable expression. Note that the operand string can also be specified as a character string enclosed in apostrophes, as shown.

operand string
 specifies the operands of the DDEF macro instruction; the operands are shown in Figure 21.

 Specified as: A continuous character string enclosed in apostrophes. The way to specify each operand is described following Figure 21.

```
┌──────────────────────────────────────────────────────────────────────────────┐
│{data definition name} [,data set organization] ,DSNAME={name }               │
│{PCSOUT              }                                   {*name}              │
│┌                                                                    ┐       │
││,DCB=([*data set name][,DSORG=organization][,MACRF=macrc form]       │       │
││      [,BUFL=buffer length][,DEVD=device][,BUFNO=number of buffers] │       │
││      [,BFTEK=buffer technique][,NCP=check number]                  │       │
││      [,RECFM=record format][,OPTCD={W|A}][,LRECL=record length]    │       │
││      [,BLKSIZE=maximum block length][,KEYLEN=key length]           │       │
││      [,PRTSP=spacing][,STACK=stacker bin][,DEN=density tape]       │       │
││      [,MODE=mode of operation][,TRTCH=recording technique]         │       │
││      [,EROPT=error option][,PAD=available space]                   │       │
││      [,RKP=key field displacement][,IMSK=error procedures]         │       │
││      [,BUFOFF=buffer offset field length])                         │       │
│└                                                                    ┘       │
│[,UNIT=({DA[,direct access type]})]                                           │
│       {TA[,tape type]          }                                             │
│       {device address          }                                             │
│                                                                              │
│[,SPACE=({TRK          },primary allocation[,secondary allocation][,HOLD])]   │
│         {CYL          }                                                      │
│         {record length}                                                      │
│                                                                              │
│[,VOLUME=({PUBLIC                  } [,{PRIVATE             } ,...])]         │
│          {PRIVATE                 }   {volume serial number}                 │
│          {volume sequence number  }                                          │
│                                                                              │
│[,LABEL=(file sequence number][,label type][,RETPD=retention period])]        │
│                                                                              │
│[,DISP=data set status][,OPTION={JCBLIB|CONC}]                                │
│                                                                              │
│[,RET=([P|T][C|L][U|R])]                                                      │
│                                                                              │
│[,PROTECT={Y|N}]                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```
Figure 21. Operands for DDEF macro instruction

data definition name
: specifies the symbolic name associated with this data set definition. It provides the link between the data control block in the user's program and the data set definition.

<u>Specified as</u>: One to eight alphameric characters, the first of which must be alphabetic. The user is not allowed to use a name that begins with SYS; the system-reserved names are prefixed with these characters.

PCSOUT
: specifies that the program checkout subsystem is being used and a data set is being defined for dumps. One PCSOUT-type DDEF command or macro instruction is required when the DUMP command is to be employed.

<u>Specified as</u>: PCSCUT

data set organization
: specifies a two-character code that indicates the organization of the data set. It must be specified for non-VAM data sets but is optional for VAM data sets.

<u>Specified as</u>:

PS - QSAM or BSAM (physical sequential access methods)
VI - VISAM (virtual index sequential access method)
VS - VSAM (virtual sequential access method)

VP - VPAM (virtual partitioned access method)
RX - IOREQ (I/O request)

Default: The data set is assigned the type of organization specified at system generation.

DSNAME=
specifies the name of the data set.

Specified as: Name or *Name, where:

name
specifies the name of the data set. This is the name under which the data set may be cataloged or referred to during the task. A relative generation number and/or a partitioned data set member name may be included with the name.

Specified as: The fully qualified name of a partitioned or nonpartitioned data set, a member of a partitioned data set, or a partitioned or nonpartitioned generation of a generation data group (identified by absolute generation name or relative generation number).

For ASCII tape input, the data set name may include any ASCII 'a' character (an alphameric or graphic character), except the underscore and alternate graphics character. The format is DSNAME= 'bname', where the leading blank indicates ASCII characters, or DSNAME=name if only alphameric characters are used.

*name
specifies the name, here prefixed by an asterisk (*), of a data set created under IBM OS or OS/VS. Subsequent references to this data set name do not include the asterisk prefix.

Specified as: same as for the DSNAME=name form, with a maximum of 44 characters, excluding the asterisk.

DCB=
specifies the data control block information, as follows:

data definition name
specifies the data definition name of a previously issued DDEF command or macro instruction. The previous name is prefixed by an asterisk (*) to indicate that the data control block field of that DDEF is to be duplicated for the current DDEF macro instruction or command. Any new suboperands given in the remainder of the field take precedence over the corresponding suboperands of the previous DDEF command or macro instruction.

Specified as: One to eight alphameric characters, the first of which must be alphabetic, preceded by an asterisk.

DCB suboperands
detailed descriptions of the data control block suboperands are given in the discussion of the DCB macro instruction for each access method, and in Appendix F. Those suboperands that cannot also be specified as operands of the DCB macro instruction are described below. The MODE and STACK suboperands are described under "DCB (MSAM) Options" in System Programmer's Guide.

BUFOFF= (BSAM)
specifies for ASCII tapes the length of the buffer offset field in bytes.

Specified as: An integer 0 to 99. See Appendix F for a table of acceptable buffer offset values for various record formats and data set dispositions.

DEN= (IOREQ)
specifies a value for the tape recording density, in bits per inch:

DEN Value	7-Track	9-Track
0	200	error
1	556	error
2	800	800
3	error	1600
4	error	6250

Note: If the data set is or will be on tape, the DEN suboperand of the DCB operand is required to specify the density of the data set. If the data set exists, or if another data set exists on the tape, the density of the existing data set is used, and the DEN suboperand is ignored. If the density of a new data set on an empty tape is not specified, the tape density established at system generation is used, in which case the BUFOFF suboperand is not required.

Caution: The DEN and BUFOFF suboperands may be specified as DCB suboperands only; they may not be specified in the DCB macro instruction.

UNIT=
specifies the type of device needed for the data set. Allowable devices are specified at system generation and, therefore, may be changed. Direct access devices may be specified for either public or private volumes. Tape may be specified for private volumes only.

Specified as:
DA - specifies that a direct access device is required for the data set.

direct access type - specifies the type of direct access device.

Specified as: A four-digit number.

Default: The system selects the type of direct access device, as specified at system generation.

TA - specifies that a tape unit is required for the data set.

tape type - specifies the type of tape required.

Specified as:

7	- any 7-track tape unit
7DC	- 7-track tape unit with Data Converter installed
9D2	- 9-track tape unit with 800 bpi capability
9D3	- 9-track tape unit with 1600 bpi capability
9D4	- 9-track tape unit with 6250 bpi capability

Default: The system selects the type of tape specified at system generation.

device address
specifies the symbolic device address of a nonstandard device.

SPACE=

specifies the direct access storage allocation for the data set. If the entire space field is defaulted, the direct access storage allocation specified at system generation is assigned.

Specified as:

TRK - specifies that the space requirements are expressed as a number of tracks.

CYL - specifies that the space requirements are expressed as a number of cylinders.

record length

specifies the average record length, in bytes, of the physical records.

Specified as: A decimal number not exceeding 32,767.

Default: If the data set organization is SAM, the unit of allocation is assumed to be a cylinder. If the data set organization is VAM, the unit of allocation is assumed to be a page (4096 bytes).

primary allocation

specifies the number of units to be allocated initially to the data set.

Specified as: A 1- to 8-digit decimal number; max=00065535.

Default: The primary space allocation assigned at system generation is assigned.

secondary allocation

specifies the number of units to be allocated each time the space allocated to the data set has been exhausted and more data is to be written.

Specified as: A 1- to 8-digit decimal number; max=00000256.

Default: The secondary space allocation specified at system generation is assigned.

Note: If more than 256 units are requested, only 256 will be allocated.

HOLD

specifies that the unused storage assigned to this data set is not to be released when the data set is closed.

Specified as: HOLD

Default: Unused storage is released.

VOLUME=

specifies the volumes on which the data set resides. This field must always be used when creating a new data set residing on a private volume or when referring to an existing uncataloged data set residing on a private volume. This field also must be used when expanding an existing private cataloged or uncataloged data set. When expanding an existing private cataloged data set, only the new volumes to be added to the data set (options of PRIVATE or volume serial number) need be referred to.

This field is not required for data sets on public volumes. However, this field may optionally be specified for new data sets on public volumes providing only existing public volume serial numbers are specified. Initial space allocation will then be limited to the specified volumes.

Specified as:

PUBLIC - the data set is to be placed on public storage volumes.

PRIVATE - volumes are to be allocated from the system pool (i.e., the scratch tapes or disks available to the system operator). Once assigned, the volume remains the user's, exclusively, until he notifies the system operator that it can be returned to the pool. The user must use this option to request initial or additional scratch volumes for data sets on private volumes.

Specified as: A one-to-four digit number.

volume sequence number - the sequence number of the first volume of the data set to be read or written. It is meaningful only if the data set has SAM organization, is cataloged, and its earlier volumes are not to be processed.

Note: If the volume sequence number is specified, the data set is cataloged and the serial numbers are retrieved from the catalog. If PRIVATE is specified, the system assigns a volume serial number. If the volume sequence number or PRIVATE options are used to extend the volume list for an existing cataloged data set, the volume serial numbers in the catalog are used for the existing portion of the data set, regardless of whether they are also specified in the current DDEF.

volume serial number
specifies the volume serial numbers identifying the volumes on which the data set resides. The volume serial number is required for old uncataloged data sets that reside on private volumes; the user must use this option to specify initial or additional volume serial numbers for data sets on private volumes. It is optional for new data sets on public volumes. For ASCII input data sets, any ASCII 'a' character (alphameric or graphic), except for the underscore and alternate graphic character, may be used.

Specified as: One to six alphameric characters; characters other than alphamerics may also be used, in which case the volume serial number must be enclosed in apostrophes. Example:

```
        VOLUME=(,'A1*%',123456,'#,"123')
```

Default: If the volume sequence number was specified, the data set is cataloged and the serial numbers are retrieved from the catalog. If PRIVATE was specified, the system assigns a volume serial number.

LABEL=
specifies the labeling conventions.

Specified as: See below.

Default: If the entire label field is defaulted, the labeling conventions specified at system generation are assigned. However, if the data set is cataloged, label information is retrieved from the catalog.

file sequence number

specifies the file sequence number of a data set when multiple data sets are on one tape volume.

Specified as: A one- or two-digit decimal number.

Default: The data set is assumed to be the first (or only) one on the tape volume.

label type

specifies either the type of labeling desired or the absence of labels.

Specified as:

NL	- no labels (ASCII or EBCDIC tapes only)
SL	- standard labels
SUL	- standard labels and user labels
AL	- standard ASCII labels
AUL	- standard ASCII and user labels

Default: The system assumes the label type specified at system generation.

RETPD

specifies the retention period of the data set. This operand applies to data sets on direct access volumes or on labeled tapes.

Specified as: A four-digit decimal number that indicates the time period, in days, that the data set is to be retained after its creation.

Default: The retention period is assumed to be zero days, thus allowing immediate rewriting.

DISP=

specifies the status of the data set.

Specified as:

NEW	- for a new data set.
OLD	- for an old data set.
MOD	- the data set exists but is being added to. MOD causes logical positioning after the last record of the data set. It applies only to SAM data sets on private volumes.

Default: OLD - for old cataloged data sets.
NEW - for a new data set or for an old uncataloged private data set.

If DISP is defaulted in a DDEF for an existing cataloged public data set, the system assumes a value of OLD. If DISP is defaulted for any data set that does not yet exist, the system assumes a default value of NEW. It should be noted that, for existing uncataloged private data sets, the DISP value must be explicitly specified as OLD. If the user tries to default such a data set, a DISP value of NEW is assumed and causes a system error.

OPTION=

specifies that either a job library is being defined or a data set is being added to the concatenated data set named in the data definition name operand.

Specified as:

JOBLIB - the data set is to be used as a job library. The data set name specified in the DSNAME field will be entered into the program library list.

CONC - this data set is to be concatenated with one or more data sets whose data definitions have the same data definition name. Only input data sets that are not job libraries can be concatenated. The order of concatenated data sets is the same as the order in which they are defined.

RET= (VAM only)
specifies codes for the storage type, deletion option, and owner access that will be used in processing the data set.

Specified as: ([P|T][C|L][U|R]), where the codes have the following meanings:

Codes	Meanings
P	Permanent storage
T	Temporary storage
C	Delete at CLOSE
L	Delete at LOGOFF
U	Read-write access
R	Read only access

Default: P; If a deletion option is not specified for a temporary-data set (T), L is assumed; U.

PROTECT=
specifies whether the tape being mounted is to have a file-protect ring in. If Y is specified, the tape is to be mounted without a file-protect ring, unless the disposition of the data set being defined requires a ring, in which case DDEF processing is terminated; a tape already mounted with a file-protect ring in it will have to be remounted in order to have the ring removed. If N is specified, the tape is to be mounted with the ring in, regardless of the disposition of the data set being defined; if the tape is already mounted without a ring, it will have to be remounted in order to have a ring put in it.

Specified as:
Y - no ring, file is protected
N - ring in, no protection.

System default: N for DISP=NEW and DISP=MOD. For DISP=OLD the default is left to the option of the installation. If this operand is omitted for DISP=OLD, there is no verification of the file protection status.

The DDEF macro instruction or command that defines any cataloged data set is brief and simple. The only required operand fields are the data definition and data set names. Other operand fields are unnecessary since the organization of the data set is described in its catalog entry.

DDEF macro instructions or commands that define uncataloged data sets may be divided into two groups: those defining new data sets (i.e., data sets that will be generated during the run but do not exist as yet) and those defining old (already existing) data sets. These old uncataloged data sets can exist only on private volumes.

To define a new data set that will be written on a public volume, the user may use the data definition name, data set name, SPACE, data set organization, and LABEL operand fields. Exactly which fields he uses other than the data definition and data set names, which are required, depends on the character of his particular data set.

To define a new data set that will be written on a private volume, the user must give the data definition name, data set name, UNIT, and VOLUME operands. If desired, he may also furnish the data set organization, SPACE, LABEL, and DISP fields.

The user defines an old, uncataloged data set just as it stands on his private volume. To do so, he must use the data definition name, data set name, VOLUME, UNIT, and DISP fields. He may also employ the data set organization and LABEL fields.

Note: The DCB is required to specify tape density for any uncatalogued data set on tape. However, it may be defaulted if the tape density matches that established at system generation.

The DDEF macro instruction or command also has several special uses:

1. To define a job library. Operand fields are as follows:

   ```
   data definition name,VP,DSNAME=data set name,
   DISP=(OLD),OPTION=JOBLIB
   ```

   No other fields are required.

2. To define a data set for dumps. Operand fields are:

   ```
   PCSOUT,VI,DSNAME=data set name
   ```

   Other fields are as needed.

3. To complete the data control block of a data set at execution time. The dcb field is included in this case; other operand fields are as needed for the particular data set.

4. To concatenate data sets (i.e., to define them, for input purposes only, so that several data sets can be read as if they formed a single data set. The OPTION=CONC field is included; other fields are as needed for each data set. The OPTION=CONC field must be given in the DDEF for each data set except the first-defined member of the concatenation. The remaining data sets in the concatenation must each have the same ddname as the first-defined data set.

The DDEF macro instruction or command causes a system entry to be established for the DDEF information so that allocation routines and access methods can refer to it. The link between this information and the problem program's reference to the data set (i.e., the data control block) is the data definition name. The entry containing the DDEF information is maintained until the user logs off or until, through the RELEASE macro instruction or command, the data set is released.

The DDEF macro instruction or command also results in a request, when necessary, for device allocation and volume mounting if the defined data set is private and resides on a demountable volume such as a reel of tape or a disk pack.

Typical Use of DDEF Operand Fields

Case	ddname	dsorg	dsname	dcb	unit	space	volume	label	disp	option
Read a cataloged data set	x		x						x	
Read an uncataloged data set	x	[x]	x		x		x	[x]	x	
Write a data set on a public volume	x	[x]	x			[x]		[x]	[x]	
Write a data set on a private volume	x	[x]	x		x	[x]	x	[x]	[x]	
Modify any data set on a private volume	x	[x]	x		x		x	[x]	x	
Concatenate cataloged data sets while reading private volumes (for each concatenated data set except first in concatenation)	x	[x]	x						x	x
Key: [] indicates operand entry is optional.										

Data Set Organization Requirements

Data Set	Data Set Organization				Comments
	PS	VS	VI	VP	
Any data set on a public volume		x	x	x	
Any data set on a private volume	x	x	x	x	PS applies to direct-access and tape volumes; VS, VI, and VP apply only to volumes on direct-access devices
Any member of a partitioned data set		x	x		The same partitioned data set may include both VS and VI members. (The member must be either VS or VI.)
SYSIN data set		x	x		
<u>Language Processing</u> Source data set for language processing			x		Line data set only. If source data sets are entered from terminal, a line data set is automatically built
Source statements stored as part of SYSIN data set		x	x		A line data set will be built from source statements

Data Set Organization Requirements (continued)

Data Set	Data Set Organization				Comments
	PS	VS	VI	VP	
Object module produced by language processor		x			The object module automatically becomes a member of the most recently defined job library, if any, or of the user's library (SYSULIB).
Job library				x	
Listing data set produced by language processor			x		
Input/Output PCSOUT data set			x		
Input to WRITE TAPE		x	x		
Input to PRINT	x	x	x		
Input to PUNCH		x	x		
Special Command Usage Data set for CALL DATA DEFINITION			x		Line data set only
Data set for LINE?			x		Line or language processor listing data set only
Data set created by DATA		x	x		User option. If VI, must be line data set
Data set created by MODIFY			x		User option determines whether VI is line data set or not

Programming Notes: The DDEF macro instruction or command may be used in conversational and nonconversational tasks.

The user's replies to diagnostic messages issued for his DDEF macro instruction or command should be guided by:

1. If the diagnostic message calls for reentering an element within a given operand field, only that element should be reentered. Preceding and/or following delimiters are unnecessary. Default is acceptable.

2. If the diagnostic message calls for reentering a complex operand field, the whole field should be reentered, including the keyword and equal sign. Default is acceptable.

3. If the diagnostic message calls for reentering an operand field that consists of only one element in addition to the keyword, the reply may be either the element alone or the keyword, equal sign, and element.

4. If the diagnostic message calls attention to an inconsistency and asks the user to enter one of two or three specified operands, the

reply must be a complete operand field. A default is acceptable only if so stated in the message.

The user is informed if the DDEF macro instruction or command cannot be completed. This action can occur for one of these reasons:

1. Invalid punctuation in the operand string.

2. User's volumes cannot be mounted.

3. Sufficient space cannot be allocated.

4. More than three logical inconsistencies were detected in the DDEF macro instruction or command.

Whenever possible, correction and completion of the command will be attempted. But if diagnostic messages indicate that a parameter was misunderstood because of a punctuation error in the operand string, the user should interrupt the operation (by pressing the ATTENTION key) and reenter the corrected command. In confirmation mode, he may prefer to wait for prompting.

The user must never reenter a parameter or part of a parameter that was not requested.

If a keyword is missing or invalid, the pertinent elements following it must be reentered after the corrected keyword and equal sign are typed.

If a parameter occurs twice in the operand string, the second occurrence is preferred. All elements belonging in the earlier occurrence are erased.

DDEF prompting messages are issued according to the operand information already supplied. Unnecessary prompting is kept to a minimum.

If the user's program is being executed in conversational mode and an undefined data definition name is referred to, prompting messages for DDEF operands will be issued to the user, regardless of confirmation mode.

Return Data: At completion of execution of the DDEF macro instruction or command, a code is loaded into the low-order byte of register 15 and register 1 contains the identification of the diagnostic message that explains the error (for nonzero codes).

Code	Significance
00	Successful completion.
04	Undefined data set name (for old data set).
08	Data set name not unique (for new data set).
0C	Attention interruption.
10	Data set organization in DDEF parameter list is not the same as in catalog.
14	Nonexistent generation name.
18	Data set name not fully qualified.
1C	Volume could not be mounted.
20	Space not available.
40	Data definition name not unique.
80	Any error condition not listed above.

APPENDIX I: INTERRUPTION HANDLING FACILITIES

Time Sharing System provides macro instructions that permit the user to control task interruptions (Figure 23).

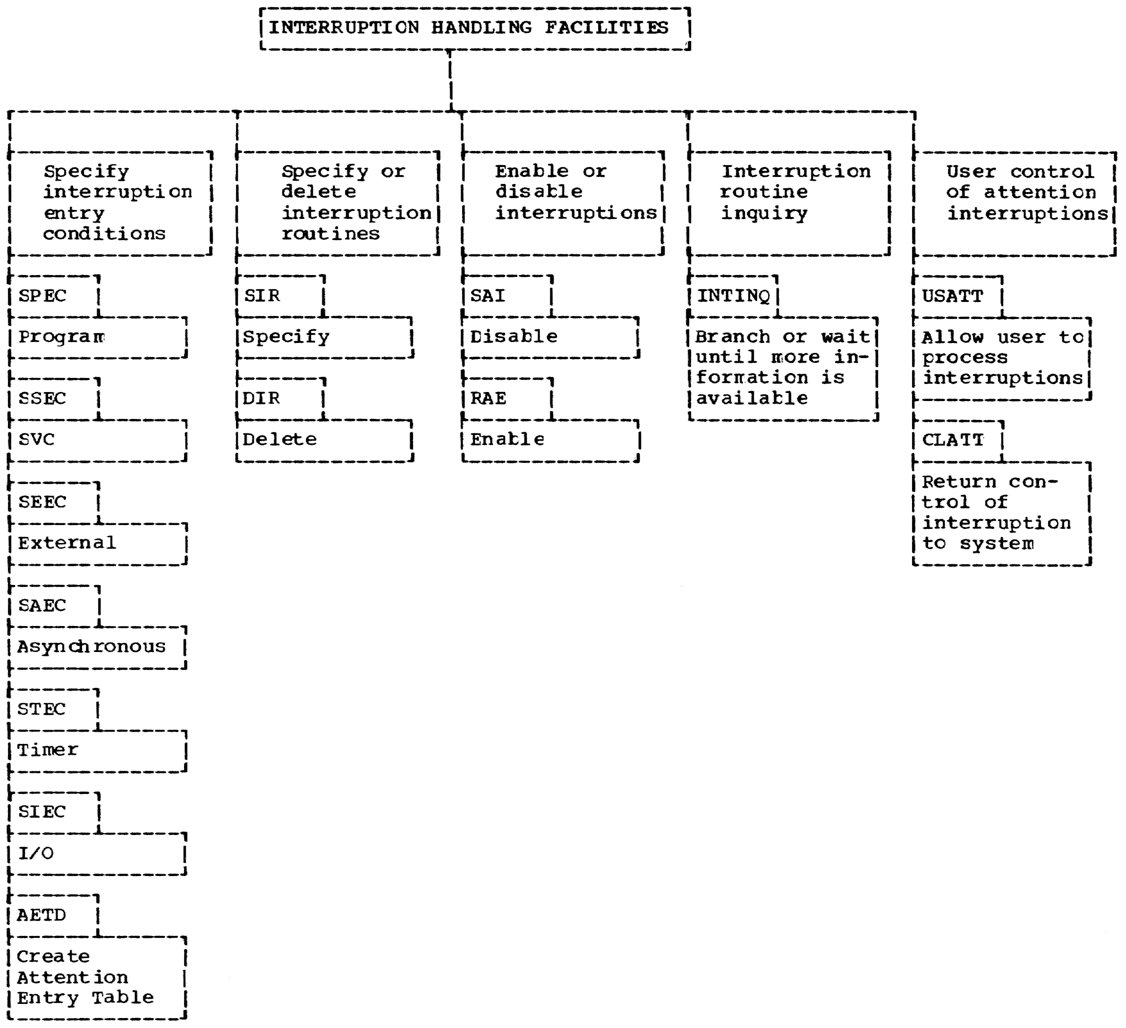

INTERRUPTION HANDLING FACILITIES
Specify interruption entry conditions
SPEC
Program
SSEC
SVC
SEEC
External
SAEC
Asynchronous
STEC
Timer
SIEC
I/O
AETD
Create Attention Entry Table
Specify or delete interruption routines
SIR
Specify
DIR
Delete
Enable or disable interruptions
SAI
Disable
RAE
Enable
Interruption routine inquiry
INTINQ
Branch or wait until more information is available
User control of attention interruptions
USATT
Allow user to process interruptions
CLATT
Return control of interruption to system

Figure 23. TSS interruption handling facilities

INTERRUPTION HANDLING

TSS employs interruptions to facilitate the dispatching of system service routines and interruption error-handling routines. Examples of service routines provided with the system are Page Handling, I/O services, and Main/Auxiliary storage allocation. System program error interruption-handling routines are dispatched in response to SYSIN attention interruptions, task timer interruptions, external, I/O, machine check, program, or SVC-generated interruptions. These interruption-servicing routines attempt to correct any hardware or software error situations that occur during the execution of privileged system programs.

Macro Name	Source of Literal		Adcon literal	Condition Under Which Literal is Generated					
	Operand	Linkage		Operand $>2^{12}-1$	E-form only	Operand specified and not register notation	Program is privileged	Standard form only	Operand not an operand string
SAVE									
SEEC	MSGLTH				X	X			
	INTTYP				X	X			
	EP		X		X	X			
SETL									
SIEC	EP		X		X	X			
SIR		X	X				X		
SPEC	When INTTYP is a sublist				X	X			
	EP		X		X	X			
SSEC	EP		X		X	X			
STEC	INTTYP				X	X			
	EP				X	X			
STIMER									
STOW		X	X				X		
SYSIN									
TRUNC		X	X				X		
TTIMER									
USAGE									
USATT									
VCCW									
WRITE		X	X				X		
	length			X	X				
WT		X	X				X		
	op. string		X			X		X	X
WTL		X	X				X		
WTO		X	X				X		
WTOR		X	X				X		

Figure 22. Literals generated by macro instructions (part 4 of 4)

APPENDIX I: INTERRUPTION HANDLING FACILITIES

Time Sharing System provides macro instructions that permit the user to control task interruptions (Figure 23).

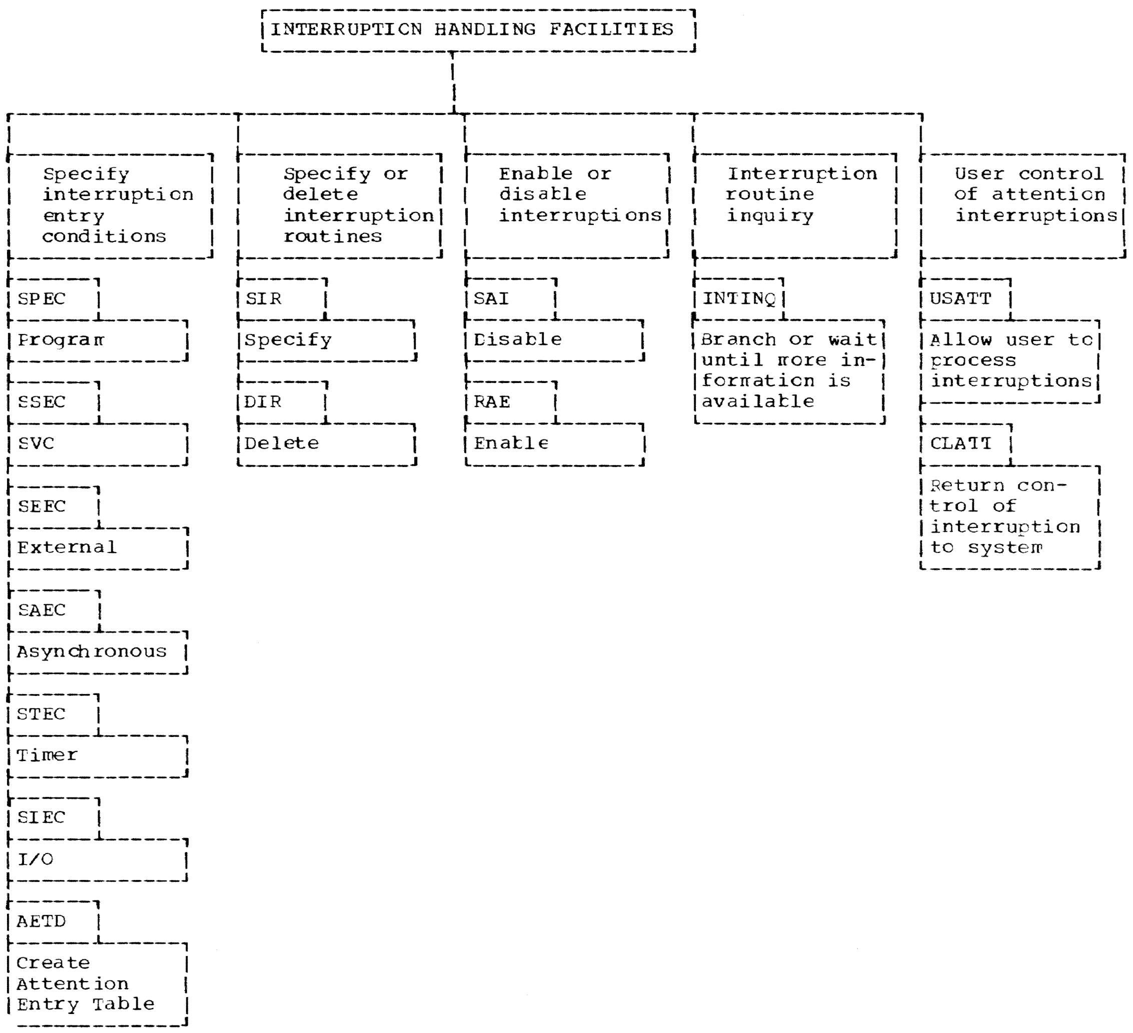

INTERRUPTION HANDLING FACILITIES
Specify interruption entry conditions
Specify or delete interruption routines
Enable or disable interruptions
Interruption routine inquiry
User control of attention interruptions
SPEC
Program
SSEC
SVC
SEEC
External
SAEC
Asynchronous
STEC
Timer
SIEC
I/O
AETD
Create Attention Entry Table
SIR
Specify
DIR
Delete
SAI
Disable
RAE
Enable
INTINQ
Branch or wait until more information is available
USATT
Allow user to process interruptions
CLATT
Return control of interruption to system

Figure 23. TSS interruption handling facilities

INTERRUPTION HANDLING

TSS employs interruptions to facilitate the dispatching of system service routines and interruption error-handling routines. Examples of service routines provided with the system are Page Handling, I/O services, and Main/Auxiliary storage allocation. System program error interruption-handling routines are dispatched in response to SYSIN attention interruptions, task timer interruptions, external, I/O, machine check, program, or SVC-generated interruptions. These interruption-servicing routines attempt to correct any hardware or software error situations that occur during the execution of privileged system programs.

Normally, except for attention interruptions at the SYSIN terminal, interruptions occurring during non-privileged problem programs are not serviced by the system in the same manner. In such cases, an interruption causes an appropriate error message to be written from the system message file onto the task's SYSOUT device. If the task is conversational, the user is prompted for corrective action. If the task is non-conversational, the task is abnormally terminated.

User Interruption Facilities

To give the user more flexibility in the handling of interruptions occurring during the execution of his problem programs, TSS provides a user with facilities for supplying his own interruption-servicing routines. Examples of service routines that may be supplied by the user include: program interruption-handling routines, routines for creating and handling unused SVC codes, routines for handling task timer interruptions, and for handling special task I/O interruptions. The basic interruption-handling logic used by the system and the interruption-handling macro facilities provided to a user by TSS are summarized below.

BASIC INTERRUPTION SERVICING LOGIC

A system interrupt table is used to establish a queueing mechanism for any interruptions occurring on the various devices in the system. The Interrupt Table (see Figure 4) holds information concerning device types, interruption-servicing routines, and interruptions to be processed. There are three main types of entries in the table: Device Entries (DE), Request Entries (RE), and Queue Entries (QE). A device entry exists for each source that can signal interruptions to the system. Thus, they exist for each physical I/O device attached to the system as well as for the logical sources causing program interruptions, SVC, external, and timer interruptions.

Request Entries are attached to the device entries. There must be an RE for every service routine which may be dispatched to handle interruptions. A QE generally represents a particular interruption, and all interruptions that are to be serviced by the same routine have their Queue Entries chained to the RE that is associated with the routine that is to service that type of interrupt.

The Device Entry is constructed by one of two methods; the SIR macro instruction builds it into the interrupt table or it is predefined at SYSGEN. A set of Request Entries is attached to appropriate device entries in the interrupt table at system assembly or dynamically during task initiation to provide for the dispatching of system service routines and interruption error-handling routines supplied by the system. Queue Entries are entered in the Interrupt Table, by the system, and attached to a particular Request Entry, when an interruption occurs.

When an interruption routine is dispatched, linkage between the system queueing mechanism and the various system or user-provided interruption servicing routines is generally established by two other areas: the Interrupt Control Block and a Communication Area. A brief description of the contents of Device Entries, Request Entries, Queue Entries, interrupt control blocks and the Communication Area follows.

Device Entries - the device entry contains a device type code, a highest priority, Request Entry Code, a highest priority active request entry code. Predefined device entries exist for four of the six interruption types (program, SVC, external, and timer). The asynchronous and synchronous I/O interruption device entries are built by SIR for each device allocable in the system.

Request Entries - the request entry contains an activity indicator, an interruption servicing priority code, a pointer to a description of the service routine (contained in the ICB), and a pointer to its queue entries (if any exist).

Queue Entries - contain the necessary interruption information from the VPSW and the sense and status information from the ISA required by the system at dispatch time. Some of the information in the QE is moved to a user defined Communication Area (com) at dispatch time so he may analyze the conditions and status at the time of the interrupt. The QE represents the occurrence of an interruption of the type specified in the RE to which the QE is attached (chained).

Interrupt Control Blocks - specifies what type of interruptions are to be processed, by a particular interruption-servicing routine, under what conditions the servicing routine is to be entered, and the entry point address of the interruption routine. It also points to a communication area in the user problem program in which information identifying the type and status of an interruption is placed by the system when an interruption occurs.

Communication Area - in addition to its primary purpose of holding interruption information for analysis by an interruption servicing routine, it allows information to be passed between an interruption routine and the interrupted program (in which the communication area must reside). Thus, a field in the communication area may be used as an event control block where completion of interruption processing can be posted. The system places QE information in the communication area when an INTINQ macro instruction with the CLEAR mode is specified, and when a QE is dispatched to its interruption servicing routine.

The relation of these areas is shown in Figure 24.

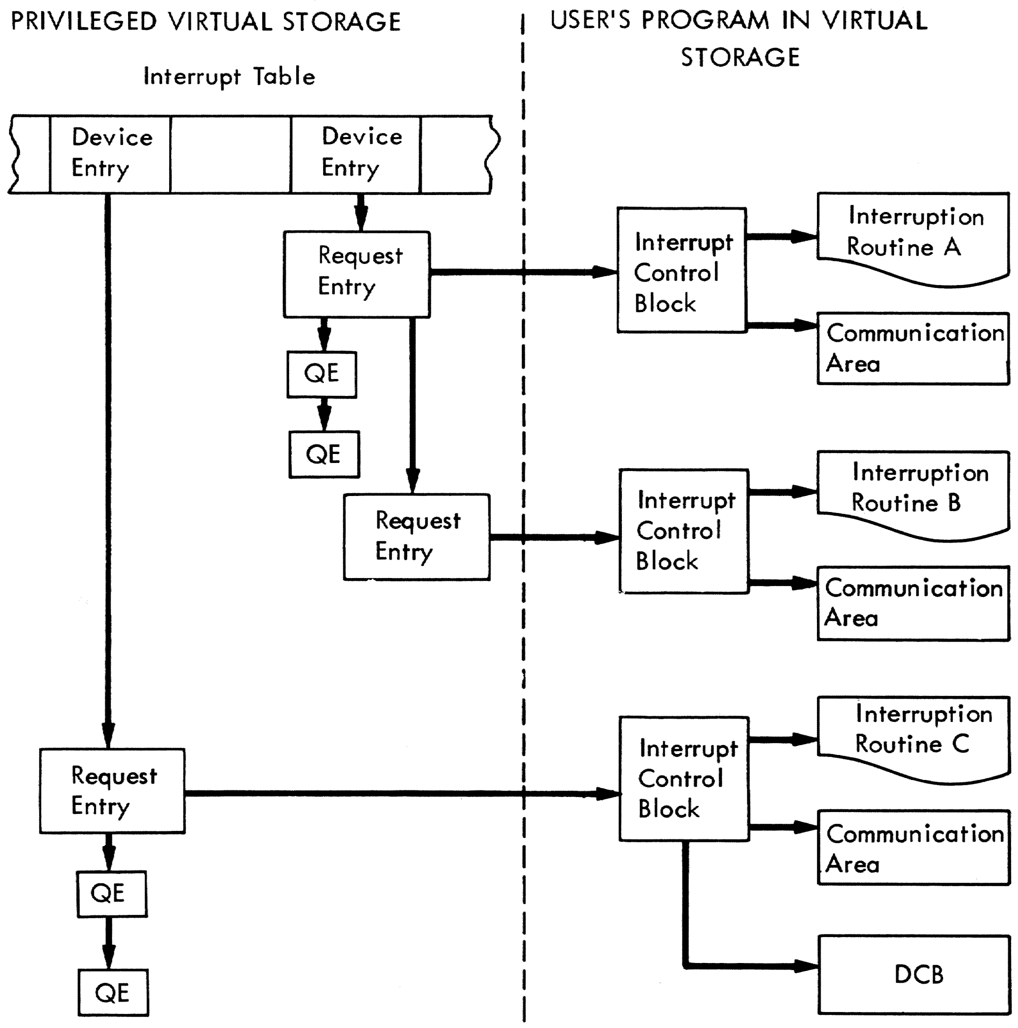
User Establishment of Interruption Handling Routines

When a user desires to service particular interruptions occurring in a problem program, he must create the entries and control areas required by the system for servicing that type of interruption. Thus, as shown in "Basic Interruption Handling Logic," the user must create a Request Entry, Interrupt Control Block, and Communication Area. Device entries not entered at SYSGEN are entered via user issuance of a SIR macro instruction. A Queue Entry will be created by the system when an interruption occurs and does not have to be created by the user.

The TSS macro facilities provide a user with an easy method of accomplishing this. During problem program execution a user can create his own servicing routines for six kinds of interruptions. These six interruption types and the macro instructions used to create the ICB for each type is indicated below.

Interruption Type	Macro Instruction for Creating ICB
Program interruption handling routines	SPEC
Routines for handling SVC codes not recognized by the system	SSEC
I/O interruption handling routines	SIEC
Task timer interruption handling routines	STEC
Asynchronous interruptions	SAEC
External interruptions	SEEC

The Communication area must be coded by the user in the format illustrated in the descriptions of the above macro instructions.



Notes: Request Entries are created by SIR Macro Instruction issued in User's Program

Interrupt Control Blocks are created by

SIEC	SEEC
STEC	SSEC
SAEC	SPEC

Macro Instructions

Figure 24. Interruption handling logic

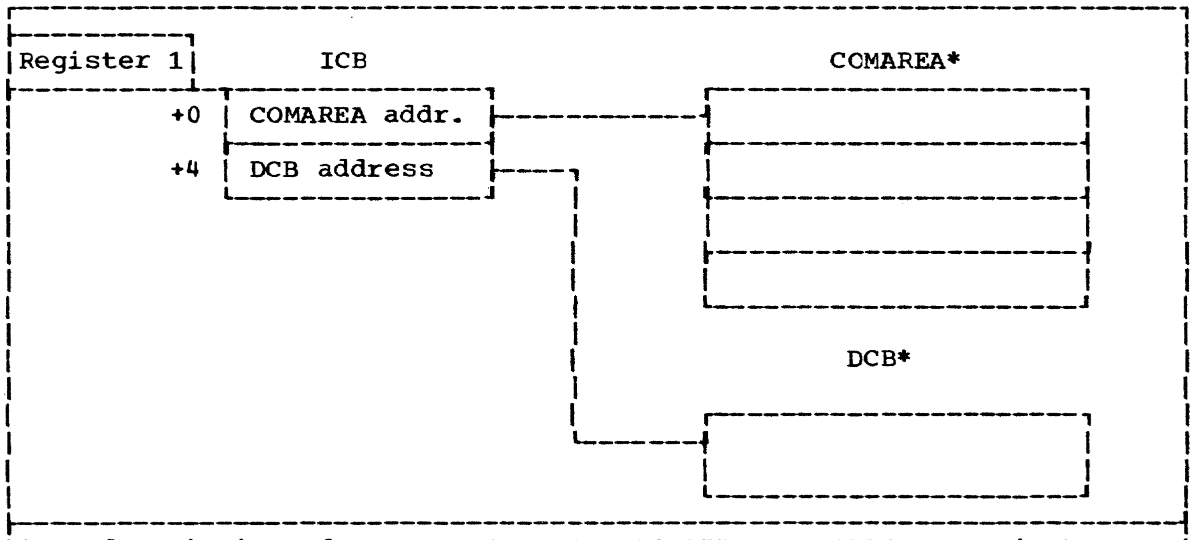
Once an ICB has been created by a problem program and a communication area is established, a user can issue the SIR macro instruction to attach a Device Entry (if one does not already exist) and a Request Entry (RE) to the interrupt table, and establish a priority for the handling of the interruption type.

The user should avoid issuing a FREEMAIN macro instruction on an ICB that has a currently active interruption routine.

After the user has established the RE, ICB, and communication area for a particular type of interruption, and that type of interruption occurs, the interrupt table is searched, the user-created RE is located, and the system attaches a QE to the RE. (Note the exception to this under "SYSIN Attention Interruption Handling.") The System's task interruption queueing mechanism then causes the subsequent dispatching to appropriate user-coded interruption handling routines by priority.

Writing Interruption Servicing Routines

When an interruption occurs in a nonprivileged routine, an asynchronous exit is taken from the interrupted routine and control is passed to the entry point of the user's interruption routine (which must be aligned on a fullword boundary). Information identifying the type of interruption that occurs is made available in a communication area in the interrupted program. When the interruption routine is entered, register 1 contains the address of a two-word parameter list. The first word of the parameter list contains the address of a communication area, and the second word contains the address of a data control block (Figure 25).



*See description of SPEC,SAEC,SIEC,SSEC,SEEC, or STEC macro instructions for format.

Figure 25. Information available upon entry to an interruption routine

Using this information, the interruption routine can perform any calculations necessary, issue input/output macro instructions, and do what is necessary to respond to the interruptions.

An interruption servicing routine might want to take one of several actions, depending on the occurrence and status of other interruption events. The user has been given the capability of inquiring as to the status of other user-specified interruption routines with respect to the existence of outstanding interruption events. The information pertaining to a particular interruption and held in its Queue Entry may be inspected by the interruption servicing routine through use of the Interrupt Inquiry (INTINQ) macro instruction in an interruption processing routine. INTINQ can be used to determine whether specified interruption events have occurred. It uses a user-specified ICB to determine if any QEs exist that are associated with the interruption-servicing routine described by that ICB. If a QE exists it indicates that an interruption of that type has occurred. The subsequent action of a user interruption-servicing routine in which INTINQ was issued is determined by the occurrence or nonoccurrence of the specified interruption and the mode set by the INTINQ macro instruction. These modes and the actions taken by the interruption-servicing routine are summarized below.

Mode	Required Interrupt QE Information		Action
	Available	Not Available	
R	X		Continue execution with the next sequential instruction in the interruption servicing routine
		X	Relinquish control, thereby passing control to one of the user's lower-priority interruption servicing routines, or return control the user's task at the point of interruption. The occurrence of the expected interruption returns control to the interruption routine and resumes execution at the instruction following INTINQ.
W	X		Continue execution with the next sequential instruction in the interruption servicing routine.
		X	Enter the wait state. When and if the expected interruption occurs, control is returned to the interruption servicing routine and execution resumes at the next sequential instruction following INTINQ. Any higher priority interruptions occurring while this routine is in the wait state will be processed.
C	X		Branch to specified branch address.
		X	Continue execution with next sequential instruction.
CLEAR	X		Moves the information from the QE to the communication area, deletes queued interruptions, and processing continues with next sequential instruction following INTINQ.
		X	Processing continues with next sequential instruction.

Disabling Interruptions During Execution of Interruption Servicing Routines

In order to ensure that an interruption servicing routine can or cannot be interrupted by subsequent interruptions, two macro instructions, SAI and RAE, are provided to the user by TSS. Issuance of the SAI macro instruction in an interruption-servicing routine inhibits subsequent interruptions from taking place while the interruption-handling routine is being executed. No interruptions will be lost, however, because they are queued up. If interruptions are not disabled by SAI, interruptions of higher priority can interrupt a lower priority interruption-servicing routine. If a user desires, he may issue a RAE macro instruction that will allow all subsequent interruptions to interrupt his servicing routine regardless of their priorities.

SYSIN Attention Interruption Servicing

Normally, user interruption-servicing routines do not replace a system interruption-servicing routine; they merely service interruptions not serviced by the system. However, a system routine does exist for the handling of the SYSIN device attention interruptions. Thus, a user must disable the system's servicing routine in order to substitute his own servicing routine. He does this by creating the Request Entry and Interrupt Control Block with the SIR and SAEC macro instructions respectively (use of the SAEC macro instruction is restricted to privileged programmers when the SYSIN device is specified). These macro instructions, provide the user with interruption-servicing routine for attention interruptions occurring in his problem program. At this point however, two attention interruption-servicing routines have Request Entries in the Interrupt Table, the system Attention Handler and the user-created servicing routine. Since the system Attention Handler is a privileged routine; it has a higher priority RE than the user-created routine. Thus, unless the Attention Handler RE is deactivated it would continue to receive attention interruptions.

A user can use the User Attention (USATT) macro instruction to deactivate the system-provided SYSIN attention-handling routine thereby leaving his lower priority servicing routine in control of handling attention interruptions on SYSIN. When the user wants processing of attention interruptions to be resumed by the system, he issues a Clear Attention (CLATT) macro instruction which enables the RE associated with the system Attention Handler.

A second way of establishing user SYSIN attention interruption-handling routines for terminals is provided by the AETD macro instruction. AETD generates a table containing addresses of routines that are to be given control when the user presses the attention key a specified number of times. Thus a user may call one or five different attention interruption servicing routines depending on whether he presses the attention key once, twice, or five consecutive times. When using AETD the user does not have to set up a Request Entry or Interrupt Control Block as when using SIR and SAEC. AETD routines make use of the Request Entry for the systems Attention Handler routine. The user routines made available by AETD are in fact made a part of the system's Attention Handler routine and remain a part of it until an AETD macro instruction with no operand is issued.

Multi-Level AETD Interruption Routines

AETD may be specified in a user program that is invoked to handle an attention interruption in another program, as defined by an AETD in that program, without causing the first AETD to be ignored. AETD macro instructions can be issued at up to ten such levels. However, if more than one AETD is issued in the same program, only the last is recognized. When a program that has issued an AETD exits, the AET entry specified by that program is deleted.

On an attention interruption, if there is no AETD for that level of program, the user is prompted for input. If an AET entry has been specified, the specified routine is given control and the routine exits to the interrupted program (the program containing the AETD that specified the interruption routine).

If the attention key is pressed several times before the command system can process the first interruption, and if no AET entry is active, all but the last attention is ignored. If an AET entry is active, each attention routine up to the number corresponding to the number of times ATTN was pressed is given control and, except for the last such routine, is immediately interrupted by the next queued interruption. As each routine exits, the next lower routine is given control until it in turn exits; finally, the user program that was first interrupted resumes control.

An AETD issued at any level with no operand disconnects from the system any AET entry specified in the same program.

AETD Versus SIR, SAFC, and USATT

If SIR, SAFC, and USATT are employed to establish a SYSIN attention interruption servicing routine, a user can establish different priorities for the handling of attention interruptions in relation to other types of interruptions. If AETD is used, the user has no control over this type of priority specification.

Another major difference is the recovery abilities for errors occurring during the execution of a user-specified attention interruption servicing routine. If the routine was established via SIR, SAFC, and USATT, there is no recovery ability and disastrous results may occur. When the routine is established with AETD, a user can press the attention key a number of consecutive times and control will be passed to the command system. The user can then proceed to attempt error recovery using the command system.

Handling Attention Interrupts

With the inclusion of buffered output support for SYSOUT, a change was made to the way attention interrupts are handled by TAMII. When a user presses the attention key, TAMII sets a software interlock on the terminal to prevent any more requests from being processed (for the user's terminal) until the attention interrupt has been processed.

TSS contains a default (ATTNMODE) which controls whose responsibility it is to reset the attention interlock and dispose of any pending I/O requests from the task. ATTNMODE is tested by the task's attention interrupt handler to determine if the interlock is to be reset by the system or the attention handler. If ATTNMODE=OLD the attention interrupt dispatcher in CZCJT will issue the TAMII macros to reset the interlock and purge any pending I/O requests. If ATTNMODE=NEW the attention dispatcher assumes that the SIRed attention routine will take the appropriate action to reset the interlock and to handle any pending I/O requests.

For handling attention interrupt control, the following TAMII macros are available for the user:

(1) To reset the software attention interlock and to restart any pending I/O requests, the ICNTRL with TYPE=RESTART macro should be used.

(2) If the application programmer wishes to purge the pending I/O before resetting the attention interlock, the programmer would issue the TCLEAR with TYPE=ALL macro. This would cause all pending I/O requests to be purged and any associated DECBs to be marked purged.

For further examples on attention handling see Appendix N.

A symbolic library is composed of a symbolic component and index component. The symbolic component may contain any collection of named groups of symbolic lines called regions; thus, a collection of macro definitions corresponding to the TSS system macro instructions, together with any regions to be accessed by means of the COPY assembler instruction, form the symbolic component of the TSS system macro and COPY library. This library provides the TSS assembler with the macro definitions and COPY regions it needs, when system macro instructions or COPY statements are encountered.

In this library, each macro definition is a group of symbolic lines whose name (region name) is the same as that of the operation of the definition's prototype and the corresponding macro instruction. Each COPY region is a group of symbolic lines to whose name a COPY statement must refer, to copy the region into a program. The symbolic component of the system macro and COPY library is normally cataloged as a virtual index sequential data set. The organization and format of this component is shown in Figure 26. The format of each symbolic line, shown in Figure 27, is that of a record in a region data set. The lines of information within the symbolic component are ordered by line number. The number of the first line of each region is used to index the symbolic component.

The index component is a table that relates the name of each region to the number of its first line. Thus, any region in the system macro and COPY library may be located within the symbolic component by matching the operation, of the corresponding macro instruction or operand of the corresponding COPY statement, to the appropriate entry in the index. The index component is normally cataloged as a virtual sequential data set. It consists of a single format-U record.

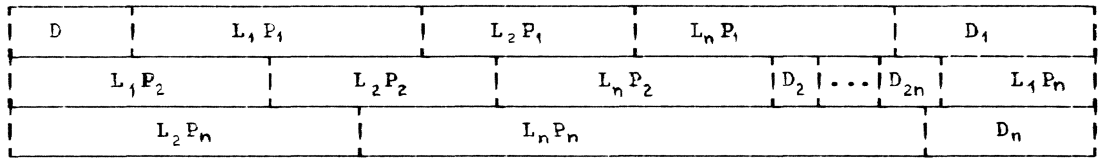


Figure 26. System macro and COPY library symbolic component format

D
: is a 21-byte line whose first character, always a right parenthesis, marks it as the delimiter line for a region. The 8-character field following the right parenthesis contains the name of the following region, left-adjusted with trailing blanks.

D_2, ... D_{2n}
: are synonyms for the following region. Any region may have synonyms (aliases).

L_jP_k
: is the jth line of the kth region; its length is four more than the number of bytes given in its length field.

Note: The first line of a region is L_1P, not D_1.

RESUME				
LL	RN	LN	C	T
4 Bytes	8 Bytes	7 Bytes	1 Byte	(LL-16) Bytes

Figure 27. Format of a line in a region data set

LL
: is the length of the line excluding the four-byte LL field.

C
: is a code whose values and their meanings are:

<u>Code</u>	<u>Meaning</u>
01	The line originated at a terminal keyboard
00	The line was obtained as a card image

<u>Note:</u> C is normally 00 for all lines of the system macro and COPY library.

RN
: is the name of the region; must be the same name as the macro or copy contained within the region.

LN
: is the line number.

T
: is the text of the symbolic line; its length is sixteen characters less than the value specified by LL.

SYSTEM MACRO AND COPY LIBRARY SERVICE FACILITIES

<u>Generating the Library</u>

The library may be created and modified by using any of the several TSS commands that create line or region data sets.

Changes are made as a function of line number. Each line in a line data set contains a line number; lines in the data set are ordered by line number. Once the line data set is created, the user may execute SYSINDEX. Alternatively, a user's program may perform the required function by calling SYSXBLD. These routines create the index (CHASLX) which relates the name of each region to its first line. When the Editor is used to change the line number of the first line of any region in the symbolic component, an updated index must be created. The use of Editor does not otherwise require the subsequent use of SYSINDEX or SYSXBLD.

<u>Using Symbolic Libraries</u>

The TSS assembler uses the symbolic library search routine (SYSEARCH) to locate a region in the system macro and COPY library when it encounters a system macro instruction or an assembler COPY statement. SYSEARCH inspects the index that the assembler has presented to it, and returns with a return code of 4 if the required region is not in the library. If the required region <u>is</u> in the library, SYSEARCH returns with the number of the first line of the region and a return code of 0.

The assembler uses the line number obtained from SYSEARCH, in conjunction with a SETL macro instruction, to position the symbolic component at the required region. Successive statements are then obtained by using the VAM GET facility.

SYSEARCH is called to determine whether the region is present and, if so, positions the symbolic component to the designated region. If the region is not present, exit is made with a return code of 4; otherwise, it exits with a return code of 0. In the latter case, SYSEARCH is repeatedly called to obtain successive lines of the region.

As each line is obtained, SYSEARCH determines whether the line is still in the required region by testing the first text character or region name. If that character is a right parenthesis if the region name is different, or if the EODAD sequence receives control, exit is made with a return code of 4; otherwise the line is presented to the assembler and exit is made with a return code of 0. When the assembler is retrieving a macro definition from the library, it will normally sense the end of the definition when it receives the definition's MEND statement.

If, instead, it detects a return code of 4 before it receives the MEND statement, it assumes that a library format-error exists. When the assembler is retrieving a COPY region, it relies upon a return code of 4 from SYSEARCH to detect the end of the region.

Requesting Symbolic Library Services

The symbolic library indexing routine (SYSINDEX) is a system utility routine that processes the user's input parameters.

The user defines his data sets thus:

```
ddef source,vi,libname
ddef index,vs,ndxname
```

where: libname specifies the name the user wishes to assign to the macro data set and ndxname specifies the name the user wishes to assign to the index data set.

The user then issues SYSINDEX and the system prompts him to enter the input parameters. The sequence is:

```
sysindex
SUBMIT CONTROL STATEMENT
header=),length=8
```

The control statement is requested only if the library is not a region data set. The system then prompts the user for the next command.

Note: SYSINDEX does not accept parameters when called. CGCKA will receive its parameter via prompting. Users of the SYSINDEX function should observe the above sequence.

The header character is the single character that is compared with the first byte of each source line to determine whether that line has an index entry. This parameter may be omitted if the user specifies:

```
scan=subroutine
```

where subroutine specifies the name of a subroutine that is supplied by the user; it is called to inspect each successive line of the symbolic component. This routine determines whether a given line has an entry in CHASLX. The SCAN option is not used if the header parameter is supplied.

The build symbolic library index routine (SYSXBLD) constructs the index portion (CHASLX) of the symbolic library. It is invoked by means of a CALL macro instruction of the following format:

Name	Operation	Operand
[symbol]	CALL	SYSXBLD,(length,[header][,scan])

length
specifies the location of the length of region names in the library.

Specified as: Register notation (2 through 12), or a relocatable expression.

header
specifies the location of a character used in determining what lines of the symbolic component require index entries. The header character is compared with the first character of each line to make this determination. If header is given, it must be the second element of the sublist and scan must not be given.

Specified as: Register notation (2 through 12), or a relocatable expression.

scan
specifies the location of an eight-character name of a user's scan routine. The name must be left-adjusted and filled with trailing blanks if necessary. The user's scan routine is called as each symbolic line is obtained to determine whether the line requires an index entry. If scan is given, it must be the third element of the sublist and header must not be given.

Specified as: Register notation (2 through 12), or a relocatable expression.

The symbolic library search routine (SYSEARCH), used to locate information stored in a symbolic library, is invoked by means of a CALL macro instruction of the following format:

Name	Operation	Operand
[symbol]	CALL	SYSEARCH,(index,name,line number)

index
is the address of the index component (CHASLX) of the symbolic library to be searched. CHASLX must be brought into storage by the user.

Specified as: Register notation (2 through 12), or a relocatable expression.

name
is the address of the first byte of the name to be located. This name must be of the length specified to SYSINDEX or SYSXBLD during the creation of the index, and must be left-adjusted with trailing blanks.

Specified as: Register notation (2 through 12), or a relocatable expression.

line number
is the location at which the SYSFARCH routine is to store the retrieval line number it obtains.

Specified as: Register notation (2 through 12), or a relocatable expression.

Return Data: On exit, a hexadecimal code will be returned to the calling program in the return code register.

Code	Meaning
00	The name was located. The retrieval line number will be placed in the location designated by the third parameter.
04	The name could not be located.

To be concurrently accessible to more than one task, a data set must have one of the following organizations:

- Virtual sequential
- Virtual index sequential
- Virtual partitioned

Physical sequential data sets cannot be used concurrently by more than one task.

To prevent several users from concurrently updating the same record of a virtual storage data set, interlocks are put on the data set while it is being used. The interlocks, read and write can be imposed at three levels: page, data set, or member.

Types of Interlocks

A read interlock is imposed to prevent other users from writing into a data set, member, or page of a data set. Multiple read interlocks may be established for a data set or member, permitting several users to read it simultaneously; or the interlocks may be set on a page basis, giving several users simultaneous access to the records within a page. A read interlock cannot be set if a write interlock has already been set for the data set, member, or page.

A write interlock prevents any user, other than the user who set the interlock, from reading or writing into a data set, member, or page. Only one write interlock can be set at a time; thus, once a write interlock is set, neither read nor write interlocks can be applied until the write interlock is reset.

Levels of Interlocks

- Data set interlock - set according to the OPEN option specified, as shown in Figure 28. This level of interlock restricts the use of subsequent OPEN macro instructions on shared data sets. The interlock is reset when the data set is closed.

OPEN option	VSAM data set	VISAM data set	VPAM data set (member level)
INPUT	read interlock set	read interlock set	read interlock when FIND issued
OUTPUT	write interlock set	write interlock set	write interlock set when FIND issued
INOUT OUTIN UPDAT	write interlock set	read interlock set	when FIND is issued: write interlock is set on VSAM members; read interlock is set on VISAM members

Figure 28. Effect of OPEN options on data set interlocks

- Member interlock - set when the FIND macro instruction is issued for a member of a virtual partitioned data set. A member interlock is reset when a STOW type-R or CLCSE or FIND macro instruction is issued.

- Page interlock - set to ensure that the user has exclusive control of a record while he is processing it. A page-level interlock is reset when a reference is made to another page in the data set or when the data set is closed.

User Considerations

The only way a user can gain exclusive control of a shared VISAM data set is to open it for OUTPUT. Although a data set is opened for CUTPUT, a user may actually only want to read the data set.

When updating a VISAM data set, the record to be updated should have been obtained by a READ (type KX). If users of a shared data set do not employ this procedure, two tasks may concurrently refer to the same page using either the GET or READ (type KY) macro instructions and decide that a record within the page is to be updated. Since both tasks WRITE to the same page, the task that issues the last WRITE macro instruction cancels the effects of the previously issued WRITE. The following sequence prevents this situation:

```
GET  (1)
.                                   decision that updating of the record
.                                   is required
.
READ DECB, KX, (1), (0), (2)
.
.                                   update record
.
WRITE DECB, KS, (1), (0), (2)
```

A READ (type KZ) by retrieval address should not be employed by users of VISAM shared data sets since the retrieval address of the desired record can be shared by another task.

Coding sequences within a task may produce task looping that cannot be detected by the access method. Consider, for example, this sequence:

```
READ DECB, KX, (1), (0), (2)
GET (1)
```

where the READ and GET macro instructions refer to different DCBs within the same task. This situation produces a task loop, since the GET macro instruction waits for the write interlock, set by the previous READ macro instruction, to be reset. The write interlock will not be reset since it was set in the same task that is waiting fcr the write interlock to be reset. The user must pay close attention to the rules of interlock setting and resetting when dealing with multiple opened DCBs within a given task.

One doubleword parameter list is generated for each data set DCB being opened or closed and placed in a table, as described below:

byte	contents
0-3	Address of the DCB
4	OPEN/CLOSE option code
5-7	(00 00 00) 16

The bit configurations for the option codes are indicated below.

bits 0-7	option
00XXXXXX	another DCB is to be opened or closed
10XXXXXX	this is the last DCB to be opened or closed
XX01XXXX	REREAD
XX11XXXX	LEAVE
XXXX0000	INPUT
XXXX1111	OUTPUT
XXXX0011	INOUT
XXXX0111	OUTIN
XXXX0001	RDBACK
XXXX0100	UPDAT

APPENDIX M: CONDITIONAL ASSEMBLY OF MACRO INSTRUCTIONS

Of the macro instructions documented in this publication, some may be assembled only with non-privileged code, while others may be assembled with either privileged or non-privileged code. The differences in the expansions of these two classes of macro instructions lie primarily in the types of linkage they develop; those to be assembled with non-privileged code assemble with one kind of linkage, those to be assembled with privileged code assemble with a different kind of linkage. Other macro instructions have no differences in their assemblies.

Determination of the type of code to be assembled for a macro instruction is made by examination of a global symbol that is set by the DCLASS macro instruction (which acts as a conditional assembly instruction and executes during the assembly). Programmers may issue a DCLASS USER macro instruction in a privileged module so that they may assemble non-privileged code; the global symbol can be reset later by means of a DCLASS PRIVILEGED macro instruction. Similarly, programmers may manipulate the global symbol to permit assembly of privileged code in a non-privileged module. Note that the global symbol in no way affects the ability to execute the macro thus assembled. However, in many cases, code that is generated on the basis of the DCLASS USER option may not be executable in a privileged module; similarly, code that is generated under the DCLASS PRIVILEGED option is rarely executable in non-privileged modules.

Figure 29 lists the macro instructions that should be assembled with the DCLASS setting appropriate to the type of code to be assembled. These macro instructions generate appropriate linkage so that the code can be executed correctly, dependent on the setting of the global symbol. (If O- or P-authority programmers wish to assemble one of the macro instructions listed in Figure 29 to be executed in nonprivileged code, they should assemble that macro instruction under the DCLASS USER option.) Figure 30 lists the macro instructions that must be assembled under the DCLASS USER option (either explicitly or by default). The macro instructions listed in Figure 31 are assembled without regard for the setting of the global symbol; no DCLASS macro instruction need be issued to set the global symbol.

ABEND		FREEPOOL	RELSE
AETD		GATRD	SAI
ATTNDST		GATWR	SETDV
ATTNRST		GDV	SIR
ATTNSAV		GETBUF	SOLICIT
BSP		GETDV	STEC
CAT		GETMAIN	STOW
CDD		GETPOOL	SYSIN
CHCKT		GTWAR	TCLEAR
CHECK		GTWRC	TCNTRL
CLOSE		GTWSR	TDCMD
CNTRL		INTINQ	TFREE
COPYDS		IOREQ	TGATRD
CSTORE		LIBESRCH	TGATWR
DDEF		LPCEDIT	TGATWS
DEL		LPCINIT	TGTWAR
DELREC	DEQ	MCAST	TGTWSR
DIAL		NOTE	TRCBUF
DIR		OBEY	TREAD
DQDECB	ENQ	OPEN	TWRITE
ERASE		PIREC	TWRTLST
ESETL		POINT	USAGE
FEOV		PR	WRITE
FIND		PRMPT	WT
FINDDS		PU	WTL
FINDJFCB		RAE	WTO
FREEBUF		READ	WTOA
FREEMAIN		REL	WTOR
		RELEX	

Figure 29. Macro instructions having conditional DCLASS assemblies

CLATT	COMMAND	STIMER
CLIC	EXIT	TTIMER
CLIP	PAUSE	USATT

Figure 30. Macro instructions requiring DCLASS USER

ADCON	DCBD		RETURN
ADCOND	DELETE		RSVSEG
ARM	DELSEG		SAEC
AWAIT	DISCSEG		SAVE
BPKDS	EBCDTIME	EXCSEG	SEEC
CALL	GET	GETSEG	SETL
CHDERMAC	HASH		SIEC
CHDPSECT	LOAD		SPEC
CHDVAL	MARKRTRN		SSEC
CKCLS	PUT	PUTSEG	TRUNC
CONSEG	PUTX		VCCW
DCB	REDTIM		VSEND
	RELSEG		XTRTM

Figure 31. Macro instructions not requiring DCLASS

APPENDIX N: TELECOMMUNICATIONS ACCESS METHOD (TAMII)

When using TAMII macros, the user must be familiar with the effects of three TAMII implicit operands which affect the execution of TAMII macros. These implicit operands are as follows:

OUTMODE={W|<u>B</u>}
INMODE={<u>W</u>|B|S}
CMDMODE={<u>W</u>|B}

The default for each of these implicit operands is underlined in the expressions above, but may be changed by the DEFAULT command.

OUTMODE controls the execution of write requests. (See Figure 32.) If OUTMODE=W, TAMII waits for the completion of the request before returning to the application program. If the request has a DECB associated with it, TAMII does not return until the DECB has been posted. If OUTMODE=B, TAMII returns to the application program as soon as the request has been scheduled for execution. If a DECB is used, the application program must perform a CHCKT to verify the completion of write requests.

```
---------
| ENTRY |
---------
    |
    |
    |                                   ------------       ------------
------------                            | USER     |       | SYSTEM   |
| OUTPUT   | YES     ------------ W     | PROVIDED | NO    | ASSIGNS  |
| REQUEST? |---------| OUTMODE= |-------| DECB?    |-------| DECB     |
------------         ------------       ------------       ------------
                          |                  |                   |
                          | B                | YES               |
                          |                  |                   |
                          |                  |             ------------
                          ---------------------------------| ISSUE    |
                                                           | REQUEST  |
                                                           | TO RTAM  |
                                                           ------------
                                                                 |
                                 ------------                    |
    ----------------             | USER     |                    |
    | ISSUE TWAIT  |         YES | PROVIDED |            W -------------
    | ON USER DECB |-------------| DECB?    |-------------| OUTMODE=  |
    ----------------             ------------             -------------
           |                           |                         |
           |                           | NO                    B |
           |                           |                         |
           |             ------------------------------          |
           |             | ISSUE CHCKT ON SYSTEM DECB |          |
           |             ------------------------------          |
           |                           |                         |
       ----------                      |                         |
       | RETURN |-------------------------------------------------
       ----------
```

Figure 32. TAMII output request flow diagram (simplified)

Note: Any request that results in data being returned to the application program is considered an input request.

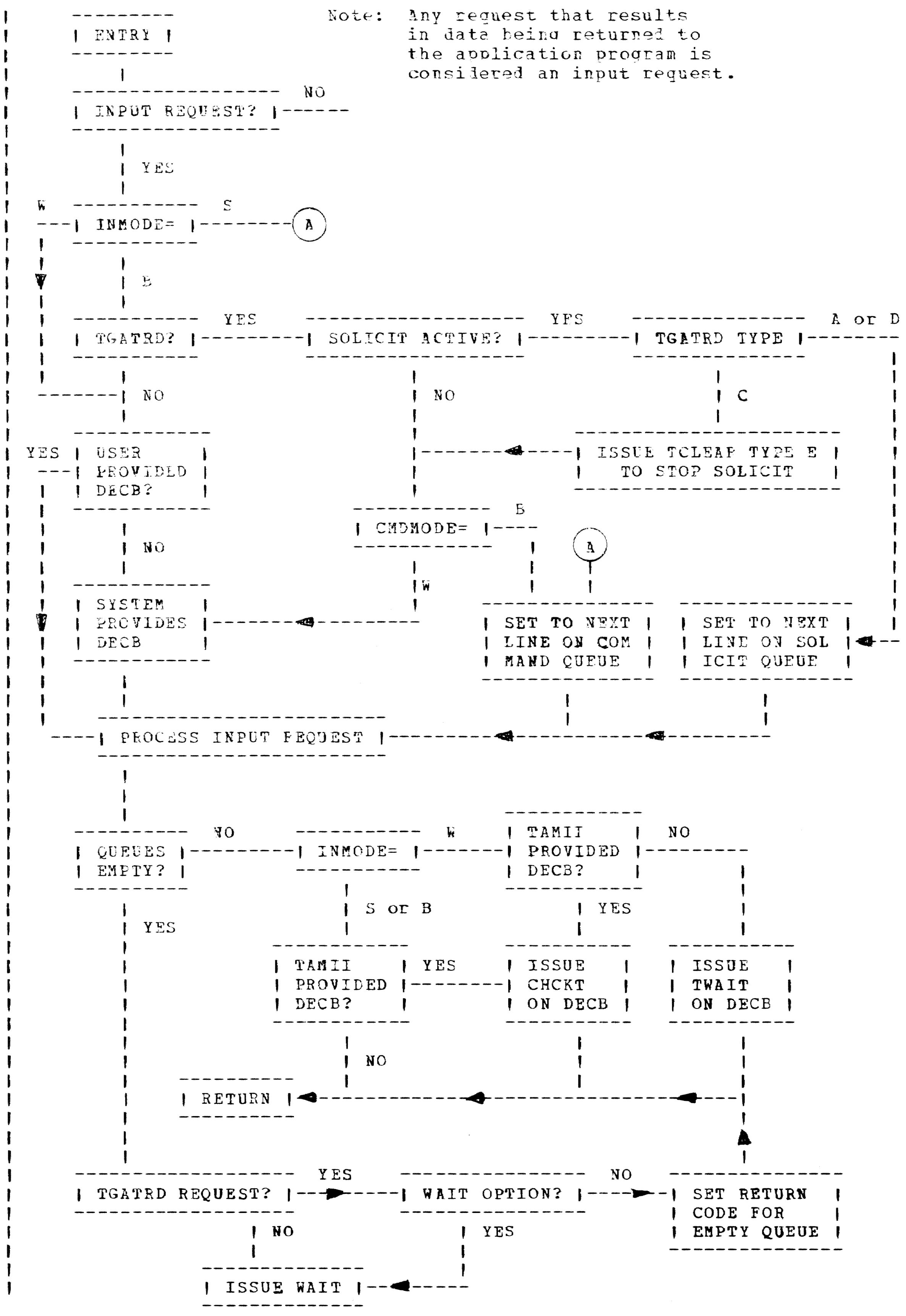


Figure 33. TAMII input request flow diagram (simplified)

INMODE controls the completion of read requests. (See Figure 33.) If INMODE=W, TAMII returns to the application program only after the read request has completed, successfully or unsuccessfully; if a DECB is associated with the read request, the return does not occur until the DECB has been posted (the same as OUTMODE). Any read request without a DECB, other than TGATRD, is handled as if INMODE=W. If INMODE=S (meaning stream input mode), TAMII tests an input queue for input; if the queue is empty and the request does not specify "wait", TAMII returns to the caller with a return code denoting no available input. If the queue is not empty, TAMII returns the first line of the queue as the requested input. If a prompt or message was associated with the request, it is ignored and not sent to the terminal (TGTWSR is an exception to this rule).

INMODE=B affects different requests in different ways: for example, it allows application programs to control buffered execution without affecting the TSS command system: any request which uses a DECB is overlapped; the SOLICIT macro is enabled and is allowed to execute asynchronously to the application program, if supported by the device support module.

When INMODE=B, TAMII maintains two input queues called queue 1 and queue 2. Queue 2 holds any input read by a SOLICIT macro; queue 1 holds any other asynchronously received input (discussed later under CMDMODE). Queue 2 is called the data queue and only a TGATRD macro with the operand TYPE=A or D (ANY or DATA) can read input from it; if TYPE=C (COMMAND), the SOLICIT operation is purged and any data in queue 2 is deleted.

When CMDMODE=W (the normal default) there is no change in the way the system works. When CMDMODE=B and INMODE=B also, the user can enter input asynchronously to the task's execution; however, this input is sent to queue 1, mentioned above, and can only be read by a TGATRD with TYPE=A or C. The main difference between INMODE=S and both INMODE=B, CMDMODE=B is the effect upon the execution of a TGTWAR macro. With INMODE=B, CMDMODE=B the prompt or message associated with a TGTWAR is displayed for the user; with INMODE=S it is not.

Figure 34 summarizes the effects that the values of INMODE, OUTMODE, and CMDMODE have on the various TAMII macros (the macros themselves will be described in detail later in this appendix).

MULTIPLE SYSIN/SYSOUT SUPPORT

The application programmer, through use of the CPO and CPI operands on TAMII macros can direct the macro's action to a specific SYSIN or SYSOUT component. If the application programmer does not code the CPO/CPI operands, the user's settings for the implicit operands SYSIN and SYSOUT determine to which component the TAMII macros will be directed.

For multiple action TAMII macros such as TGTWAR (which involves both a write and a read), different values can be specified for CPO and CPI so that a message can be written to the primary SYSOUT and input data read from a secondary (or tertiary) SYSIN component.

The valid values for CPO and CPI are given later in the discussions of the individual TAMII macros.

MACROS	DECB Y	DECB N	OUTMODE= W	OUTMODE= B	INMODE= W	INMODE= B	INMODE= S	INMODE=B CMDMODE=B
DIAL			wait	rtrn	--	--	--	--
SOLICIT			--	--	simulate prompt	execute	simulate noprompt	execute
TCNTRL			wait		--	--	--	--
	X			rtrn				
		X		depends on TYPE; see macro description				
TGATRD			--	--	see TYPE and WAIT values below			
TYPE=C					wait	wait	rtrn next Q1 line	rtrn next Q1 line
=A					wait	rtrn next Q2 line	rtrn next Q1 line	rtrn next Q1 line
=D					wait	rtrn next Q2 line	rtrn next Q1 line	rtrn next Q2 line
WAIT=Y					wait	wait	wait	wait
=N					wait	see TYPE	rtrn	see TYPE
TGATWR			wait	rtrn	--	--	--	--
TGATWS			wait	rtrn	--	--	--	--
TGTWAR			--	--	wait	wait	rtrn Q1 line; if none wait	wait
	X				--	rtrn	--	rtrn
TGTWSR			--	--	wait	wait	wait	wait
	X		--	--	--	rtrn	rtrn	rtrn
TWRTLST			wait	rtrn	--	--	--	--
Note: CMDMODE=W has no effect.								

Figure 34. Effects of TAMII implicit operands on TAMII macros

ATTENTION HANDLING SUPPORT

TAMII provides macros for handling any queued requests which may have been interrupted by an asynchronous (attention) interrupt from the (terminal) user. Through the use of the ATTNDST, ATTNRST and ATTNSAV macros, the application programmer has control of any queued requests.

After an attention interrupt the pending queue is placed in a hold state by TAMII. This hold state can only be reset by the application programmer issuing a TCNTRL with TYPE=RESTART macro. Until the RESTART request is issued, the application programmer can manipulate the pending queue using the ATTNxxx and TCLEAR macros. The ATTNxxx macros may be issued after a RESTART, but these macros only affect the pending queue and do not interfere with any active requests.

The ATTNxxx macros work on what are called levels. When the ATTNSAV macro is issued, it saves a level of information concerning the primary SYSIN and SYSOUT. Each saved level is assigned a unique level number which is used to identify the level for use with the ATTNRST and ATTNDST macros.

When the ATTNSAV macro is issued, all queued requests both output and input in both the task and the supervisor are dequeued and saved in virtual storage. Also, all request dependent information, for example the DECBs, are saved and new ones are allocated. After the information has been dequeued and saved, a special entry point in the Format Control Module for the device is called to do any device dependent processing.

To restore a previously saved level, the ATTNRST macro is issued with the level numbers to be restored given as operands. The timing of the issuance of the ATTNRST macro is important because it purges any currently pending requests before restoring the information from the specified level as the current information. Once the restore has been completed, the terminal is again at the same status as when the save was issued.

The ATTNDST macro is used to delete unwanted save levels from the ATTNSAV stack. Again, the level number to be deleted or destroyed is passed as an input operand to the macro processor.

The information saved by the ATTNSAV macro is as follows:

a. any queued asynchronous input
b. any SOLICIT input
c. any pending requests
d. the two system DECBs allocated and used by TAMII
e. any pending SOLICIT and/or locate mode input information
f. terminal status information from the work table
g. specially formatted save area for any queued requests that were queued in the supervisor
h. any device dependent information saved by the Format Control Module for the terminal

ATTNDST -- Delete Saved Attention Level (S)

The ATTNDST macro deletes a previously saved user terminal attention level.

Name	Operation	Operand
[symbol]	ATTNDST	LEVLOUT=number,LEVLIN=number [,USN=user number][,MF={I|L|(E,address of L form)}]

LEVLOUT
specifies the address of a fullword containing the number assigned to the attention level for the primary SYSOUT component that is to be deleted, or the value -1 denoting that all levels are to be deleted.

Specified as: register notation (2 through 12) or an RX address.

Default: none

LEVLIN
specifies the address of a halfword containing the number assigned to the attention level for the primary SYSIN component that is to be deleted, or the value -1 denoting that all levels are to be deleted.

Specified as: register notation (2 through 12) or an RX address.

Default: none

USN
specifies the address of a halfword containing the number assigned by TAMII to the user to be used for user identification.

Specified as: register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner

Programming note: LEVLOUT and LEVLIN must be given.

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	successful completion
X'04'	invalid level given for LEVLOUT
X'08'	invalid level given for LEVLIN
X'0C'	invalid levels given for LEVLOUT and LEVLIN
X'10'	invalid parameters given

ATTNRST -- Restore Saved Attention Level (S)

The ATTNRST macro restores a previously saved attention level.

Name	Operation	Operand
[symbol]	ATTNRST	LEVLOUT=number,LEVLIN=number [,USN=user number][,MF={I|L|(E,address of L form)}]

LEVLOUT
specifies the address of a fullword containing the number assigned to the attention level for the primary SYSOUT component that is to be restored.

Specified as: register notation (2 through 12) or an RX address.

Default: none

LEVLIN
specifies the address of a halfword containing the number assigned to the attention level for the primary SYSIN component that is to be restored.

Specified as: register notation (2 through 12) or an RX address.

Default: none

USN
specifies the address of a halfword containing the number assigned by TAMII to the user to be used for user identification.

Specified as: register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner

Programming note: LEVLOUT and LEVLIN must be given.

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	successful completion
X'04'	invalid level given for LEVLOUT
X'08'	invalid level given for LEVLIN
X'0C'	invalid levels given for LEVLOUT and LEVLIN
X'10'	invalid parameters given

ATTNSAV -- Save Current User Terminal Information (S)

The ATTNSAV macro saves current terminal information so that it can be later restored for normal processing after an attention interrupt has been processed. ATTNSAV saves information about the user's primary SYSIN and SYSOUT only; secondary and tertiary components are not supported. Up to ten saves can be recorded in a TAMII push down stack.

Name	Operation	Operand
[symbol]	ATTNSAV	[,USN=user number][,MF={I|L|(E,address of L form)}]

USN
specifies the address of a halfword containing the number assigned by TAMII to the user to be used for user identification.

Specified as: register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Return codes: upon completion of an ATTNSAV execution, registers 0 and 1 contain the save level numbers for SYSOUT and SYSIN respectively. Register 1 contents may be zero which means that the SYSOUT and SYSIN are the same (device) and only one level was created (saved). Also, register 15 contains a return code as follows:

Code	Meaning
X'00'	successful completion
X'04'	maximum levels for SYSOUT reached
X'03'	maximum levels for SYSIN reached
X'0C'	maximum levels for SYSIN and SYSOUT reached
X'10'	invalid parameter list or address

CHCKT -- Check Completion of TAMII DECB (S)

The CHCKT macro is used by TAMII applications to check the completion status of those TAMII requests for which the application program specified a DECB.

Name	Operation	Operand
[symbol]	CHCKT	DECB=decb address [,USN=user number address] [,WAIT={YES|WAIT|NO|NOWAIT}] [,TYPE={T|TWAIT|A|AWAIT}] [,MF={I|L|(E,address of L form)}]

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
address of the DECB to be marked upon completion of the DIAL request.

Specified as: register notation (2 through 12) or an RX address.

Default: none

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

WAIT
determines if CHCKT is to wait for completion of the request.

Specified as:

YES or WAIT - CHCKT waits for the DECB to be posted before returning to the caller.

NO or NOWAIT - CHCKT returns immediately to the caller, whether the request has completed or not; the user must test the return code to determine the status of the request.

Default: YES (WAIT)

TYPE
determines the type of wait if the WAIT operand is specified as YES or WAIT. For non-privileged programs, this operand is ignored by TAMII and a TWAIT is always done. For the privileged routine, this operand may be used to allow the program to synchronize with the system's schedule table.

Specified as:

T or TWAIT - a TWAIT SVC will be used.

A or AWAIT - an AWAIT SVC will be used.

Default: T (TWAIT)

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming note: TAMII overlaps requests using DECBs only if the implicit operand INMODE=B or S and/or OUTMODE=B. All output only and control requests are governed by the OUTMODE operand; all input requests are governed by the INMODE operand. The combination requests such as

TGTWAR and TGTWSR which perform a write and a read are governed by the INMODE operand.

Return codes: the valid return codes for a CHCKT request are in | register 15 (byte 3) as follows:

Code	Meaning
X'00'	request completed successfully. If input data is expected, register 0 contains the data length and register 1 contains the data address. Also, Register 15 may contain return codes in bytes 0 and 2 which describe the input record. If byte 0 of register 15 is X'80' the input is in cardboard format; if byte 0 is X'00' the input is in keyboard format. Byte 2 may contain one of the following values: X'00' - normal input X'01' - record ends with a continuation character X'02' - record truncated to fit user's input area; the rightmost characters have been lost
X'04'	request is active and NOWAIT was specified.
X'08'	attention was received on request. If input was expected register 0 contains the input length and register 1 contains the input address.
X'0C'	the CHCKT request was not processed due to a pending attention.
X'10'	request was purged by a TCLEAR macro.
X'14'	invalid for CHCKT.
X'18'	error in the CHCKT parameter list (probably an invalid or inactive DECB address).
X'1C'	invalid for CHCKT.
X'20'	invalid for CHCKT.
X'24'	terminal disconnected.
X'28'	permanent I/O error on request; sense is valid and is from the last retry.

DIAL -- DIAL a Specified Telephone Number (S)

The DIAL macro activates and executes a call-out sequence, using the hardware auto-call unit, to connect a specific terminal.

Name	Operation	Operand
[symbol]	DIAL	OUTADDR=dial digit address,OUTLGH=number of digits ,DECB={decb address|(name,Y)} [,USN=user number] [,CPO=sysout number][,CPI=sysin number] [,MF={I|L|(E,address of L form)}]

OUTADDR
address of an area containing the digits to be dialed. The digits must be a full telephone number in EBCDIC form.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
address of a fullword containing the number of dial digits pointed to by OUTADDR.

Specified as: register notation (2 through 12) or an RX address.

Default: none

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
address of the DECB to be marked upon completion of the DIAL request.

<u>Specified as</u>: register notation (2 through 12) or an RX address.

Default: none

name
the name, label or symbol to be assigned to the DECB.

Specified as: one to eight characters, the first of which must be alphabetic.

Default: none

Y
signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

USN
address of a halfword containing the number of the user for which the DIAL request is being made.

Specified as: register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

CPO
address of a halfword containing the number for the user's SYSOUT for which the DIAL request is being made.

Specified as: register notation (2 through 12) or an RX address.

Default: the value in the default SYSOUT.

CPI
address of a halfword containing a number of the specified user's SYSIN for which the DIAL request is being made.

Specified as: register notation (2 through 12) or an RX address.

Default: the value in the default SYSIN.

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming notes: CPI or CPO may be specified, but not both. Specifying both does not cause an assembly error, but TAMII will return a user error code upon execution of the DIAL macro. DIAL may be used for a communication line that has been sysgened with the auto-call feature, but if not sysgened, TAMII returns the unsupported device code.

Return codes: the following are the return codes, in register 15 following the execution of either the DIAL macro, or the CHCKT macro for the DECB assigned to the DIAL request.

Code	Meaning
X'00'	successful request
X'18'	user error in parameter list
X'20'	line does not have autocall feature or device support module does not support a DIAL request
X'24'	permanent error on DIAL
X'28'	permanent error on DIAL

SOLICIT -- Read from Specified SYSIN with Prompt (S)

The SOLICIT macro reads data input from a specified SYSIN. Each read request may be preceded with a given prompt or an incrementing number. The SOLICIT request is ended by the user entering a command break character, a null line, by an incrementing prompt reaching an ending bound, or a line count going to zero.

Name	Operation	Operand
[symbol]	SOLICIT	TYPE={C|B|N|D},USN=user number,SIC={SIC|1|2} ,CPI=sysin number [,NULL={Y|N}] [,PRMPT=prompt value,LENG=prompt length] [,INCR=prompt increment,END=prompt end value] [,NUM=number of lines to read]],MF={I|L|(E,address of L form)}]

TYPE
    specifies the type of prompt to be used.

    Specified as: N - no prompt
                   C - character prompt, no incrementing allowed
                   D - number prompt given in packed decimal
                   B - number prompt given in binary

    Default: N (no prompt)

USN
    the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

    Specified as: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

    Default: user number 0, the task's owner.

SIC
    a code identifying the level of translation and editing to be done on the input data.

    Specified as:

    SIC - data to be translated but not edited
      1 - same as SIC
      2 - data to be passed untranslated and unedited

    Default: input data will be edited and translated.

CPI
    the SYSIN component for which this request is to be executed.

    Specified as:

    0 - uses the value in the default SYSIN.
    1 - uses the primary SYSIN.
    2 - uses the secondary SYSIN.
    3 - uses the tertiary SYSIN.

    Default: uses the value in the default SYSIN.

NULL
    specifies whether or not a null line ends the SOLICIT request.

    Specified as: Y (yes) or N (no)

    Default: N

PRMPT
    specifies the address of the prompt value to be used by the SOLICIT

request when soliciting input. For TYPE=C the address is that of the prompt character string; INCR and END are ignored. For TYPE=D the address must point to a valid packed decimal number whose length is four bytes. For TYPE=B the address must point to a fullword containing the starting prompt value. For TYPE=D or B, RTAM converts the value to a printable number of the format NNNNNNNb where N is an EBCDIC digit; the number is right-justified and padded with zeroes to make the seven digits.

Specified as: register notation (2 through 12) or an RX address.

Default: none

LENG
specifies the address of a fullword value containing the length of the prompt value. If PRMPT is given, LENG must be given.

Specified as: register notation (2 through 12) or an RX address.

Default: none

INCR
specifies a fullword containing either a packed or binary number that is to be added to the prompt value after every successful read completion. The type of INCR (packed decimal or binary) must agree with the TYPE operand. A length of four bytes is assumed.

Specified as: register notation (2 through 12) or an RX address.

Default: none

END
specifies a fullword containing either a packed decimal or binary number that is to be used as a stop value for incrementing a prompt. The SOLICIT is ended when the prompt value equals or exceeds the given END value. The type of END must agree with the TYPE operand. A length of four bytes is assumed.

Specified as: register notation (2 through 12) or an RX address.

Default: none

NUM
for TYPE=C or N, specifies a fullword containing the number of lines to be read from SYSIN. The SOLICIT will be ended when the specified number of reads has been completed. The value is treated as an unsigned 32-bit logical number.

Specified as: register notation (2 through 12) or an RX address.

Default: none

Initialization: if this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Return codes: The valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	successful request
X'18'	user error in parameter list

Programming note: TAMII supports the SOLICIT macro for all SYSINs.

TCLEAR -- Purge Pending & Active I/O Requests (S)

The TCLEAR macro purges all or specific types of I/O requests from the scheduled and active request queue.

Name	Operation	Operand
[symbol]	TCLEAR	TYPE={A or ALL|O or OUTPUT|S or SOLICIT| I or INPUT|D or DECB} [,DECB={decb address|(name,Y)}][,USN=user number] [,{CPI=sysin number|CPO=sysout number}] [,MF={I|L|(E,address of L form)}]

TYPE
identifies the request(s) to be purged.

Specified as:

A or All - purge all ascheduled and active I/O, and release any buffered input records.

O or OUTPUT - purge all scheduled and active transmissions.

S or SOLICIT - purge the current SOLICIT request and any input records read by the SOLICIT request.

I or INPUT - purge all pending input records and all currently active input I/O.

D or DECB - purge the request using the given DECB address.

Default: none

DECB
for TYPE=D or DECB only; specifies the address of the DECB to be purged.

Specified as: register notation (2 through 12) or an RX address.

Default: none

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

Default: user number 0, the task's owner.

CPI
the SYSIN component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.

Default: uses the value in the default SYSIN.

CPO
the SYSOUT component for which this request is to be executed.

<u>Specified as</u>:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.

<u>Default</u>: uses the value in the default SYSOUT.

<u>Initialization</u>: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

<u>Programming note</u>: the use of the CPO or CPI operand must be consistent with the TYPE specified; for example, specifying TYPE=O and specifying CPI (instead of CPO) results in the default SYSOUT being purged and CPI is ignored.

After issuing the TCLEAR macro for a specific DECB, the DECB may then be reused without issuing a CHCKT macro.

<u>Return codes</u>: the valid return codes, in register 15, are as follows:

<u>Code</u>	<u>Meaning</u>
X'00'	request completed successfully.
X'18'	error in parameter list, invalid DECB pointer, or USN, CPO/CPI is invalid

<u>TCNTRL -- Transmit a Control Request (S)</u>

The TCNTRL macro transmits a control-type request to either TAMII or to a node or terminal; it is not normally used for data transmission.

Name	Operation	Operand
[symbol]	TCNTRL	TYPE={BELL|INHIBIT|TRSTRT or RESTART|ERASE| ENABLE or DROP|DISABLE or HOLD|PREPARE| SETCUPSR|ENABINP} [,OUTADDR=data address][,OUTLGH=data length addr] [,{CPO=sysout number|CPI=sysin number}] [,USN=user number][,DECB={decb address|(name,Y)}] [,MF={I|L|(E,address of L form)}]

TYPE
specifies the control operation to be performed.

<u>Specified as</u>:

BELL - causes the alarm-bell located on the device to be rung. If the alarm bell does not exist but is valid for the device type, the request is ignored by the hardware. If the request is invalid for the device type, a code indicating an unsupported device is returned.

INHIBIT - causes TAMII to set a software device interlock so as to prevent normal I/O to the terminal. High priority requests (BRK=Y) override this interlock.

TRSTRT or RESTART - causes TAMII to reset the software device interlock and to resume any pending I/O requests.

ERASE - causes the screen on a display terminal to be erased and the cursor to be positioned to row 0, column 0.

ENABLE or DROP - causes the terminal line to be enabled to accept incoming terminal connections or calls.

DISABLE or HOLD - causes the communication line to be reset and not accept any incoming terminal connections or calls.

(Note: the routine using this macro with an ENABLE or DISABLE type code must be privileged and must have issued a DCLASS PRIVILEGED macro before using these code types on this macro.)

PREPARE - causes an enabled line to be monitored for any terminal activity.

SETCURSR - causes the cursor on a display screen to be positioned at a specific buffer address.

ENABINP - causes a display terminal keyboard to be unlocked and input enabled. The cursor is positioned at the address in the SHDR control block.

(Note: TYPE=SETCURSR or ENABINP requires a SHDR control block as output; refer to the Terminal User's Guide.)

Default: none

The device supported type codes are as follows:

TYPE	2741	TTYs	3215	3270	3066
BELL	NS	S	S	S	S
INHIBIT	S	S	S	S	S
TRSTRT	S	S	S	S	S
RESTART	S	S	S	S	S
ERASE	NS	NS	NS	S	S
ENABLE	S	S	NS	NS	NS
DROP	S	S	NS	NS	NS
DISABLE	S	S	NS	NS	NS
HOLD	S	S	NS	NS	NS
PREPARE	S	S	NS	NS	NS
SETCURSR	NS	NS	NS	S	S
ENABINP	NS	NS	NS	S	S
N=SUPPORTED; NS=NOT SUPPORTED					

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

<u>Specified as</u>: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

<u>Default</u>: user number 0, the task's owner.

CPO
the SYSOUT component for which this request is to be executed.

<u>Specified as</u>:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.

<u>Default</u>: uses the value in the default SYSOUT.

CPI
the SYSIN component for which this request is to be executed.

<u>Specified as</u>:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.

<u>Default</u>: uses the value in the default SYSIN.

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
specifies the address of the DECB

<u>Specified as</u>: register notation (2 through 12) or an RX address.

<u>Default</u>: none

name
the name, label or symbol assigned to the DECB.

<u>Specified as</u>: one to eight characters, the first of which must be alphabetic.

<u>Default</u>: none

Y
signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

<u>Note</u>: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

<u>Initialization</u>: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

<u>Programming note</u>: privileged routines must have the address of a 76-byte save area in register 13 before executing this macro.

<u>Return codes</u>: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	request started successfully
X'04'¹	device busy,request scheduled
X'08'	attention received on this request
X'0C'¹	request not processed due to pending attention
X'10'¹	request purged by a TCLEAR request
X'14'	invalid for TCNTRL
X'18'	error in user's parameter list
X'1C'	invalid for TCNTRL
X'20'	requested operation is not supported on this device
X'24'	terminal has disconnected
X'28'¹	permanent I/O error on request

¹These return codes are invalid for TYPE=INHIBIT or TRSTRT or RESTART.

TDCMD -- Transmit Device Control Commands (S)

The TDCMD macro is used by application programs for sending Device Control Commands to the Device Control Command module.

Name	Operation	Operand
[symbol]	TDCMD	[OUTADDR=data address][,OUTLGH=data [,{CPO=sysout number|CPI=sysin number}] [,USN=user number][,MF={I|L|(E,address of I form)}]]

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

CPO
the SYSOUT component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.

Default: uses the value in the default SYSOUT.

CPI
the SYSIN component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.

Default: uses the value in the default SYSIN.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

Default: user number 0, the task's owner.

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming note: either CPO or CPI should be given; if both are given, CPO is ignored and CPI is used. If CPI is not given, the request will be issued to the SYSOUT specified by CPO, or to the default SYSOUT if CPO is also not given.

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	successful request
X'18'	invalid command or parameter in request

TFREE -- Disconnect a User or Component (S)

The TFREE macro disconnects a user or users from a task.

Name	Operation	Operand
[symbol]	TFREE	[USN=user number][,TYPE={PHD|LOG}]
		[,OUTADDR=data address][,OUTLGH=data length addr]
		[,{CPO=sysout number|CPI=sysin number}]
		[,MF={I|L|(E,address of L form)}]

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

Default: user number 0, the task's owner.

TYPE
specifies the type of disconnect to be performed.

Specified as:

PHD - physical disconnect; user cannot reconnect.

LOG - logical disconnect; user has two minutes to LOGON or reconnect to an application task.

Default: none

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

CPO
the SYSOUT component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.

Default: uses the value in the default SYSOUT.

CPI
the SYSIN component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.

Default: uses the value in the default SYSIN.

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming notes: OUTADDR and OUTLGH are used to send a message to the user/component at the time of disconnection, but these operands are ignored for SYSIN components.

If USN only is given, the user is completely disconnected from the application. If CPO or CPI is also given, the specific component is disconnected. If the CPO or CPI has more than one node connected, only the top active node is disconnected.

If a specific node is connected as both a SYSOUT component and a SYSIN component, and TFREE is issued against either component, the node will be disconnected from both components.

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	successful request
X'18'	error in user's parameter list

TGATRD -- Get a Record from Specified SYSIN (S)

The TGATRD macro retrieves a record from a specified SYSIN and makes the data available to the application program.

Name	Operation	Operand
[symbol]	TGATRD	INADDR=data address,INLGH=data length address [,MODE={M|L}][,SIC={SIC|1|2}][,USN=user number] [,TYPE={C|D|A}][,WAIT={Y|N}][,CPI=sysin number] [,MF={I|L|(E,address of L form)}]

INADDR
address of the area to receive data to be sent to the application. This operand is not required if MODE=L.

<u>Specified as</u>: an address in register notation (2 through 12) or an RX address.

<u>Default</u>: none

INLGH
the length of the data pointed to by INADDR; maximum length is 4000 bytes. This operand is not required if MODE=L.

<u>Specified as</u>: the address of a fullword in register notation (2 through 12) or an RX address.

<u>Default</u>: none

MODE
specifies how TAMII is to handle the input area.

<u>Specified as</u>:

M - input data is moved in to the user-provided area indicated by the INADDR and INLGH operands; on return, register 0 contains the length of the data and register 1 points to the input area.

L - input data is placed in a system allocated buffer; on return, register 0 contains the length of the data, and register 1 points to the buffer. The buffer is released after the next request to schedule input is received. When MODE=L, INADDR and INLGH are ignored, if specified.

<u>Default</u>: M

SIC
a code identifying the level of translation and editing to be done on the input data.

<u>Specified as</u>:

SIC - data to be translated but not edited
  1 - same as SIC
  2 - data to be passed untranslated and unedited

<u>Default</u>: input data will be edited and translated.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

<u>Specified as</u>: the address of a halfword in register notation (2 through 12) or an RX address.

<u>Default</u>: user number 0, the task's owner.

TYPE
specifies the type of input data to be read.

Specified as: C - command input
D - data input
A - any available input

Default: A

WAIT
specifies whether or not the task waits for a read completion before returning to the caller.

Specified as: Y (yes) or N (no); if specified as N and no data is in the queue, an X'1C' code (no input available) is returned to the application program.

Default: Y

CPI
the SYSIN component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.
L - uses the internal queue.

Default: uses the value in the default SYSIN.

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

| Return codes: the valid return codes, in register 15 (byte 3) , are as follows:

Code	Meaning
X'00'	successful completion. Also, Register 15 may contain return codes in bytes 0 and 2 which describe the input record. If byte 0 of register 15 is X'80' the input is in cardboard format; if byte 0 is X'00' the input is in keyboard format. Byte 2 may contain one of the following values: X'00' - normal input; X'01' - record ends with a continuation character; X'02' - record truncated to fit user's input area; the rightmost characters have been lost
X'04'	device busy, request scheduled
X'08'	attention received on this request
X'0C'	request not processed due to pending attention
X'10'	request purged by a TCLEAR request
X'14'	EOD on a SOLICIT request
X'18'	error in user's parameter list
X'1C'	no input available to fulfill request
X'24'	terminal has disconnected
X'28'	permanent I/O error on request

TGATWR -- Put a Record on Specified SYSOUT (S)

The TGATWR macro instruction schedules a record to be transmitted to a specified user's SYSOUT.

Name	Operation	Operand
[symbol]	TGATWR	[OUTADDR=data address][,OUTLGH=data length addr] [,SIC={SIC|1|2}][,USN=user number] [,DECB={decb address|(name,Y)}][,CPO=sysout number] [,BRK={Y|N}][,CC={Y|N}] [,MF={I|L|(E,address of L form)}]

Note: if the E and L form pair of this macro is used the SIC, BRK, and CC operands must be specified on the E form.

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

SIC
a code identifying the level of translation and editing to be done on the output data.

Specified as:

SIC - data to be translated but not edited
1 - same as SIC
2 - data to be passed untranslated and unedited

Default: output data will be edited and translated.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

Default: user number 0, the task's owner.

DECB
identifies the 46-byte DECB area to be used by TAMII.

decb address
specifies the address of the DECB

Specified as: register notation (2 through 12) or an RX address.

Default: none

name
the name, label or symbol assigned to the DECB.

Specified as: one to eight characters, the first of which must be alphabetic.

Default: none

Y

signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

Note: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the DECB.

CPO

the SYSOUT component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.
L - uses the internal queue.

Default: uses the value in the default SYSOUT.

BRK

denotes the priority of the request.

Specified as:

Y - top priority; this request will be scheduled ahead of any pending requests and will also interrupt any currently active request.

N - not a priority request.

Default: N

CC

specifies whether or not the output data is preceded by a carriage control character.

Specified as: Y (yes) or N (no).

Default: N

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming note: prior to TAMII, a GATWR followed by a GTWRC followed by another GATWR in a nonconversational task would have caused a skip to a new page. This does not happen in TAMII. To skip a page in TAMII, issue TGATWR with CC=Y (the output data must start with the character "1").

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	request started successfully
X'04'	(for MTT only) scheduling this output request has caused the specified SYSOUT to reach it's buffer limit. Any more requests should be delayed until an output complete return is received from a FINDQ.
X'08'	attention received on this request; this return is possible only if OUTMODE=W.
X'0C'	the normal attention return code if the user presses attention key while the request is being scheduled for transmission.

X'18'	user has passed an invalid parameter address or the length is zero, or greater than 4000. If the DECB parameter is used, this return code is received if the DECB is still marked active for a previous request.
X'24'	see X'28'
X'28'	causes an ABEND when the primary SYSIN/SYSOUT was used; reflects a permanent I/O error for all other SYSIN/SYSOUTs.

Programming note: if a DECB operand is specified, the DECB may not be reused until either a CHCKT or a TCLEAR with TYPE=D has been issued against the DECB; otherwise, the request will be denied and a X'24" code will be returned.

TGATWS -- Write to User's SYSOUT (S)

The TGATWS macro schedules a record to be transmitted to a specified user's primary SYSOUT.

Name	Operation	Operand
[symbol]	TGATWS	[OUTADDR=data address][,OUTLGH=data length] [,SIC={SIC|1|2}][,USN=user number] [,DECB={decb address|(name,Y)}][,BRK={Y|N}] [,CC={Y|N}[,MF={I|L|(E,address of L form)}]

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

SIC
a code identifying the level of translation and editing to be done on the output data.

Specified as:

SIC - data to be translated but not edited
1 - same as SIC
2 - data to be passed untranslated and unedited

Default: output data will be edited and translated.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
    specifies the address of the DECB

    Specified as: register notation (2 through 12) or an RX address.

    Default: none

name
    the name, label or symbol assigned to the DECB.

    Specified as: one to eight characters, the first of which must be alphabetic.

    Default: none

Y
    signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

Note: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

BRK
    denotes the priority of the request.

    Specified as:

    Y - top priority; this request will be scheduled ahead of any pending requests and will also interrupt any currently active request.

    N - not a priority request.

    Default: N

CC
    specifies whether or not the output data is preceded by a carriage control character.

    Specified as: Y (yes) or N (no).

    Default: N

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	request started successfully
X'04'	(for MTT only) scheduling this output request has caused the specified SYSOUT to reach it's buffer limit. Any more requests should be delayed until an output complete return is received from a FINDQ.
X'08'	attention received on this request; this return is possible only if OUTMODE=W.
X'0C'	the normal attention return code if the user presses attention key while the request is being scheduled for transmission.
X'18'	user has passed an invalid parameter address or the length is zero, or greater than 4000. If the DECB parameter is used, this return code is received if the DECB is still marked active for a previous request.
X'24'	see X'28'

X'28' causes an ABEND when the primary SYSIN/SYSOUT was used; reflects a permanent I/O error for all other SYSIN/SYSOUTs.

TGTWAR -- Write and Read (S)

The TGTWAR macro schedules a transmission on the specified user's SYSOUT of the data specified by the OUTADDR and OUTLGH operands; it also moves any data from the queue or terminal into the area specified by the INADDR and INLGH operands.

Name	Operation	Operand
[symbol]	TGTWAR	[,OUTADDR=data address] [,OUTLGH=data length] [,INADDR=data address,INLGH=data length address] [,SIC={SIC|1|2|3|4|5|6|7|8} [,USN=user number] [,DECB=decb address] [,BRK={Y|N} [,MODE={M|L}] [,{CPI=sysin number|CPO=sysout number}] [,CC={Y|N}] [,MF={I|L|(E,address of L form)}]

Note: if the E and L form pair of this macro is used, the SIC, BRK, CC and MODE operands must be specified on the E form.

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

INADDR
address of the area to receive data to be sent to the application.

Specified as: an address in register notation (2 through 12) or an RX address.

Default: none

INLGH
the length of the data pointed to by INADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

SIC
a code identifying the level of translation and editing to be done on the input/output data.

Specified as:

SIC - input data to be translated but not edited; output handled normally
1 - same as SIC
2 - output data to be translated but not edited; input handled normally

3 - both input and output to be translated but not edited
4 - input data to be neither edited nor translated; output handled normally
5 - output data to be neither edited nor translated; input handled normally
6 - neither input nor output data is to be edited or translated
7 - input data to be neither edited nor translated; output data to be translated but not edited
8 - output data to be neither edited nor translated; input data to be translated but not edited

<u>Default</u>: both input and output data will be edited and translated.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

<u>Specified as</u>: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

<u>Default</u>: user number 0, the task's owner.

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
specifies the address of the DECB

<u>Specified as</u>: register notation (2 through 12) or an RX address.

<u>Default</u>: none

name
the name, label or symbol assigned to the DECB.

<u>Specified as</u>: one to eight characters, the first of which must be alphabetic.

<u>Default</u>: none

Y
signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

<u>Note</u>: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

BRK
denotes the priority of the request.

<u>Specified as</u>:

Y - top priority; this request will be scheduled ahead of any pending requests and will also interrupt any currently active request.

N - not a priority request.

<u>Default</u>: N

MODE
specifies how TAMII is to handle the input area.

<u>Specified as</u>:

M - input data is moved in to the user-provided area indicated by the INADDR and INLGH operands; on return, register 0 contains the length of the data and register 1 points to the input area.

L - input data is placed in a system allocated buffer; on return, register 0 contains the length of the data, and register 1 points to the buffer. The buffer is released after the next request to schedule input is received.

Default: M

CPI

the SYSIN component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.
L - uses the internal queue.

Default: uses the value in the default SYSIN.

CPO

the SYSOUT component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.
L - uses the internal queue.

Default: uses the value in the default SYSOUT.

CC

specifies whether or not the output data is preceded by a carriage control character.

Specified as: Y (yes) or N (no).

Default: N

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming note: if the user or the application program is using any of the following implicit operands:

INMODE=S SYSIN=L CPI=L

the output transmission is not scheduled, but ignored, and the next input record from the input stack is returned to the caller.

| Return codes: the valid return codes, in register 15 (byte 3), are as follows:

Code	Meaning
X'00'	successful completion. Also, Register 15 may contain return codes in bytes 0 and 2 which describe the input record. If byte 0 of register 15 is X'80' the input is in cardboard format; if byte 0 is X'00' the input is in keyboard format. Byte 2 may contain one of the following values:

X'00'	- normal input
X'01'	- record ends with a continuation character
X'02'	- record truncated to fit user's input area; the rightmost characters have been lost

X'04'	device busy, request scheduled
X'08'	attention received on this request
X'0C'	request not processed due to pending attention
X'10'	request purged by a TCLEAR request
X'18'	error in user's parameter list
X'24'	terminal has disconnected
X'28'	permanent I/O error on request

TGTWSR -- Write with Synchronous Response (S)

The TGTWSR macro transmits the data pointed to by the OUTADDR operand to the user's primary SYSOUT and returns to the application program the user's response from the user's primary SYSIN. Use of this macro in a nonconversational task causes termination of the task after the data has been transmitted to the user's primary SYSOUT if the user is the task owner.

```
┌─────────┬─────────┬──────────────────────────────────────────────────────┐
|Name     |Operation|Operand                                               |
├─────────┼─────────┼──────────────────────────────────────────────────────┤
|[symbol ]|TGTWSR   |[OUTADDR=data address][,OUTLGH=data length]           |
|         |         |[,INADDR=data address,INLGH=data length address]      |
|         |         |[,SIC={SIC|1|2|3|4|5|6|7|8}[,USN=user number]         |
|         |         |[,DECB={decb address|(name,Y)}][,BRK={Y|N}            |
|         |         |[,MODE={M|L}]                                         |
|         |         |[,{CPI=sysin number|CPO=sysout number}]               |
|         |         |[,CC={Y|N}][,MF={I|L|(E,address of L form)}]          |
└─────────┴─────────┴──────────────────────────────────────────────────────┘
```

Note: if the E and L form pair of this macro is used, the SIC, BRK, CC and MODE operands must be specified on the E form.

OUTADDR
address of the data to be transmitted from the application.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
the length of the data pointed to by OUTADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

INADDR
address of the area to receive data to be sent to the application.

Specified as: an address in register notation (2 through 12) or an RX address.

Default: none

INLGH
the length of the data pointed to by INADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

SIC
a code identifying the level of translation and editing to be done on the input/output data.

Specified as:

SIC - input data to be translated but not edited; output handled normally
1 - same as SIC
2 - output data to be translated but not edited; input handled normally
3 - both input and output to be translated but not edited
4 - input data to be neither edited nor translated; output handled normally
5 - output data to be neither edited nor translated; input handled normally
6 - neither input nor output data is to be edited or translated
7 - input data to be neither edited nor translated; output data to be translated but not edited
8 - output data to be neither edited nor translated; input data to be translated but not edited

Default: both input and output data will be edited and translated.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
specifies the address of the DECB

Specified as: register notation (2 through 12) or an RX address.

Default: none

name
the name, label or symbol assigned to the DECB.

Specified as: one to eight characters, the first of which must be alphabetic.

Default: none

Y
signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

Note: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

BRK
denotes the priority of the request.

Specified as:

Y - top priority; this request will be scheduled ahead of any pending requests and will also interrupt any currently active request.

N - not a priority request.

Default: N

MODE
specifies how TAMII is to handle the input area.

Specified as:

M - input data is moved in to the user-provided area indicated by the INADDR and INLGH operands; on return, register 0 contains the length of the data and register 1 points to the input area.

L - input data is placed in a system allocated buffer; on return, register 0 contains the length of the data, and register 1 points to the buffer. The buffer is released after the next request to schedule input is received.

Default: M

CC
specifies whether or not the output data is preceded by a carriage control character.

Specified as: Y (yes) or N (no).

Default: N

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Return codes: the valid return codes, in register 15 (byte 3), are as follows:

Code	Meaning
X'00'	successful completion. Also, Register 15 may contain return codes in bytes 0 and 2 which describe the input record. If byte 0 of register 15 is X'80' the input is in cardboard format; if byte 0 is X'00' the input is in keyboard format. Byte 2 may contain one of the following values: X'00' - normal input X'01' - record ends with a continuation character X'02' - record truncated to fit user's input area; the rightmost characters have been lost
X'04'	device busy, request scheduled
X'08'	attention received on this request
X'0C'	request not processed due to pending attention
X'10'	request purged by a TCLEAR request
X'18'	error in user's parameter list
X'24'	terminal has disconnected
X'28'	permanent I/O error on request

Programming note: the TGTWSR macro can be used by the application program to synchronize an input record with an output record when the implicit operand INMODE=S. TAMII will transmit the message (output data) to the primary SYSOUT and then will return the user's response to the message as the input data. For nonconversational task owner's primary SYSIN/SYSOUT, input is always synchronized to output messages.

TRCBUF -- Read Terminal's Buffer (3270 & 3066) (S)

The TRCBUF macro retrieves a line from the terminal's conversational buffer.

Name	Operation	Operand
[symbol]	TRCBUF	[OUTADDR =data address][,OUTLGH=data [,INADDR=data address,INLGH=data length address] [,DECB={decb address | (name,Y)}][,USN=user number] [,{CPI=sysin number | CPO=sysout number}] [,MODE={M | L}][,MF={I | L | (E,address of L form)}]

OUTADDR

address of an area that contains the frame number in EBCDIC of the line to be retrieved or a single blank character to denote a read of the next line.

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
a fullword containing the length of the frame number.

Specified as: register notation (2 through 12) or an RX address.

Default: none

INADDR
address of the area to receive data to be sent to the application.

Specified as: an address in register notation (2 through 12) or an RX address.

Default: none

INLGH
the length of the data pointed to by INADDR; maximum length is 4000 bytes.

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
specifies the address of the DECB

Specified as: register notation (2 through 12) or an RX address.

Default: none

name
the name, label or symbol assigned to the DECB.

Specified as: one to eight characters, the first of which must be alphabetic.

Default: none

Y
signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

Note: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

CPI
the SYSIN component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSIN.
1 - uses the primary SYSIN.
2 - uses the secondary SYSIN.
3 - uses the tertiary SYSIN.

Default: none

CPO
the SYSOUT component for which this request is to be executed.

Specified as:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.

Default: none

MODE
specifies how TAMII is to handle the input area.

Specified as:

M - input data is moved in to the user-provided area indicated by the INADDR and INLGH operands; on return, register 0 contains the length of the data and register 1 points to the input area.

L - input data is placed in a system allocated buffer; on return, register 0 contains the length of the data, and register 1 points to the buffer. The buffer is released after the next request to schedule input is received.

Default: M

Initialization: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

Programming note: either CPO or CPI should be given, but not both. If both are given, CPO is ignored and CPI is used. If CPI is not given, the request will be issued to the SYSOUT specified by CPO, or to the default SYSOUT if CPO is also not given.

Return codes: the valid return codes, in register 15, are as follows:

Code	Meaning
X'00'	successful request
X'08'	reached end of conversational buffer
X'18'	invalid parameter in request
X'24'	TPCBUF is not supported by specified SYSIN/SYSOUT

Note: the returned data line is in the following format:

```
XXXXYYYZtext of line in EBCDIC
```

where XXXX=the frame number, YYY=the line number, and Z=the attribute character for the line.

TREAD -- Device Dependent Direct Control Read (3270s only) (S)

The TREAD macro requests an exact read type to be performed by TAMII; the data read is returned to the application program as received by TAMII.

Name	Operation	Operand
symbol	TREAD	INADDR=address of input area ,INLGH=address of length of input area ,TYPE={RDBUF|RDMOD} [,CPI=address of component number] [,DECB={decb address|(name,Y)}] [,USN=address of user number] [,MF={L|(E,address of list)}]

INADDR=
: the address of an area where the data, received by TAMII after the execution of the TREAD request, is to be moved. The data will be preceded by a 16-byte header created by TAMII. (See DSECT CHASHDR for a description of the header.)

  Specified as: register notation (2 through 12) or an RX address.

  Default: none

INLGH=
: the address of a fullword containing the length of the INADDR input area.

  Specified as: register notation (2 through 12) or an RX address.

  Default: none

TYPE=
: a symbol defining the type of read channel program to be performed by TAMII.

  Specified as:

  RDBUF - perform a read full buffer operation (causes execution of an X'02' channel command word).

  RDMOD - perform a read modified fields operation upon receipt of an attention (causes execution of an X'06' channel command word).

  Default: none

CPI=
: the address of a halfword containing the component number of the specified user's SYSIN for which the TREAD request is destined.

  Specified as: register notation (2 through 12) or an RX address.

  Default: uses the value in the default SYSIN.

DECB=
: identifies the DECB to be marked upon completion of the TWRITE request.

decb address
: specifies the address of the DECB

  Specified as: register notation (2 through 12) or an RX address.

  Default: none

name
: the name, label or symbol assigned to the DECB.

Specified as: one to eight characters, the first of which must be alphabetic.

Default: none

Y

signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

Note: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

USN=

the address of a halfword containing the number of the user for which the TREAD request is destined.

Specified as: register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

Initialization: if this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming notes: this macro is only supported for use with the 3270 display terminals. The use of this macro allows the application program to directly control the 3270 terminal. TAMII will pass to the user all data received from the 3270 terminal in response to the execution of the TREAD macro, except for a PA1 response. TAMII maintains control of the PA1 key for all cases.

The data moved to the input area will be preceded by a 16-byte header described by the DSECT CHASHDR. This header area is added by the I/O section of TAMII and may or may not contain useful information depending on the previous write requests.

The TREAD with TYPE=RDBUF is scheduled by TAMII to be executed as soon as any active and pending requests have been executed. A TREAD with TYPE=RDMOD is not executed until the receipt of an attention interrupt from the 3270 device. The TREAD with TYPE=RDMOD, upon reaching the top of the pending queue, causes any following requests to remain pending until the user has pressed one of the attention interrupt keys. The modified fields are read and the data is made available to the application program upon completion of the TREAD with TYPE=RDMOD request.

Return codes: upon return from a TREAD macro instruction registers 0 and 1 will contain the length and address respectively, of the input data, and register 15 (byte 3) will contain a zero return code. If a DECB is specified or if register 15 is not zero, the contents of registers 0 and 1 are not relative to the completion of the request.

The following return codes are valid for a TREAD macro instruction:

Code	Meaning
X'00'	successful completion. Also, Register 15 may contain returrn codes in bytes 0 and 2 which describe the input record. If byte 0 of register 15 is X'80' the input is in cardboard format; if byte 0 is X'00' the input is in keyboard format. Byte 2 may contain one of the following values:

	X'00' - normal input
	X'02' - record truncated to fit user's input area; the rightmost characters have been lost
X'08'	attention received while request was active
X'0C'	attention received while request was pending
X'10'	request purged by a TCLEAR macro instruction
X'18'	user error in parameters specified
X'20'	request issued to a device other than a 3270
X'28'	permanent I/O error on request

TWRITE -- Device Dependent Direct Control Write (3270s only) (S)

The TWRITE macro instruction is used to request a specific write type to be performed by TAMII. The application program is responsible for the data stream and display formatting.

Name	Operation	Operand
symbol	TWRITE	TYPE=symbol, OUTADDR=address of output data ,OUTLGH=address of length of output data [,INADDR=in area addr,INLGH=in area length addr] [,DECB={decb address|(name,Y)}]],BRK={Y|N}] [,USN=address of user number] [,CPO=address of output component] [,MF={L|(E,addr list)}]

TYPE=
  identifies the type of write to be performed.

  Specified as:

WRITE	- write text to display (execution of an X'05' channel command word).
ERASWRT	- erase display and write text to display (execution of an X'05' channel command word).
WRTRD	- write text to display and read response (execution of an X'01' channel command word followed by execution of an X'06' channel command word upon receipt of an attention interrupt).
ERAWRTR	- erase display, write text to display, and read response (execution of an X'05' channel command word followed by execution of an X'06' channel command word upon receipt of an attention interrupt).

  Default: none

OUTADDR=
  the address of the formatted data stream to be written to the device. For 3277s, TAMII requires that the text be preceded by a 16-byte header. (See DSFCT CHASHRD for the header description.)

  Specified as: register notation (2 through 12) or an RX address.

  Default: none

OUTLGH=
  the address of a fullword containing the length of the data stream to be written plus 16 bytes for the required header.

  Specified as: register notation (2 through 12) or an RX address.

  Default: none

INADDR=
  the address of an input area where the data read by TAMII is to be moved.

  Specified as: register notation (2 through 12) or an RX address.

  Default: none

INLGH=
the address of a fullword containing the length of the INADDR input area.

Specified as: register notation (2 through 12) or an RX address.

Default: none

DECB=
identifies the DECB to be marked upon completion of the TWRITE request.

decb address
specifies the address of the DECB

Specified as: register notation (2 through 12) or an RX address.

Default: none

name
the name, label or symbol assigned to the DECB.

Specified as: one to eight characters, the first of which must be alphabetic.

Default: none

Y
signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

Note: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 43 bytes of the previous DECB.

USN=
the address of a halfword containing the number of the user for which the TWRITE request is destined.

Specified as: register notation (2 through 12) or an RX address.

Default: user number 0, the task's owner.

CPO=
the address of a halfword containing the component number of the specified user's SYSOUT for which the TWRITE is destined.

Specified as: register notation (2 through 12) or an RX address.

Default: uses the value in the default SYSOUT.

BRK=
specifies whether or not this TWRITE request is of high priority.

Specified as:

Y - the request is of high priority; it is queued at the head of the pending I/O queue and is to be started immediately.

N - the request has no priority.

Default: N

Initialization: if this macro instruction is to be executed in a privileged module, the most recently issued DCLASS macro instruction in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro instruction is executed.

Programming notes: this macro is only supported for use with the 3270 display terminals. The use of this macro allows the application program to directly control the 3270. When using TWRITE, the application is responsible for all display formatting and keyboard resetting. It is the responsibility of the application program using a TWRITE to enable the keyboard if the user is to be able to enter input; TREAD does not do a keyboard enable.

When using TYPE=WRTRD or ERAWRTR, the operands INADDR and INLGH must be specified.

If the input options are coded, TAMII returns all input read except for a PA1 key interrupt (same as for TREAD). The PA1 interrupt is restricted by TAMII for use as the attention key. If the application program is to use the PA1 key, it must follow normal TSS attention handling procedures using SIR, USATT, and CLATT.

Upon return from a TWRITE without a DECB, if input data was to be read, registers 0 and 1 will contain the length and address respectively of the input data. If input data is not expected, the contents of registers 0 and 1 are not pertinent to the completion of the TWRITE.

| Return codes: the following return codes (register 15, byte 3) are valid for a TWRITE macro instruction:

Code	Meaning
X'00'	successful completion. Also, Register 15 may contain return codes in byte 0 and 2 which describe the input record. If byte 0 of register 15 is X'80' the input is in cardboard format; if byte 0 is X'00' the input is in keyboard format. Byte 2 may contain one of the following values: X'00' - normal input X'02' - record truncated to fit user's input area; the rightmost characters have been lost
X'06'	attention received while request was active
X'0C'	attention received while request was pending
X'10'	request purged by a TCLEAR macro instruction
X'18'	user error in parameters specified
X'20'	request issued to a device other than a 3270
X'28'	permanent I/O error on request

If the DECB operand is used, a zero return signifies that the request was scheduled for execution successfully. Any return code other than zero signifies that the condition occurred during the scheduling and the DECB is not active. Therefore, the CHCKT for the DECB should be bypassed. If a zero is returned upon execution of TWRITE, the actual completion code will be returned upon return of the CHCKT issued against the DECB assigned to this request.

TWRTLST -- Perform Gather Write (S)

The TWRTLST macro is a gather-write request. The application program passes a list of addresses and lengths and TAMII gathers all the data into one buffer and then schedules the buffer for transmission. Each entry in the list is assumed to be one printable line.

Name	Operation	Operand
[symbol]	IWRTLST	[OUTADDR=data address][,OUTLGH=data number] [,SIC={SIC|1|2}][,USN=user number] [,DECB={decb address|(name,Y)}][,CPO=sysout number] [,BRK={Y|N}][,CC={Y|N}] [,MF={I|L|(E,address of L form)}]

Note: if the E and L form pair of this macro is used the SIC, BRK, and CC operands must be specified on the E form.

OUTADDR
pointer to a list of lengths and addresses on a fullword boundary, to be transmitted: for example,

```
LGH1 ADDR1 LGH2 ADDR2 LGHN
```

Specified as: register notation (2 through 12) or an RX address.

Default: none

OUTLGH
pointer to a fullword containing the number of entries in the list (maximum of 64).

Specified as: the address of a fullword in register notation (2 through 12) or an RX address.

Default: none

SIC
a code identifying the level of translation and editing to be done on the data.

Specified as:

SIC - data to be translated but not edited
1 - same as SIC
2 - data to be passed untranslated and unedited

Default: data will be translated and edited.

USN
the number assigned by TAMII to be associated with the connected terminal that is to be scheduled for the requested operation.

Specified as: the address of a halfword in register notation (2 through 12) or an RX address, or *ALL. If specified as *ALL, or if the halfword value is set to X'FFFF', all connected MTT users SYSIN/SYSOUTs will be scheduled for the requested operation.

Default: user number 0, the task's owner.

DECB
identifies the 48-byte DECB area to be used by TAMII.

decb address
specifies the address of the DECB

Specified as: register notation (2 through 12) or an RX address.

Default: none

name
the name, label or symbol assigned to the DECB.

<u>Specified as</u>: one to eight characters, the first of which must be alphabetic.

<u>Default</u>: none

Y
: signifies that the DECB area is to be constructed as part of the macro expansion, immediately following the parameter list.

<u>Note</u>: Before using a DECB area TAMII checks to determine that the area is available for use, that is, the area is not being used by some other request; if not in use, it clears all 48 bytes of the previous DECB.

CPO
: the SYSOUT component for which this request is to be executed.

<u>Specified as</u>:

0 - uses the value in the default SYSOUT.
1 - uses the primary SYSOUT.
2 - uses the secondary SYSOUT.
3 - uses the tertiary SYSOUT.

<u>Default</u>: uses the value in the default SYSOUT.

BRK
denotes the priority of the request.

<u>Specified as</u>:

Y - top priority; this request will be scheduled ahead of any pending requests and will also interrupt any currently active request.

N - not a priority request.

<u>Default</u>: N

CC
specifies whether or not the output data is preceded by a carriage control character.

<u>Specified as</u>: Y (yes) or N (no).

<u>Default</u>: N

<u>Initialization</u>: If this macro is to be executed in a privileged module, the most recently issued DCLASS macro in the assembly must have specified PRIVILEGED. Also, the address of a save area must be placed in register 13 before this macro is executed.

<u>Return codes</u>: the valid return codes, in register 15, are as follows:

<u>Code</u>	<u>Meaning</u>
X'00'	request started successfully
X'04'	(for MTT only) scheduling this output request has caused the specified SYSOUT to reach it's buffer limit. Any more requests should be delayed until an output complete return is received from a FINDQ.
X'08'	attention received on this request; this return is possible only if OUTMODE=W.
X'0C'	the normal attention return code if the user presses attention key while the request is being scheduled for transmission.
X'18'	user has passed an invalid parameter address or the length is zero, or greater than 4000. If the DECB parameter is used, this return code is received if the DECB is still marked active for a previous request.
X'24'	see X'28'
X'28'	causes an ABEND when the primary SYSIN/SYSOUT was used; reflects a permanent I/O error for all other SYSIN/SYSOUTs.

TAMII MACRO EXAMPLES

Examples of the use of TAMII macro instructions for communicating with the task owner's SYSIN and SYSOUT are given below; these examples illustrate the following:

- writing output to SYSOUT using the various output options

- reading input from SYSIN using the various input options; the SOLICIT macro for requesting controlled input from SYSIN is also illustrated

- the TCLEAR, TCNTRL and TDCMD macros for controlling the SYSIN and/or SYSOUT

- using DECBs and the CHCKT macro for overlapped I/O and processing

- the correct method of handling attentions by an application program that has replaced the system's attention handler with its own

Writing to SYSOUT

TAMII supports a logical device concept. It is a simple device that understands an EBCDIC character string which may contain some optional control characters. When writing to this logical device, the application programmer issues a TAMII output macro (i.e., TGATWR, TGATWS or TWRTLST) pointing to a simple EBCDIC character string. TAMII will do all editing and translation required to make the output intelligible to the actual SYSOUT. For example, to write the message 'GOOD MORNING' to any SYSOUT device the application programmer could code:

```
       TGATWR  OUTADDR=MSG1,OUTLGH=LMSG1
           .
           .
           .
LMSG1  DC A(L'MSG1)
MSG1   DC C'GOOD MORNING'
```

This example results in the EBCDIC character string contained in area 'MSG1', whose length is contained in the fullword 'LMSG1' to be written to the task owner's SYSOUT specified in the SYSOUT operand or to the user's primary SYSOUT if the user's SYSOUT is defaulted. In the previous example the application programmer did not define (code) which SYSOUT would receive the message 'GOOD MORNING'. The determination of the destination SYSOUT was up to the user, who could control it by setting the SYSOUT's value in the TAMII user's operand called 'SYSOUT'.

If the application programmer has to send the message to a particular SYSOUT, regardless of the user's SYSOUT value, there are two ways of doing so. One is by using the CPO operand on TAMII macros to route the message to the required SYSOUT. The programmer may do so as follows:

```
       TGATWR  OUTADDR=MSG1,OUTLGH=LMSG1,CPO=CPO1
           .
           .
           .
LMSG1 DC  A(L'MSG1)      length or error message
CPO1  DC  H'1'           number of SYSOUT component to receive
MSG1  DC  C'PARAMETER XXX IS INVALID'
```

The above example would cause the message to be written to the user's primary SYSOUT because of the value of the CPO operand. It could also direct the message to either the user's secondary or tertiary SYSOUT by changing the value of the CPO operand to 2 or 3 respectively.

Another way the application programmer can direct the message to the user's primary SYSOUT is to use the TGATWS macro; this macro always transmits to the user's primary SYSOUT regardless of the user's SYSOUT operand specification. As a result, there is no CPO operand in the TGATWS macro. The primary SYSOUT is assumed by TAMII when a TGATWS macro is executed by an application program. The previous example could have been coded using the TGATWS macro as follows:

```
       TGATWS  OUTADDR=MSG1,OUTLGH=LMSG1
           .
           .
           .
LMSG1  DC  A(L'MSG1)
MSG1   DC  C'PARAMETER XXX IS INVALID'
```

The above example would write the error message to the user's primary SYSOUT following any other output that had been issued before the TGATWS macro. For example, if the application program had executed six TGATWSs and then executed the TGATWR or TGATWS in the previous two examples, the user would not see the error message until the first six TGATWS's output had been completed. If, however, the application programmer wanted a message or output line to be given a high priority so as to be written to the user ahead of any pending output, he can do so with the BRK operand. When BRK=Y in any TAMII output macro, TAMII schedules the output request for immediate execution; for example:

```
        TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,CPO=CPO1,BRK=Y
             .
             .
             .
LMSG1   DC   A(L'MSG1)
CPO1    DC   H'1'
MSG1    DC   C'APPLICATION SHUTDOWN SCHEDULED FOR 11:50'
```

In the above example a TGATWR macro was used with the 'break' option to send the user's primary SYSOUT a message about a pending scheduled shutdown. Instead of the TGATWR, the application programmer could have used a TGATWS to send the message. The BRK operand is valid for both macros.

TAMII supports the FORTRAN ASA carriage control characters for use with SYSOUT output requests. TAMII assumes that the first character of the output data area is the ASA carriage control when the CC=Y operand of any TAMII macro is specified. TAMII strips the first character from the output data and adds whatever control information is required to either perform or simulate the control function at the specified SYSOUT. The following example illustrates an ASA control function -- skip to a new page -- in an output write:

```
        TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,CC=Y
             .
             .
             .
LMSG1   DC   A(L'MSG1)
MSG1    DC   C'1  THIS IS A NEW PAGE HEADER WRITE'
```

For some TAMII supported units, a skip to new page function is simulated by doing a skip to new frame (3270s) or (for 2741s) a triple space followed by the write of the output line. For device implementation notes on ASA characters refer to Figure 35.

The previous examples all considered the writing of data from a single output area. TAMII supports the writing of data from multiple data areas with a 'gather write' macro, TWRTLST. This macro allows the application programmer to supply a list of output areas with the length of the data in each area. TAMII validates the list and the data and then determines the most efficient way to transmit the data to the specified SYSOUT. Both the BRK and CC options apply to the TWRTLST macro. The BRK option signifies that the whole output is to be sent high priority and the CC option signifies that each element pointed to by the data list starts with an ASA control character.

When using the TWRTLST macro, the application programmer first builds a list of 8-byte entries, starting on a fullword boundary, containing the length of the data and the address of the data to be transmitted to the user's SYSOUT. This list is pointed to by the OUTADDR operand; OUTLGH points to a fullword that contains the number of entries in the

ASA Carriage Control Simulation		
FUNCTION	CODE	SIMULATION
skip no line before printing	+	treated as a single space for terminals 2741, TTY33/35 and 3215; treated as a single space forced write by the 3270 and 3066 support
skip 1 line before printing	(blank)	same as function
skip 2 lines before printing	0	same as function
skip 3 lines before printing	-	same as function
skip to channel 1 (new page) before printing	1	treated as a triple space for terminals 2741, TTY33/35 and 3215; treated as a skip to new frame by the 3270 and 3066 support
skip to channel n before printing (n=2-12)	2-C	treated as a triple space for all terminals

Added 3270/3066 Carriage Control Characters		
FUNCTION	CODE	SIMULATION
device control command string	S	treated as a single space by the 2741, TTY33-35, and 3215 support
write bright line	*	treated as a single space by the 2741, TTY33-35, 3215 and 3066 support
skip (both 3270 and 3066) to new frame and write (3270 only) a bright line	@	treated as a single space by the 2741, TTY33-35, and 3215 support
user message prompt overlay	M	treated as a single space by 2741, TTY33-35 and 3215 support

Figure 35. ASA Characters

list. The example below shows a TWRTLST that could be used to write a formatted page of output using an ASA control character:

```
         TWRTLST  OUTADDR=LIST1,OUTLGH=NENTRY,CC=Y
            .
            .
            .
NENTRY  DC   A((LISTN-LIST1)/8)  number of entries in the list
LIST1   DC   A(L'MSG1,MSG1)                header line
        DC   A(L'MSG2,MSG2)                subheader line
        DC   A(L'MSG3,MSG3)                first text line
        DC   A(L'MSG4,MSG4)                second text line
LISTN   EQU
MSG1    DC   C'1 THIS IS THE PAGE HEADER'  skip to new page
MSG2    DC   C'  THIS IS THE SUB HEADER'   write newline
```

```
MSG3     DC   C'0 THIS NORMAL OUTPUT LINE'  double space and write
MSG4     DC   C'  THIS NEXT OUTPUT LINE'    text of page
```

The use of the TWRTLST macro is more efficient for the system and the application programmer than attempting to write each line with an individual TGATWR. With TWRTLST, an application programmer can write up to 64 lines of output as long as the amount of output data, plus the number of lines times four, is less than or equal to 4000 bytes; i.e.,

$$X + NL*4 \leq 4000$$

where X is the total amount of output data, and NL is the number of list entries.

Along with BRK and CC there is a third operand that applies to the TGATWR, TGATWS, and TWRTLST macros -- SIC. This option controls the editing and translation that is to be performed by TAMII on the output data. When this operand is not specified, the output data is searched for control characters with defined functions, and those functions are performed or the control characters are deleted if the function is not supported on the receiving SYSOUT. After the search and edit, the output data is translated, using the user's translation table and retranslated if requested using one of two special tables used to fold lower case to upper case, or to fold upper case to lower case and lower case to upper case. Finally, the output data is translated to the line code expected by the receiving SYSOUT device.

By using the SIC option the application programmer can turn off the above processes in favor of data transparency. By using one value of SIC for output, the application programmer can bypass all of TAMII's editing and translation of output data to achieve data transparency.

The following two examples show the use of two values of SIC to obtain special formatting results at the specified SYSOUT. The first, a no edit, line code translate only, for a 2741 is as follows:

```
         TGATWR  OUTADDR=MSG1,OUTLGH=LMSG1,SIC=1
             .
             .
             .
LMSG1    DC   A(L'MSG1+L'MSG2)
MSG1     DC   C'THIS UNDERLINES THE LAST WORD'
MSG2     DC   X'16161616____1517171717'    underline & do a new line
```

The next example is a no edit, no line code translate for a graphic TTY33:

```
         TGATWR  OUTADDR=MSG1,OUTLGH=LMSG1,SIC=2
             .
             .
             .
LMSG1    DC A(L'MSG1)
```

Reading from SYSIN

Since TAMII supports the logical device concept, the data received from SYSIN by the application program is in the form of a variable length record, whose length is in register 0 and whose address is in register 1. The data is an EBCDIC character stream with all control characters deleted. Each read retrieves one logical record from SYSIN. If the record is larger than the application program's input area, the record is truncated to fit, by deleting the rightmost characters and a return code indicating the truncation is set in register 15 before control is returned to the application program. The truncated data is lost. A continuation return code is set in register 15 if the data ends with a continuation character sequence which fulfills one of two continuation conventions. The conventions used are determined by the origin of the SYSIN data. If the data is entered at a keyboard, the last data character must be the continuation character, blanks are not stripped. If the data is from a data set or a card reader, then either column 72 must be a non-blank or the last non-blank data character must be a continuation character. In this case, blanks are stripped.

The TAMII macro instructions allow the application program to specify either move mode or locate mode by specifying MODE=M or MODE=L on the TAMII input request macros. Move mode causes TAMII to move the retrieved logical record to the input area specified by the macro instruction. Locate mode causes TAMII to provide a 256 byte input area where the retrieved record is placed and the address of this area is returned to the caller. On the next input request, the allocated input area is released.

An example of a normal SYSIN read follows:

```
         TGATRD   INADDR=AREA1,INLGH=LAREA1
             .
             .
             .
LAREA1 DC   A(L'AREA1)          length of the input area
AREA1  DC   XL256'00'           input area
```

Upon return from the execution of the TGATRD macro instruction register 15 will have a return code and register 0 and 1 will have the length and address of the retrieve record. In this example the address in register 1 will be the address of AREA1 and length in register 0 is the actual length of the retrieved record. The contents of LAREA1 is not changed. The above example assumed move mode because MODE=L was not specified. The following example is the same as the one above except that MODE=L was specified:

```
         TGATRD   MODE=L
             .
             .
             .
```

Since locate mode is specified the INADDR and INLGH operands are not required. On return from the execution of the TGATRD macro, register 15 contains a return code and registers 0 and 1 will contain the data's actual length and address respectively. The area pointed to by register 1 will be released by TAMII upon execution of the next input macro instruction for this particular user's SYSIN unit.

As with TGATWRs in the first two examples, both the user and the application programmer have the capability to satisfy the input request from a specific SYSIN component. The user does it by setting his TAMII implicit operand SYSIN to name the particular SYSIN component that is to be used to satisfy the request. The application programmer does it by using a TAMII macro with a CPI option and assigning to it the value of the SYSIN component to be used to satisfy the request. The following example shows a locate mode request with an input component specified:

```
         TGATRD   MODE=L,CPI=CPI3
             .
             .
             .
CPI3    DC   H'3'            retrieve record from tertiary SYSIN unit
```

So far in the examples for reading from SYSIN, TAMII would not have returned to the application program until the TGATRD request had been satisfied. If there was not a record available TAMII would have taken whatever action was required to make the record available and then have waited until the record was available before returning. When overlapped I/O and execution is allowed by the user, TAMII provides the application programmer with a WAIT operand for specifying whether TAMII is to wait until the record is available or to return.

When WAIT is not specified or is specified as Y, TAMII will not return to the application program until a record is available to fulfill the request. If WAIT=N, TAMII returns to the application immediately if there is no record available to fulfill the request. The application programmer must test the return code in register 15 to determine if the request did retrieve a record. TAMII sets no indicator for the application program when a record is available. The application program must reissue the TGATRD to determine if a record is available. The following shows the coding for a TGATRD with WAIT=N:

```
       TGATRD  MODE=L,WAIT=N
         LR   2,15              save miscellaneous return information
         N    15,=X'000000FF'   return code 0; record is available,
*                               registers 0 and 1 are valid
         BNZ  TESTRTC           no record; test further
          .
          .
          .
TESTRTC  DS   0H
         CH   R15,=Y(UNAVAIL)   unavailable input return code
         BE   NORECORD          yes, go to no input label
*                               error or attention encountered on TGATRD
```

On an attention return code (X'08'), registers 0 and 1 contain the data entered up to the attention. For all other return codes, except X'00', registers 0 and 1 are unpredictable.

Along with the TGATRD MACRO TAMII provides a complementary macro called SOLICIT. This macro is used to initialize and start the controlled reading of records from a specific SYSIN. When an application program executes a SOLICIT macro, TAMII initializes some tables and then depending on the SOLICIT operands, starts retrieving records from the specific SYSIN. These records are placed in a queue of their own, called the SOLICIT or data queue. TAMII will continue to retrieve records from the SYSIN unit until one of the specified ending conditions is met. One of these records is made available to the application program each time a TGATRD without a TYPE operand, or with a TYPE=A (any) or D (data) is executed. Once all the records have been read by the application program and the SOLICIT ending condition has been reached, an EOD (end of data) code is returned in register 15.

The SOLICIT macro allows the application program a flexible and controlled form of requesting input from the user. The SOLICIT mechanism can prompt the user with an incrementing number which also has a starting and ending number in addition to the increment value which can be specified. Also, the SOLICIT can just prompt with a static character prompt, or not prompt but just read each record as the record becomes available. The limiting or ending conditions can be specified by the application program as an ending number for an incrementing prompt, or as a number of lines to be read and/or specifying a null line (a line without data) to end the SOLICIT request. The SOLICIT request will also be ended if a TGATRD with a TYPE=C is executed. The TSS command system always reads commands using a TGATRD with TYPE=C. This prevents possible SOLICIT input from being interpreted as a command. Anexample of a SOLICIT--TGATRD use is the TSS editor requesting input for an INSERT command. The insert module would execute the following SOLICIT macro followed by a series of TGATRDs to read the retrieved records:

```
          SOLICIT  PRMPT=STARTNO,LENG=NLENG,TYPE=D,INCR=INCRNO,END=ENDNO
*           this SOLICIT starts the prompting and reading of input
          TGATRD  MODE=L,TYPE=D
*           this TGATRD now reads the input records
          CLM  R15,B'0001',=AL1(EOD)      end of SOLICIT return code?
            .
            .
            .
          BNE  READ1                      no; go read next record, etc.
            .
            .
            .
STARTNO   DC   P'1000'                starting prompt number
NLENG     DC   F'4'                   length of prompt
INCRNO    DC   P'100'                 increment
ENDNO     DC P'2500'                  ending number
```

The SOLICIT in the above example would cause TAMII to start prompting (assuming the user's INMODE=W or B) the user for input records. The first prompt would be number 1000 in the line number format of 0001000b with 'carriage' positioned after the blank. As soon as the user had entered data in response to the prompt, TAMII would add the increment to the current prompt value, test the result against the ending number, and prompt with the new number if it was less than the ending number. So the next prompt would be 0001100b, and the prompt after that would be 0001200b until 2500 was reached. Once the 2500 value was reached, the SOLICIT request would be ended and an EOD code returned to the application program on the next TGATRD after the TGATRD which had read the last input record.

Another SOLICIT example, one using an editor-like REDIT where there is no prompt and the ending condition is a null line could be coded as follows:

```
           SOLICIT  TYPE=N,NULL=Y,NUM=LARGENUM
               .
               .
               .
*            this SOLICIT will continue to read until ended by
*            the user entering a null line
           TGATRD  INADDR=AREA1,INLGH=LAREA1,TYPE=D
*            the TGATRD to do the actual transmission of data
*            from TAMII to the application program
               .
               .
               .
LARGENUM   DC   F'65000'           large number to prevent
AREA1      DC   XL256'00'          ending on read count
LAREA1     DC   F'256'
```

If the application program wanted to allow only a specific number of reads, the value for the NUM parameter would be set to this number and TAMII would end the SOLICIT when that number of reads had been reached.

There are two other ways a SOLICIT may be ended. One way is for the application program to execute a TCLEAR TYPE=E. This causes the SOLICIT and any queued input to be purged. The other way is for the user to enter a data line which starts with a single command break character. This causes TAMII to force an early end to the SOLICIT input queue and the record which started with the command break character is placed on the command queue to be read by the next TGATRD with TYPE=C.

TAMII provides the application programmer with two other macro instructions to use when the user has to be prompted for input. One is TGTWAR -- write with available response. It is used to write an output record to a specific SYSOUT and to return the next available input record from a specific SYSIN. To use TGTWAR, the application programmer could code:

```
        TGTWAR  OUTADDR=MSG1,OUTLGH=LMSG1,INADDR=AREA1,
                INLGH=LAREA1
           .
           .
           .
LMSG1   DC  A(L'MSG1)
LAREA1  DC  A(L'AREA1)
MSG1    DC  C'READY FOR NEXT COMMAND. ENTER'
AREA1   DC  XL256'00'
```

TGTWAR should be used for user predictable sequences and prompts. When the user is running with INMODE=S, TAMII ignores the output portion of a TGATWAR and just returns the next queued input line as the input record. By using TGTWAR for predictable prompts, the user when using INMODE=S can work ahead of the user's task because the user can predict with reasonable accuracy what the task is going to do.

Since the TGTWAR is just a contraction of a TGATWR WITH A TGATRD, the several options that apply to the TGATWR and TGATRD macros also apply to the TGTWAR macro. TGTWAR does not support the TYPE and WAIT options of the TGATRD. As with the TGATWR and TGATRD macros the TGTWAR macro may be directed to a specific SYSOUT and SYSIN. The following example shows a TGTWAR which uses the BRK and CC options of TGATWR and the locate mode option of a TGATRD:

```
        TGTWAR  OUTADDR=MSG1,OUTLGH=LMSG1,CC=Y,BRK=Y,MODE=L,
                CPO=CPO1,CPI=CPI1
           .
           .
           .
LMSG1   DC  A(L'MSG1)
MSG1    DC  C'0 DOUBLE SPACE AND READ INPUT'
CPO1    DC  H'1'       write and read the primary SYSOUT and SYSIN
```

Upon completion of the TGTWAR macro in the last two examples, registers 0 and 1 will contain the length and address of the input record and register 15 will contain any associated return code.

For unexpected or unpredictable prompts that require a response, the application programmer should use the TGTWSR macro. TGTWSR is a write and read synchronous response operation directed to the primary SYSIN and SYSOUT. This macro is normally used by application programs to request missing input parameters or when error conditions are encountered and the user is prompted as to what action is to be taken by the application program. In the example that follows the application program has encountered some errors attempting to process certain data records and prompts the user as to whether the records would be included in the report or deleted from the report. If the user is using INMODE=S, the user may already have other command and input records enqueued waiting to be processed. If the application program executed a TGTWAR, TAMII would ignore the message and return the next available input line. So in the following example the application programmer has coded a TGTWSR macro using ASA carriage control and locate mode input:

```
        TGTWSR  OUTADDR=MSG1,OUTLGH=LMSG1,CC=Y,MODE=L
           .
           .
           .
LMSG1  DC  A(L'MSG1)
MSG1   DC  C'ENTER D TO DELETE OR A TO ADD ERROR RECORDS TO REPORT'
```

Upon completion of the TGTWSR macro in the above example, registers 0 and 1 will contain the length and address of the input record and register 15 will contain any associated return code.

As with the output macro instructions TGATWR, TGATWS, and TWRTLST, TAMII supports an input transparency mode. The SIC option is used to determine the amount of editing and translation to be done to the input record.

The normal TAMII sequence of translation and editing for input is to first translate the input record from transmission line code to EBCDIC. The input record is translated again to either fold lower case to upper case or to reverse lower case to upper case and upper case to lower case. Next the input is scanned for control characters. The function defined for the control character is performed and the control character is deleted from the input record. And finally, the input record is translated again, using the user's input translate table.

By specifying a value for SIC, the normal TAMII data manipulation can be bypassed. The simple SIC value of 1 for TGATRD and a SOLICIT macro causes TAMII to bypass all editing and translations except the line code translate. A SIC value of 2 sets data transparency and the data will be passed to the application in the form received from the user's SYSIN unit. The SIC operand for TGTWAR and TGTWSR is more complicated because both input and output can be treated separately.

When using the SIC option with SOLICIT, both the SOLICIT and the TGATRD macro instructions should specify the same SIC values; if the SIC values are different, the results will be unpredictable.

Miscellaneous Macros for Controlling SYSIN & SYSOUT

Besides macros for moving data between the user and an application program, TAMII provides two macros which are used to directly control the I/O queues for the user's unit and to perform many device dependent miscellaneous control functions. The TCLEAR macro is used to purge specific requests or types of requests from both the pending and active request queues and the pending input queues. The TCNTRL macro can be used to control the stopping and restarting of the pending request and to perform specific control functions at the user's specific SYSIN/SYSOUT unit.

The TCLEAR macro purges requests from TAMII's pending I/O queue. The pending I/O queue has requests -- TGATWR, TGATRD, etc. -- which are scheduled but have not yet been started because the unit is busy with a previous request. The TCLEAR macro also stops and purges any active I/O if the active I/O meets the specified TCLEAR condition. Finally, the TCLEAR macro can be used to purge the pending input queues of input records.

The application program identifies the purge condition by the TYPE operand on the TCLEAR macro.

When TYPE=A (or ALL), TAMII removes and releases all pending and active I/O requests and releases any pending input records. For requests that have a DECB assigned, TAMII marks the DECB purged. Having been marked purged, the DECB is available for reuse without the need of a CHCKT macro. If the application program executed:

```
        TCLEAR   TYPE=A
```

all queues, upon return from TAMII, would be empty, any active DECBs would be marked purged, and the user's unit would be in an idle state.

The TCLEAR is always performed synchronously to the application program; i.e., TAMII does not return until the TCLEAR request has completed.

TYPE=O or OUTPUT causes TAMII to remove and release any pending output requests. If the request is active, it is halted and released and any DECBs assigned to it are marked purged. Output requests are TGATWR, TGATWS, TCNTRL, and TWRITE. (The TGTWAR, TGTWSR and TWRITE (with response) macros are considered input requests by TAMII, since their main function is to read input.) All other requests are not affected by this TYPE operand. In the following example the TGATWR would be purged by the following TCLEAR request, but the TGTWAR would be unaffected:

```
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1
            .
            .
            .
         TGTWAR   OUTADDR=MSG2,OUTLGH=LMSG2,MODE=L
            .
            .
            .
         TCLEAR   TYPE=O
            .
            .
            .
LMSG1    DC   A(L'MSG1)
LMSG2    DC   A(L'MSG2)
MSG1     DC   C'MESSAGE 1'
MSG2     DC   C'MESSAGE 2'
```

TYPE=I or INPUT is used to purge any pending input requests and any input which has already been read by TAMII. Input requests are SOLICIT, TGATRD, TGTWAR, TGATWSR, and TWRITE (with input). Any other requests are not affected by this TYPE operand. If an input request is active, it is halted and released. Any DECBs associated with it would be marked purged.

TYPE=E or SOLICIT is used to purge a pending or active SOLICIT request and any input read by the SOLICIT. Upon issuance of the TCLEAR macro the SOLICIT request is halted, if active, and released. Any input which has been read by the SOLICIT request is released.

Caution: When INMODE=S any input in the input queue is considered SOLICIT input and is released.

TYPE=D or DECB is used to purge a particular request. When a TCLEAR with TYPE=D is issued, all requests are searched to locate the request associated with the specified DECB address. When the request is found, it is halted if active, and released, and the DECB is marked purged.

In the following example the TGATWR is associated with a DECB. After issuing the TGATWR, the application program decides to cancel the TGATWR request and only that request. This is accomplished by issuing the TCLEAR request with the DECB address and type code. This is the only way to cancel a particular request in TAMII. If the request does not have a DECB associated with it, it is not possible to do a specific cancel:

```
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,DECB=DECBA
            .
            .
            .
         TCLEAR   TYPE=D,DECB=DECBA
            .
            .
```

```
       CHCKT  DECB=DECBA
DECBA  DC   12F'0'
LMSG1  DC   A(L'MSG1)
MSG1   DC   C'MESSAGE 1'
```

The CHCKT in the above example will return a purged return code if the TGATWR was pending or active. A return code of other than purged will be returned if the TGATWR has already completed.

Besides the TCLEAR macro, TAMII provides the application programmer with one other control macro instruction -- TCNTRL. This macro is used to perform both device dependent and device independent control functions. The two independent functions supported are INHIBIT and RESTART.

The issuance of a TCNTRL with TYPE=INHIBIT causes the pending request queue to be placed in a hold state. At the completion of the currently active request, no pending request will be started until the hold state is reset by a TCNTRL with TYPE=RESTART, or until the device is disconnected.

A TCNTRL with TYPE=RESTART resets a hold state and restarts any pending requests. The hold state could have been set by a TCNTRL with TYPE=INHIBIT or by receipt of an attention from the device. Upon receiving an attention indication from a device, TAMII automatically puts the pending queue in a hold state and it is up to the attention handling routine to issue TCNTRL with TYPE=RESTART to reset the hold state. The following example shows both a coded RESTART and an INHIBIT:

```
STOP    DS  0H
        TCNTRL  TYPE=INHIBIT
          .
          .
          .
GO      DS  0H
        TCNTRL  TYPE=RESTART
```

Asynchronous Processing Using CHCKT and DECB

TAMII supports the use of DECBs on all I/O request macros except the TGATRD. To test DECBs for completion, the TAMII CHCKT macro is used. This macro allows more flexible application programming because of the programming objectives normally associated with interactive user support.

The application programmer should be careful using DECBs with TAMII to avoid unnecessary overhead. Any output request which has a DECB associated requires approximately twice the system overhead to process, than one that does not have a DECB. This is because the DECB associated request after completion has to be posted back to the task by TAMII so that the DECB can be marked complete and the completion status filled in; the request that does not have a DECB is released upon completion. The application programmer should only use DECBs when knowledge of the completion of an operation is required.

To use DECBs with TAMII is simple. The application programmer may request TAMII to build a DECB and associate the DECB with a macro at assembly time, or during execution can allocate and use as a DECB 48 bytes of storage. (Such storage must start on a fullword boundary.) The example below shows a TGATWR with an assembly-time generated DECB. When this method is used the DECB is always generated at the end of the generated parameter list:

```
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,DECB=(DECB1,Y)
```

The 'Y' as the second operand of the DECB operand signifies to the macro expansion that a DECB area called DECB1 is to be generated at the end of the macro expansion for the TGATWR macro. This same DECB area may be used by subsequent TAMII macros as long as the DECB is not currently in use. The following example shows this; the CHCKT in between the macros is required; otherwise, when the second TGATWR was executed, the request would not be accepted because the DECB would still be marked in use with the first TGATWR:

```
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,DECB=(DECB1,Y)
             .
             .
             .
         CHCKT    DECB=DECB1
             .
             .
             .
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,DECB=DECB1
             .
             .
             .
```

Once a DECB has been associated with a TAMII macro either a CHCKT or a TCLEAR macro must have been executed for the DECB before it can be reused with a new TAMII macro. If this rule is not followed and a TAMII macro is executed with an in use DECB, TAMII will not accept the request and will return a user error return code. Also, if TAMII returns a non-zero return code on the execution of a macro with a DECB associated the CHCKT must <u>not</u> be executed. The DECB is not active, because the TAMII request was not accepted.

Unlike the normal CHECK macro instruction with other access methods, the application programmer can control whether TAMII waits for completion to be marked in the DECB or returns with a return code signifying that the request is still active. This is accomplished by the application programmer using the WAIT option on the CHCKT macro. If WAIT=N, TAMII will test the DECB and return to the caller with a return code signifying whether the request has completed or is still active. If WAIT=Y, TAMII will not return until the request has been posted as complete, with or without errors. The following example shows two coded CHCKT macros, one with WAIT=N and the other with WAIT=Y:

```
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,DECB=DECBB
             .
             .
             .
         CHCKT  DECB=DECBB,WAIT=N         test for completion
         CH   R15,=H'4'                   request completed?
         BE   NOTDONE                     no, go test for other work
         TGATWR   OUTADDR=MSG1,OUTLGH=LMSG1,DECB=DECBB
             .
             .
             .
         CHCKT  DECB=DECBB,WAIT=Y         wait for completion
```

Attention Handling

Attention handling with TAMII is relatively complex because of the input and output buffering capability. With output buffering, TAMII, upon receiving a user attention request, sets a software interlock to prevent any pending writes from being initiated. This software interlock prevents all pending requests from being started except a TCLEAR, TFREE, or a TAMII request which has the BRK=Y option set.

TAMII contains an implicit operand called ATTNMODE that determines who does the reset and restart following an attention. If ATTNMODE=OLD, task monitor (CZCJT) upon receipt of the attention, purges the pending queues and then resets the TAMII software interlock. If ATTNMODE=NEW, it is left to the attention handler to reset the interlock and to do any other processing that may be necessary.

For normal attentions, the sequence for the attention handling routine to follow would be a TCLEAR to purge all pending input and output, followed by a TCNTRL to reset the interlock. The following example shows this process followed by a prompt for input:

```
         TCLEAR  TYPE=A          purge all pending input and output
            .
            .
            .
         TCNTRL  TYPE=RESTART     reset software interlock
            .
            .
            .
         TGTWSR  OUTADDR=MSG1,OUTLGH=LMSG1,INADDR=AREA,INLGH=LAREA
            .
            .
            .
LMSG1    DC   A(L'MSG1)
LAREA    DC   A(L'AREA)
MSG1     DC   C'ENTER REQUEST'
AREA     DC   XL256'00'
```

By using the ATTNSAV macro it is possible to save a current set of pending input and output, and with ATTNRST to restore it at a later time. For example, an application program, upon an attention, may issue a ATTNSAV to save all pending requests, both input and output. After the ATTENSAV, the application program would issue a TCNTRL with TYPE=RESTART and then prompt the user for instructions. At a later time, on command from the user for instance, the application program could issue an ATTNRST to restore the pending queues and the user would be back to the state existing before the attention occurred. The following example shows the sequence of macro instructions needed to accomplish this result:

```
         ATTNSAV
         STM  R0,R1,SAVLEV                save attention level numbers
            .
            .
            .
         TCNTRL  TYPE=RESTART              reset software interlock
            .
            .
            .
         TGTWSR  ...                       request action from user
            .
            .
            .
         LM  R2,R3,SAVLEV                  get saved level numbers
         ATTNRST  LEVLIN=(R3),LEVOUT=(R2)  restore attention
            .
            .
            .
SAVLEV DC   2F'0'                          save area for level
```

In the above example any of the TCLEAR macro instruction types could have been executed to purge requests that the application program may not have wanted saved. For example, a TCLEAR with TYPE=O could have been executed to purge any pending output; then the ATTNSAV would only have saved the pending input records. Also, in the above example, an ATTNDST could have been executed (instead of the ATTNRST) to delete any saved attention levels.

INDEX

When more than one page reference is given, the major reference is first. All references are within at least plus or minus one of the indicated page number.