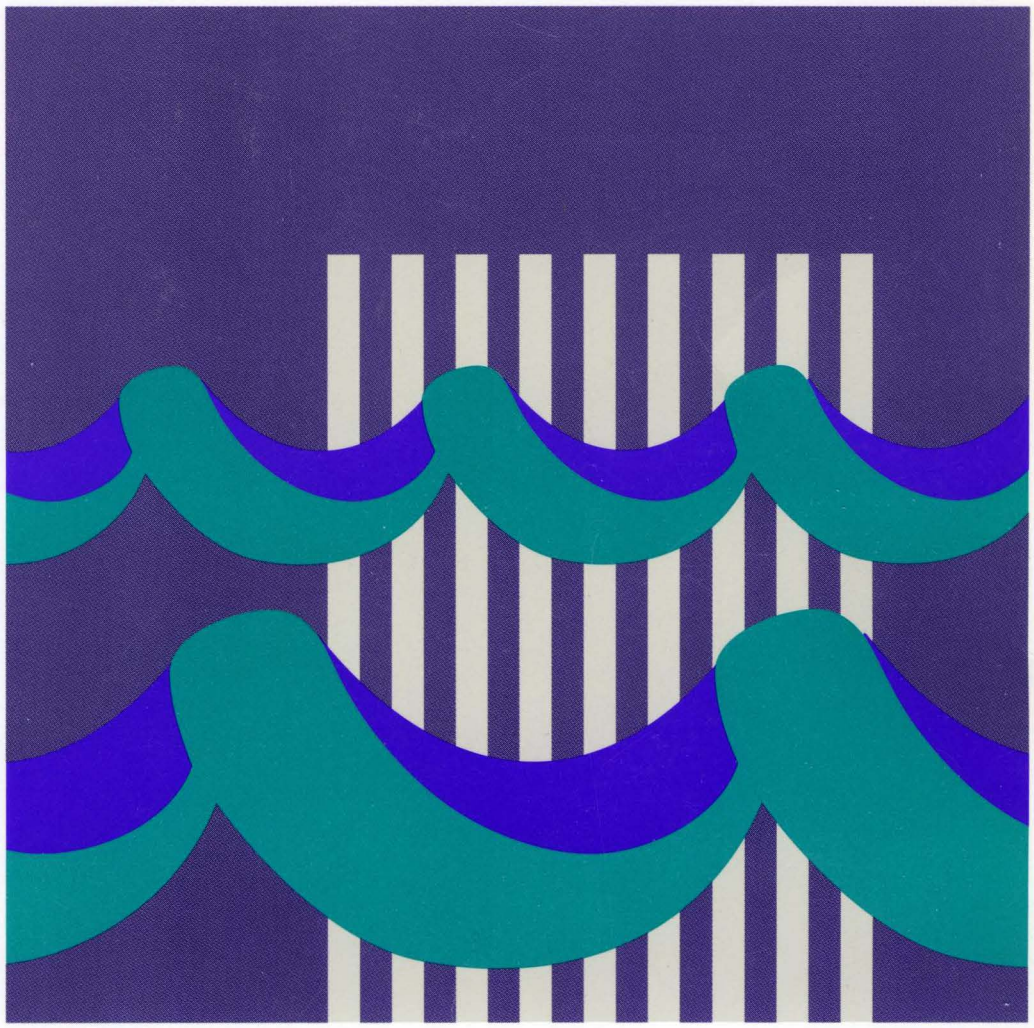
LY43-0032-01

Network Control Program
System Support Programs

Customization Reference

NCP Version 7 Release 2
SSP Version 4 Release 2



LY43-0032-01

Network Control Program
System Support Programs

Customization Reference

NCP Version 7 Release 2
SSP Version 4 Release 2

Note

Before using this document, read the general information under "Notices" on page xi.

Second Edition (October 1994)

This major revision replaces LY43-0032-00. This licensed document applies to the following IBM licensed programs:

- Advanced Communications Function for Network Control Program Version 7 (program number 5648-063).
- Advanced Communications Function for System Support Programs Version 4 for MVS (program number 5655-041), for VM (program number 5654-009) Release 1, and for VSE (program number 5686-064) Release 1.

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Department E15
P.O. Box 12195
Research Triangle Park, North Carolina 27709
U.S.A.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Tables

Notices

Any reference to an IBM licensed program in this licensed document does not imply that IBM intends to make it available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
International Business Machines Corporation
500 Columbus Avenue
Thornwood, New York 10594
United States of America

The licensed programs described in this document and all licensed material available for them are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

Programming Interface Information

This book is intended to help the customer customize Advanced Communications Function for Network Control Program (NCP) and System Support Programs (SSP). This book primarily documents Product-Sensitive Programming Interface and Associated Guidance Information provided by NCP.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

However, this book also documents General-Use Programming Interface and Associated Guidance Information.

General-Use programming interfaces allow the customer to write programs that obtain the services of SSP.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

```
┌──────────── General-Use Programming Interface ────────────┐
```

General-Use Programming Interface and Associated Guidance Information...

```
└──────────── End of General-Use Programming Interface ─────┘
```

Trademarks

The following terms, denoted by an asterisk (*) at their first occurrence in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

BookManager	MVS/XA	VTAM
IBM	NetView	VSE/ESA
Library Reader	System/370	
MVS/ESA	VM/ESA	

About This Book

This book provides information to help you customize Advanced Communications Function for Network Control Program (NCP) and System Support Programs (SSP).

Customizing NCP means modifying it to enhance support for certain stations or to provide support for stations that are not currently supported by the IBM-supplied programs. Customization can also include writing programmed Systems Network Architecture (SNA) resources that reside in the communication controller.

Customizing SSP means modifying it to control the generation process for both NCP and Emulation Program (EP). You can create generation applications that process the NCP or EP generation definition as part of the SSP generation process. This enables you to define network resources not provided by the IBM-supplied definition functions.

NCP and SSP Customization Reference is intended to supplement *NCP and SSP Customization Guide* and provides a quick reference to IBM-supplied NCP customization macros and NDF internal utilities for user-written generation applications.

Who Should Use This Book

This book is for programmers who want to modify or enhance an NCP or the generation process for NCP and EP. You must have a detailed knowledge of the function and operation of NCP. You must also be familiar with the basic concepts of data communication and SNA.

How to Use This Book

Before you begin to customize an NCP you should read *NCP and SSP Customization Guide* to learn how to use the NCP customization macros and how to write and use applications. You may need to refer to *IBM 3704/3705 Assembler Language*, which describes the assembler language used with all IBM communication controllers. You may also need to refer to *Principles of Operation* for your controller.

Read Part 1 to learn about the NCP customization macros and the attachments for user-written line control code in NCP code. This part contains supervisor call (SVC) codes and functional groupings of macros, an alphabetical listing of the NCP macros, and a description of entrances and exits for user-written line control.

Read Part 2 to learn about the NCP/EP definition facility (NDF) internal utilities you can use in your generation applications. You can use these utilities to manipulate strings, pass link-editing statements, and create error messages. These utilities are listed in alphabetical order.

Terms Used in This Book

The following descriptions explain how terms are used in the NCP, SSP, and EP library.

"MVS," "VM," and "VSE"

The term *MVS* means the MVS/XA* and MVS/ESA* systems. The term *VM* means the VM/ESA* systems in the CMS environment. The term *VSE* means the VSE/SP, VSE/ESA*, and VSE/Advanced Function operating systems. If information is applicable to only one system, the specific system name is used.

"Port" and "Channel" with LPDA

In discussions concerning link problem determination aid (LPDA) for multiport and data-multiplex mode (DMPX) modems, the terms *port* and *channel* are synonymous. Although *port* is the more commonly used term, *channel* can be used in sections describing LPDA.

"IBM Special Products or User-Written Code"

This book sometimes refers to *IBM special products or user-written code.* This phrase means IBM* special products such as Network Terminal Option (NTO), Network Routing Facility (NRF), and X.25 NCP Packet Switching Interface (NPSI), or user-written code.

IBM 3745 Communication Controller Model Numbers

In this book, the term *IBM 3745 Communication Controller* refers to all IBM 3745 models. When particular models are discussed, the appropriate model numbers are specified. Model numbers include IBM 3745-130, 3745-150, 3745-160, 3745-170, 3745-17A, 3745-210, 3745-21A, 3745-310, 3745-31A, 3745-410, 3745-41A, 3745-610, and 3745-61A.

"NCP V7R2"

In this book, unless otherwise specified, the term *NCP V7R2* refers to NCP Version 7 Release 2 with or without the optional NCP feature for 3746 Model 900 connectivity subsystem support. To use this feature, you must have the 3746 Model 900 installed in your controller.

How Numbers Are Written

This book shows numbers over 9999 in metric style, which means that a space is used instead of a comma to separate groups of three digits. For example, the number ten thousand five hundred fifty-two is written 10 552. However, keyword values, for example, SALIMIT=65535, do not use a blank.

What Is New in This Book

This edition contains no new editorial or technical changes.

Where to Find More Information

A good place to start any task regarding NCP, SSP, or EP is *NCP V7R2, SSP V4R2, and EP R12 Library Directory*. This directory introduces the enhancements for the current release and shows where these enhancements are described in the NCP library. It gives you an overview of NCP, SSP, and EP and directs you to information on a variety of tasks related to these programs. When you are using the book online, you can use *hypertext links*[1] to move directly from task and enhancement descriptions to the appropriate chapters of other books in the library.

Information for NCP Tasks

The books in the NCP, SSP, and EP library are listed below according to task, along with closely related books and tools you may find helpful. See "Bibliography" on page 547 for a brief summary of each book in the NCP, SSP, and EP library and listings of related publications.

Table 1 (Page 1 of 2). Sources of Information by Task

Order No.	Title	Hardcopy	Softcopy
Planning			
SC31-7122	*Planning for NetView, NCP, and VTAM*	■	■
SC31-7123	*Planning for Integrated Networks*	■	■
SX75-0092	*Planning Aids: Pre-Installation Planning Checklist for NetView, NCP, and VTAM*	■	
SC31-6259	*NCP V7R2, SSP V4R2, and EP R12 Library Directory*	■	■
Installation and Resource Definition			
SC31-6221	*NCP, SSP, and EP Generation and Loading Guide*	■	■
SC31-6258	*NCP V7R2 Migration Guide*	■	■
SC31-6223	*NCP, SSP, and EP Resource Definition Guide*	■	■
SC31-6224	*NCP, SSP, and EP Resource Definition Reference*	■	■
Customization			
LY43-0031	*NCP and SSP Customization Guide*	■	
LY43-0032	*NCP and SSP Customization Reference*	■	
Operation			
SC31-6222	*NCP, SSP, and EP Messages and Codes*	■	■
N/A	*Online Message Facility*		D

D Available on diskette for the IBM OS/2 environment.

[1] A *hypertext link* is a pointer from a location in an online book to another location in the same book or another book. By selecting highlighted information, such as a message number, you can move quickly to related information and, if desired, back again.

Table 1 (Page 2 of 2). Sources of Information by Task

Order No.	Title	Hardcopy	Softcopy
Diagnosis			
LY43-0033	*NCP, SSP, and EP Diagnosis Guide*	■	
LY43-0037	*NCP, SSP, and EP Trace Analysis Handbook*	■	
LY43-0029	*NCP and EP Reference*	■	
LY43-0030	*NCP and EP Reference Summary and Data Areas*	■	
LK2T-1999	*NCP, SSP, and EP Diagnosis Aid*		**D**
Monitoring and Tuning			
SC31-6247	*NTune User's Guide*	■	■
LY43-0035	*NTuneNCP Reference*	■	

D Available on diskette for the IBM OS/2 environment.

Those publications available as softcopy books have cross-document search and hypertext links for speedy, online information retrieval. These softcopy books are grouped together on an electronic bookshelf and are part of the *IBM Networking Systems Softcopy Collection Kit* on compact disc read-only memory (CD-ROM).

You can view and search softcopy books by using BookManager* READ products or by using the IBM Library Reader* product included on CD-ROM. For more information on CD-ROMs and softcopy books, see *IBM Online Libraries: Softcopy Collection Kit User's Guide* and BookManager READ documentation.

Part 1. NCP Customization

Chapter 1. NCP Macro Overview

This chapter lists supervisor call (SVC) codes for Network Control Program (NCP) and groups these macros by function.

Before you begin to customize NCP you should read *NCP and SSP Customization Guide* to learn how to write customized NCP routines and use the NCP customization macros. You may need to refer to *3704/3705 Communications Controller Assembler Language*, which describes the assembler language used with all IBM communication controllers. Controller assembler is a modified version of System/370* Assembler. You may also need to refer to *Principles of Operation* for your controller.

Note: The following macros might be in the IBM-supplied NCP code, but are not supported for use in the NCP customization process:

EXECBHR	SAPTR	SUPVNUC@
GPTINV	SEQJMP	XIO CHANANS
POST	SETXIT	XIO CHANNEL

SVC Codes for NCP Macros

An SVC code is generated whenever level 5 uses a supervisor macro. Table 2 shows these codes and their associated macros.

Table 2 (Page 1 of 2). SVC Codes for Supervisor Macros

Code	Macro	Code	Macro
SVC01	LEASE	SVC36	ABORT,CC≠0,SYSOPT
SVC02	RELEASE	SVC37	RSLVNAD
SVC03	CHAIN	SVC38	XIO SDLC Link
SVC04	UNCHAIN	SVC39	XPORTVR
SVC05	SCAN	SVC40	COPYPIU (LEASE=NO)
SVC06	POINT	SVC41	(Reserved)
SVC07	DEQUE	SVC42	COPYPIU (LEASE=YES)
SVC08	ENQUE	SVC43	RSLVSSCP
SVC09	ADVAN	SVC44	XIO (CHANANS)
SVC10	INSERT	SVC45	XIO (Channel,CAB=)
SVC11	EXTRACT	SVC47	NCHNG
SVC12	RETURN	SVC48	COMMIT
SVC13	CHAP	SVC49	DECOMMIT
SVC14	TRIGGER	SVC60	PEPSWH
SVC15	QPOST	SVC61	BFREVENT
SVC16	CALL	SVC62	ABORTVR
SVC17	XIO (BSC/SS line)	SVC64	PRELEASE
SVC18	XIO (channel)	SVC65	(Reserved)
SVC19	RSLVSNP SVC66	SVC66	ATTACHVR
SVC20	XIO (set mode)	SVC67	DETACHVR
SVC21	XIO (immediate)	SVC68	RSLVNET
SVC22	SETIME	SVC70	ALLOCATE
SVC23	TPPOST	SVC72	ACTVRIT
SVC24	RSLVRID	SVC73	DACTVRIT
SVS25	COPYBCU	SVC74	LINKTGB
SVC26	SYSXIT	SVC75	RSLVTGB
SVC27	FLIPPS	SVC76	SETTGB

Table 2 (Page 2 of 2). SVC Codes for Supervisor Macros

Code	Macro	Code	Macro
SVC28	(Reserved)	SVC77	XIO(TG)
SVC29	GETBYTE	SVC78	ROUTE
SVC30	PUTBYTE	SVC79	TRACEPIU
SVC31	GETIME	SVC81	NVRID
SVC32	EXECBHR	SVC82	VRACT
SVC33	ABORT,CC=0	SVC83	UACTRTN
SVC34	ABORT,CC≠0,RELEASE	SVC83	URETURN
SVC35	ABORT,CC≠0,PASS	SVC84	FETRACE

The SVC code is contained in the linkage between levels 5 and 4. The linkage, generated by the RNSVC macro, consists of an EXIT instruction, SVC code, flags, communication bits, parameters, and space for output variables. Table 3 shows the linkage generated by RNSVC.

Table 3. Layout of the Linkage Generated by RNSVC

Generated Instructions	RNSVC Variables	Definition		
		Byte	Bit	Value
EXIT		0 to 1		EXIT instruction
	code	2	0 to 6	SVC code
DC AL1 ( )	oneprm		7	One-parm flag
	b0	3	0	Three-parms flag
	b1		1	Communication bit
	parm1		2 to 4	Parameter 1 code
DC AL1 ( )	parm2		5 to 7	Parameter 2 code
	ipvars	4		Unused (alignment only)
		5	0	Pseudo parm 3 flag
			1	Communication bit
			2 to 4	Parameter 3 code
DC AL4 ( )			5 to 7	Communication bits
DC AL4 ( )	xtrvars			Inline parameter 3
LA VAR,0	opvars			Output variables

Bytes 2 and 3 are always generated. If the one-parm flag (byte 2, bit 7) is on, byte 3 contains one input parameter indicated in bits 2 to 4. Therefore, bits 0, 1, 5, 6, and 7 in byte 3 are available for use as communication bits. In this case, only bytes 2 and 3 are generated.

If the one-parm flag is off, only bit 1 of byte 3 is available for use as a communication bit. Bits 2 to 7 are used as input parameters. In this case, if more than one communication bit is required, the three-parms flag (byte 3, bit 0) is on, bytes 4 and 5 are generated, and all communication bits are in byte 5. If the pseudo parm 3 flag is on (byte 5, bit 0), that byte does not contain a parameter 3 code and exists solely for the communication bits. Inline parameters are 4-byte address constants generated if parameters are in label notation rather than register notation.

The values for parameter 1 codes (byte 3, bits 2 to 4) are as follows.

Code Value	Location of Input Parameter
0	Register 2
1	Register 3
2	Register 4
3	Register 5
4	Register 6
5	Register 7
6	Inline parameter
7	Active queue control block (QCB)

The values for parameter 2 and 3 codes (byte 3, bits 5 to 7 or byte 5, bits 2 to 4) are the same as the parameter 1 codes, except that bit 7 is unused.

When the EXIT instruction is executed, level 4 receives control and sets up the level 4 linkage registers as follows.

Register	Content
0	Address register
1(0)	Level 5 register 1, byte 0
1(1)	Communication bits (byte 3 or 5)
2	Parameter 1
3	Parameter 2
4	Parameter 3
5	Service routine work register
6	Level 4 save area
7	Level 5 instruction address register (updated past communication bits)

The SVC code is used as an index into the branch table of supervisor nucleus routines to pass control to the appropriate routine.

NCP Macros Grouped by Function

This section groups the NCP macros by function. The groups are in alphabetical order. Each macro appears in the functional group in which it is most commonly used; however, many macros can be used in other groups as well.

Binary Tree Macros: Allow you to work with binary trees, mostly used in routing functions.

BTCHECK BTINSERT BTSEARCH
BTDELETE

Block Handler Macros: Provide access to the block handling routines (BHRs). The block handling routines are supplied by NCP or are user-written. Block handling routines provide message processing functions within the communication controller.

ABORT GETPARM GRPEND
BHEXIT GETPT GRPENTRY
BLKENTRY

BSC and Start-Stop Support Macros: Are used exclusively for binary synchronous communications (BSC) and start-stop line control routines:

BLDR	INCRP	SETRP1C
COPYBCU	PUTBYTE	TPPOST
GETBYTE		

Buffer Management Macros: Reserve buffers from and return buffers to the buffer pool. Buffers are the working storage of NCP. You can also use these macros to get information contained in the buffers, or to chain the buffers together and manipulate buffer chains.

CHAIN	PRELEASE	TAGBUFF
CHECKSSI	RELEASE	UNCHAIN
LEASE	SCAN	

Channel Macro: Communicates across the channel from NCP to the host processor. To use this macro, you must have detailed knowledge of the channel adapters and NCP configuration.

CAIO

Flow Control Macros: Control the pacing and flow of path information units (PIUs) along virtual routes.

ALLOCATE	DECOMMIT	VRIMTASK
COMMIT	VREVENT	

Interrupt Macros: INHIBIT and RESET are used in program levels 2 to 4 to enable or disable certain interrupts. The interrupts are enabled or disabled by placing a mask in a specified register. Use PCIL4 in program levels 1 to 4 to request level 4 to carry out certain types of program control interrupts.

INHIBIT	PCIL4	RESET

Level 5 Input/Output Macros: May be used by program level 5 routines to send request and response PIUs from the current NCP resource (origin) to any other resource in the network (destination). These macros are also used by program level 5 to request I/O code to carry out specific functions. To use these macros, you must have a knowledge of the network addresses, virtual routes, and the queue control block (QCB) associated with the task that you want to perform.

NEOENQ	XIO IMMED	XIO LINK
NEOXPORT	XIO LINE	XIO SETMODE
NPAPIU		

Linked List Management Macros: Manage a form of linked list called an ACHAIN.

ACHAIN	ASCAN	AUNCHAIN
AFIND		

Miscellaneous Macros: Improve the quality and productivity of programming by replacing frequently used coding sequences with macros.

Branch macros:

BAL	BNC	BNZ
BCR	BNDH	BP
BH	BNE	BPZ
BM	BNH	BZR
BMZ	BNL	

Other miscellaneous macros:

COMPARE	LA	SWAP
DEFMSK	SETXTRN	

NCP Information Macros: Obtain information about the network and the subarea containing this NCP. These macros also provide information about NCP itself and the resources defined to NCP.

BUFCHK	PACEMAP	RSLVRID
DEVPARMS	RCBSCAN	RSLVSNP
GETCB	ROUTEMAP	RSLVSSCP
GETIME	RSLVCAP	RSLVTGB
NPAQINFO	RSLVDYN	RSLVVVTI
NPAQSTAT	RSLVNAD	UPARMS
NPARMS	RSLVNET	VRACTCK

PIU Management Macros: Copy, alter, or get information from a PIU.

BUILDPIU	ECBINIT	MOVECHAR
COPYPIU	EXCR	PIUEND
ECB	MOVE	

Problem Determination Macros: Aid in problem determination for NCP.

ABEND	GALERT	SAVESQ
DTRACE	MAINT	TRACEPIU
FETRACE	MAINTCS	

Queue Management Macros: Add elements to any place on the queue, move elements from one queue to another, set the priority of elements within a queue, check validity on the queue control block (QCB), and purge the QCB.

ADVAN	INSERT	PURGQCB
DEQUE	MVQUE	SETPRI
ENQUE	POINT	VALQCB
EXTRACT		

Save Area Management Macros: Create and use save areas. Save areas retain register contents during program execution.

CXTSVX	RESTORE	SAVEAREA
LDM	SAVE	STRM

Scanner Macros: Communicate across the scanners in the communication controller from NCP to a terminal or other link-attached station. These macros also start input and output operations from background code. You must know how scanners operate and you must understand NCP configuration to use these macros.

SETLATO	IOHM	OUTICW1

Shift Macros: Shift the bit positions of a specified register to the left or right.

ASHIFT	SLLH	SRLB
SLL	SRL	SRLH
SLLB		

Structuring Macros: Enable structured programming in IBM controller assembler language. These are the most frequently used macros in NCP programming. The functions available with these macros are similar to those of other high-level languages.

ANDIF	DOUNTIL	ENDIF
CASE	DOWHILE	IF
CASEIF	ELSE	LEAVEDO
CASENTRY	ENDCASE	ORIF
CASEXIT	ENDDO	THEN

Subroutine Macros: Transfer control within NCP from one routine to another.

CALL	PERFORM	ROUTINE
LINK	RETURN	SUBRTN

Supervisor Macros: Are used by NCP supervisor services to establish linkage between the background routines and the supervisor service routines.

RNSVC	SVLINK

Task Control Macros: Control task execution within NCP regarding priorities and other task states.

BFREVENT	QPOST	SYSXIT
CHAP	SETEVNTL	TRIGGER

Timer Macros: Set, refresh, or refer to NCP timer values. If you are creating your own line-control routines, you must also create your own timer routines. These are the macros that create those routines.

SETIME	TVSNEW	TVSRTRN
TVSIDL	TVSRAS	TVSTIME
TVSMOD	TVSREF	

Transmission Group Macros: Control transmission group protocol. These macros activate or deactivate a transmission group. They also control data flow and check the specific states and conditions of a transmission group.

LINKTGB	TESTTGB	XIOFL
ROUTE	XIO TG	XPC

User Line Control and Programmed Resource Macros: Allow you to carry out your own line control in place of the line control supplied by IBM within NCP.

FINDUACB	NCHNG	SDB
FVTABLE	NEOAXT	UACTRTN
LASTUACB	POSTUACB	URETURN

Virtual Route Macros: Control virtual route protocols. They also attach and detach NCP resources from internal NCP virtual routes. They function the same way in networks that are attached to NCP through the SNA network interconnection process.

ABORTVR	CONVRT	PIUDEALL
ACTVRIT	DACTVRIT	VRACT
ATTACHVR	DETACHVR	XPORTVR
CHECKVR	NVRID	

Chapter 2. Macro Directory

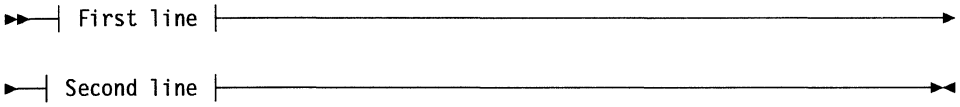
This chapter describes all of the NCP macros available for customizing NCP. For easy reference, these macros are arranged alphabetically.

Keyword Syntax Diagrams

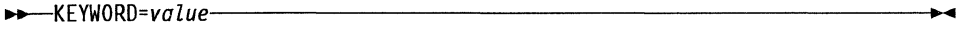
This book uses standard IBM syntax diagrams to describe the syntax of the NCP macros. These diagrams use the following conventions:

- Read the diagrams from left to right. In general, any path between the start symbol (►►) and the end symbol (►◄) represents valid coding for a keyword.

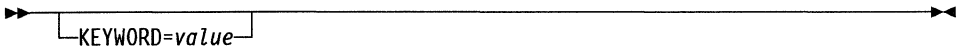
- A long diagram may be broken into two or more lines.



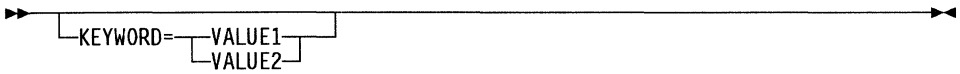
- A keyword that appears on the main path is required. You must code all required keywords.



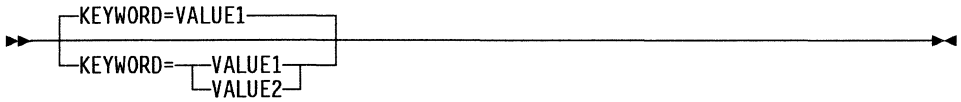
- A keyword that appears below the main path is optional. You do not need to code optional keywords. Most NCP keywords are optional.



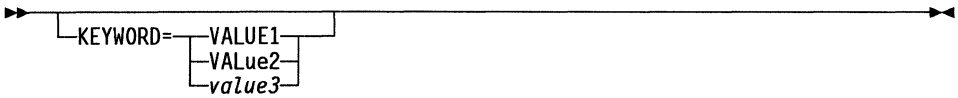
- When you can choose from more than one keyword value, those values are stacked vertically below the main path.



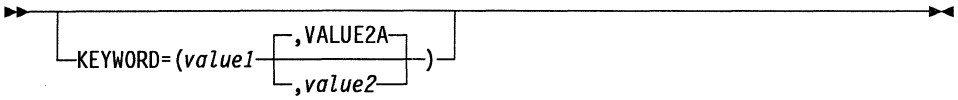
- If a keyword has a default value, the keyword and value appear above the main path.



- Uppercase characters show values you code exactly as shown (VALUE1 below). Uppercase characters in a mixed-case string indicate that you can code an abbreviation (VALue2 below means you can code VAL or VALUE2). Lowercase italics show variables for which you need to supply a value, such as a number or string (*value3* below). Do not code a space or comma between the digits of a numeric value.

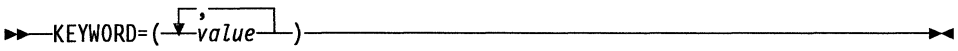


- Multiple keyword values in parentheses are called *suboperands*. Required suboperands appear on the main path between the parentheses (*value1* below). Optional suboperands appear below the main path between the parentheses (*value2* below). Default values for optional suboperands appear above the main path between the parentheses (VALUE2A below). All suboperands must be separated by commas.

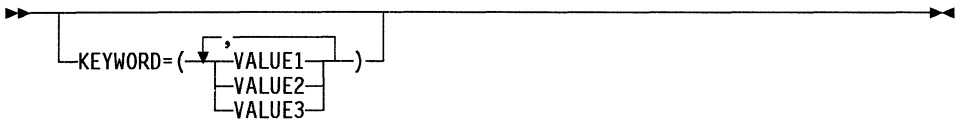


  Note: If a default is not shown for an optional suboperand, this means that either there is no default or that the default depends on what you code on other keywords or suboperands. You have to read the keyword description to find out about this.

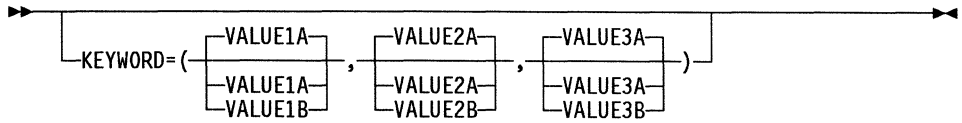
- An arrow returning to the left above a suboperand indicates that you can code multiple values, enclosed in parentheses and separated by commas.



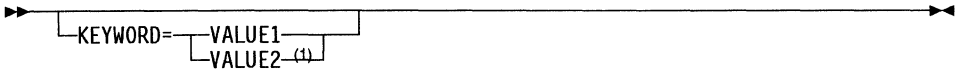
- An arrow returning to the left above a stacked list of suboperands means that you can code as many of the suboperands as you need and in any order. Do not code the same suboperand more than once. Separate each suboperand you code with a comma. For example you could code KEYWORD=(VALUE3,VALUE1) for the keyword below.



- If you omit an optional suboperand, code a comma to indicate its position. For example you could code KEYWORD=(VALUE1B,,VALUE3B) for the keyword below to omit the second suboperand. Commas are not required if you omit optional suboperands at the end of a suboperand list. For example you could code KEYWORD=(VALUE1B,VALUE2A) or KEYWORD=(VALUE1B,VALUE2A,) to omit the last suboperand for the keyword below; the result would be the same. If you code only the first suboperand, parentheses are not required. For example you could code KEYWORD=VALUE1B or KEYWORD=(VALUE1B) for the keyword below; the result would be the same. If you omit all suboperands or do not code the keyword, the defaults are used. For example you could code KEYWORD=(), KEYWORD=(,,), or not code the keyword at all for the keyword below; the result would be the same.

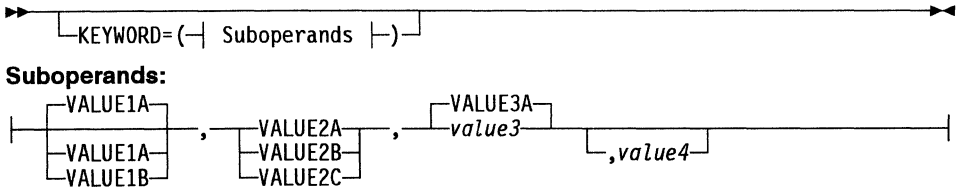


- Restrictions on the use of a keyword value are indicated by superscript numbers in parentheses and are explained below the diagram. Do not code the superscript number or parentheses.



[1] Restriction on VALUE2.

- A section of a long diagram may appear separately below the main diagram.



Description Conventions

The following conventions are used to describe certain keyword values:

- An ellipsis (...) shows that you can code a sequence of values within parentheses.
- Quotation marks must be used to frame a character string if it can be confused with a keyword value for a keyword. Using quotation marks allows you to use certain names as symbols.
- Within syntax diagrams, keywords that are always required appear first, followed by keywords that are conditional or optional.
- Data set (file) names must begin with an alphabetical character or *$*, *@*, or *#*.

Some of the format descriptions of the macros make a distinction between *absolute* and *label* (or *equated*) notation. When a format description does not include either of these terms, both forms are valid. These two terms are defined as follows:

Absolute A notation expressed in terms which can be immediately evaluated.

Label A notation expressed as a label which is resolved as the value defined by an assembler statement label. The label is also known as an *equated notation* and may refer to a constant, a memory address, or a register.

Table 4 shows examples of notations expressed in absolute terms and as labels.

Table 4 (Page 1 of 2). Absolute and Label Terminology in Format Description Notations

Notation	Example
Absolute decimal	`LEASE  COUNT=1,...`
Absolute hexadecimal	`LEASE  COUNT=X'1',...`
Absolute register[1]	`LEASE  ECB=2,...`
Absolute byte register[1]	`LEASE  COUNT=1(1),...`
Label decimal	`ONE  EQU  1` `LEASE  COUNT=ONE,...`

Table 4 (Page 2 of 2). Absolute and Label Terminology in Format Description Notations

Notation	Example
Label hexadecimal	`ONEH  EQU  X'1'` `LEASE  COUNT=ONEH,...`
Label register[1]	`REG2  EQUR  2` `LEASE  ECB=REG2,...`
Label byte register[1]	`REG1LO  EQUR  1(0)` `LEASE  COUNT=REG1LO,...`

1 Unless otherwise indicated, do not specify register 0.

Coding the SUPV Keyword

In some cases, the keyword format of a supervisor macro depends on the coding of the SUPV keyword. Where these differences occur, formats are supplied for the variation of the macro.

Note: If you are adding user routines to NCP, exercise caution in the use of SUPV. Incorrect usage will cause the system to abend.

Code SUPV=NO (or omit the keyword) when level 5 code uses task, queue, or buffer management macros. These macros generate an EXIT instruction followed by a DC instruction initialized with an SVC code. When the EXIT instruction is used in level 5, the level 5 instruction address register is updated to point to the SVC code. The EXIT from level 5 causes a level 4 interrupt.

Code SUPV=YES when routines at levels 1, 2, 3, and 4 use task, queue, or buffer management macros. The following macros generate inline code when SUPV=YES. (LEASE generates inline code when SUPV=CHARSERV.)

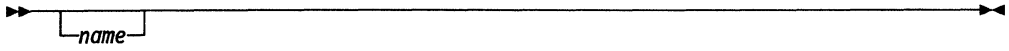
ACHAIN	CONVAT	SCAN
ADVAN	DEVPARMS	SETEVNTL
AFIND	ECBINIT	SETXTRN
ASCAN	LEASE	TAGBUFF
AUNCHAIN	MOVE	UNCHAIN
CHAIN	POINT	VRIMTASK
CHECKSSI	RCBSCAN	XIOFL
CHECKVR	RSLVVVTI	VALQCB

The remaining macros with SUPV=YES generate a branch to the service routine that does the requested function. The service routine, which executes at the same level as the branch instruction, saves and restores all registers that it uses.

Macro Names

You can assign a name to most macro definitions to label the macro. You are required to assign a name to some macros. The macro name is shown as the *name* parameter in the illustration of each macro for which it is valid. Unless otherwise indicated, the description of the macro name is as follows.

Parameters



Function Provides a label to be inserted on the first statement of the generated source.

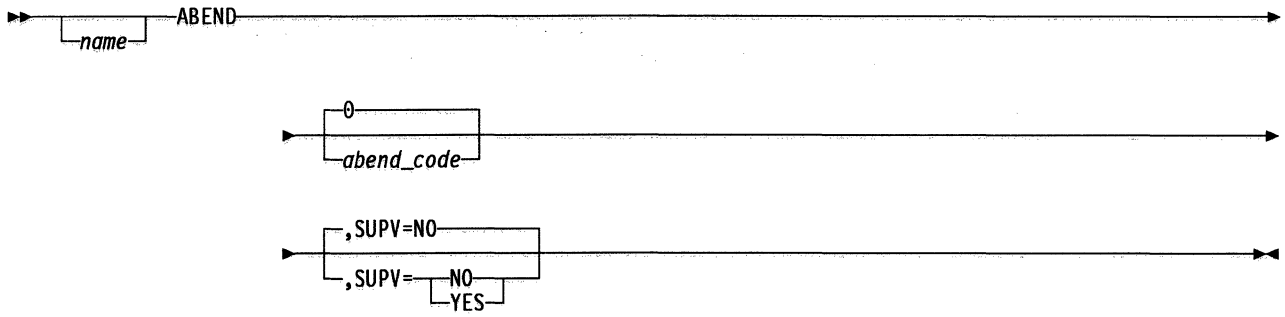
Format Up to 8 characters. The first character can be A to Z, $, #, or @. The remaining characters can be A to Z, 0 to 9, $, #, or @.

Default None.

ABEND—Terminate NCP Abnormally

The ABEND macro abnormally terminates NCP execution and posts a specified abend code at storage location XDH + '60'.

Syntax



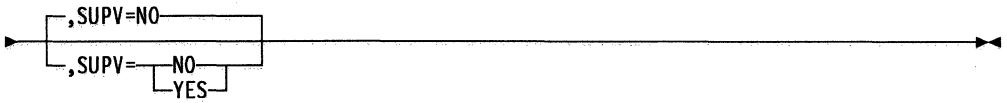
Parameters

Function Specifies the abend code to be posted by the supervisor, indicating the cause for stopping NCP execution.

Format Absolute or register notation.

Default 0.

Remarks The macro always loads the abend code into register 1. Register 0 is not allowed.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

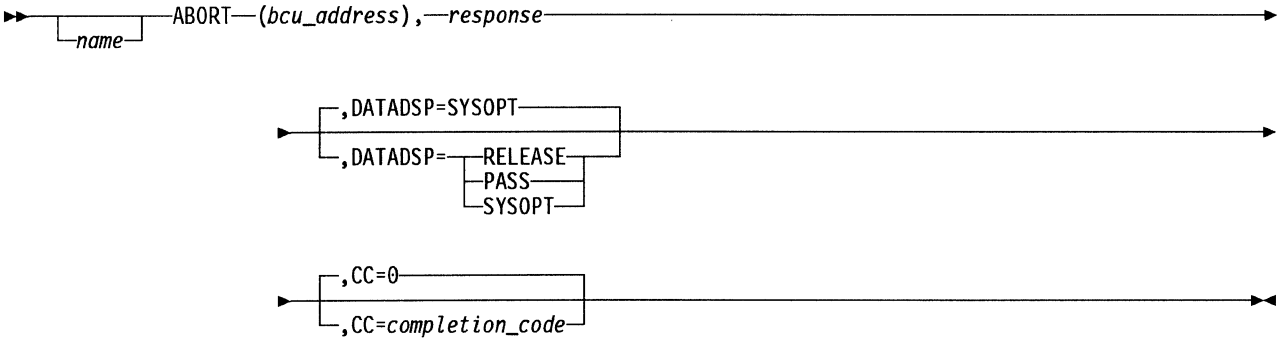
Default NO.

ABORT—Stop the Execution of a Block Handler

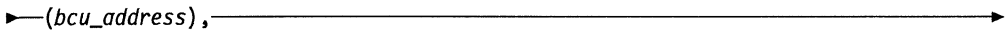
The ABORT macro is used by block handling routines (BHRs) to prematurely stop the execution of the current block handler (BH). The completion code determines how the block is to be processed.

Register 0 is not allowed for register parameters.

Syntax



Parameters

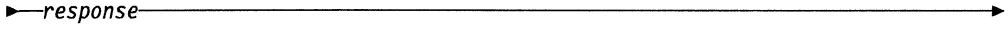


Function Specifies the register containing the address of the block control unit (BCU) that is being aborted.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. Be sure that the BCU that is specified does not reside on a queue.

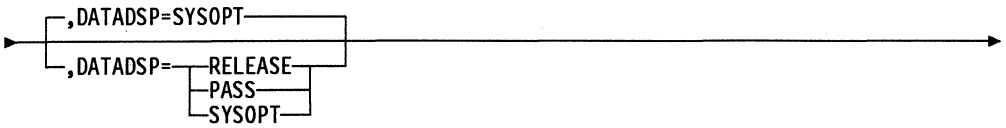


Function Specifies the bit configuration to be stored in the system response byte of the basic transmission unit (BTU).

Format Byte register notation or an absolute value from 0 to 255.

Default None.

Remarks Register 1(0) is standard.



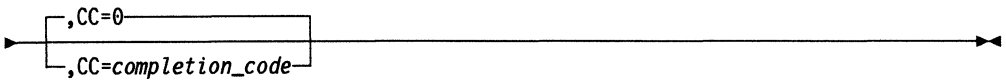
Function Specifies what happens to the buffers used for the BTU to be sent to the host.

Format RELEASE, PASS, or SYSOPT.

Default SYSOPT.

Remarks Specify RELEASE if the buffer chain associated with the BTU is to be released. Specify PASS if the buffer chain is not to be released.

Specify SYSOPT when the general communication byte (SYSFLG0) is to be examined to determine whether to release data.



Function Specifies the completion code, which determines the type of abort being performed.

Format Absolute value of 0 to 255.

Default 0.

Remarks A completion code of 0 indicates that only the current block handler execution is to be aborted and normal block processing is to continue. All other keywords are ignored.

A completion code other than 0 indicates that the BTU is to be sent to the host by a TPPOST macro. The BCU has been removed from the queue.

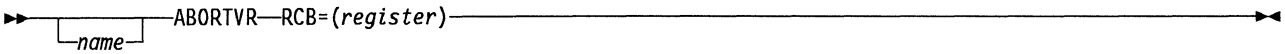
ABORTVR—Cancel a Virtual Route Activation Attempt

The ABORTVR macro cancels an ongoing NCP virtual route activation attempt for a resource if that process was started by the VRACT macro. The input is the pointer to the resource connection block (RCB) of the resource undergoing activation. When ABORTVR is issued, the RCB could be in one of three states:

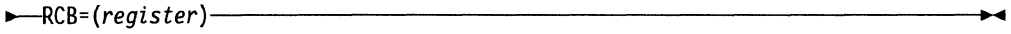
- If the RCB is on the ACTVR queue, it is removed from the queue.
- If the RCB is already attached to a VRB that is waiting for an ER.ACT reply or ACTVR response, the RCB is taken off the VRB.
- If the RCB is attached to an active VRB, the RCB is detached. In this case, the resource is triggered even though ABORTVR was issued.

Note that you can use this macro only in level 5.

Syntax



Parameters



Function Specifies the register containing the address of the RCB for the resource that is undergoing virtual route activation.

Format Register notation.

Default None.

Remarks Registers 0, 1, and 6 are not allowed.

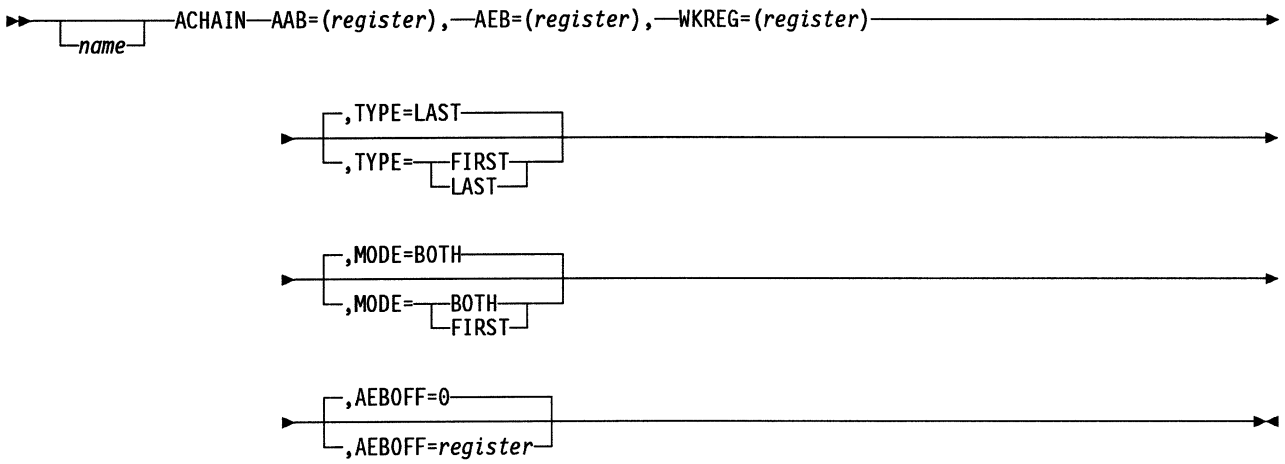
ACHAIN—Add an Element to an ACHAIN

The ACHAIN macro chains an element to either the head or the tail of an ACHAIN. An ACHAIN is a chain of elements to which an anchor block points. An anchor block may contain two pointers: one pointer points to the first element in the chain, and the other pointer points to the last element in the chain. However, the achain anchor block (AAB) may contain just a first pointer. Elements within an ACHAIN are usually control blocks, but they do not have to be control blocks.

See *NCP and EP Reference Summary and Data Areas*, Volume 1, for more information on the AAB and the achain element block (AEB).

Register 0 is not allowed for register parameters.

Syntax



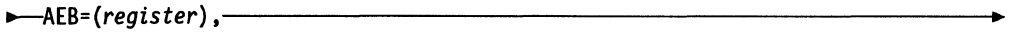
Parameters

Function Specifies the register that contains the address of the AAB to which the new element is being chained.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified foi AEB or WKREG.

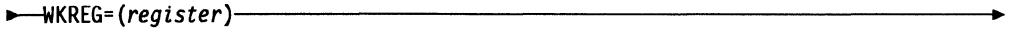


Function Specifies the register that contains the address of the AAB to be chained to the AAB.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AAB or WKREG.

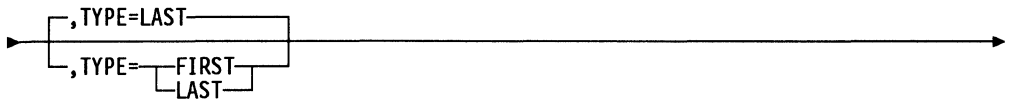


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AAB or AEB.

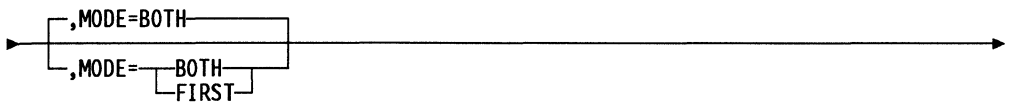


Function Specifies whether the element is to be chained to the head or tail of the ACHAIN.

Format FIRST or LAST.

Default LAST.

Remarks If MODE=FIRST, the default for TYPE is FIRST.

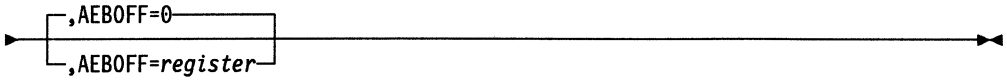


Function Specifies whether the AAB contains both a first and a last control block pointer, or just a first control block pointer. When MODE=BOTH, the AAB has pointers to both the first and last control blocks.

Format BOTH or FIRST.

Default BOTH.

Remarks When MODE=FIRST, the AAB only contains a pointer to the first control block and the default for TYPE is FIRST.



Function Specifies the register containing the value that is the offset to the AEB within the control block to be chained.

Format Register notation.

Default No offset.

Remarks AEBOFF=0 indicates that the embedded AEB is the first field in the control block to be chained.

After the control block has been chained, the AAB chain pointers point to the beginning of the chained control blocks and not necessarily to the embedded AEBs. The AEB pointers point to the beginning of the next control block and not necessarily to the next AEB.

ACTVRIT—Activate an Internal Virtual Route

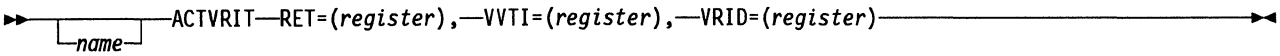
The ACTVRIT macro generates an SVC. The macro activates an internal virtual route (VR) and returns an index into the virtual route vector table (VVT). In addition, a code is returned indicating whether the virtual route was newly activated, was already activated, or could not be activated.

Note that you can use this macro only in level 5.

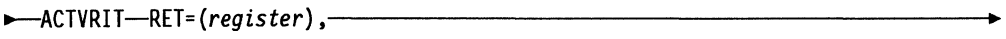
Register 0 is not allowed for register parameters.

Warning: If you share a virtual route, remember that one user can deactivate the virtual route and destroy the sessions that other users have on that virtual route. If the response to the ACTVRIT macro shows that the virtual route is already active, try to start a different virtual route rather than use the virtual route that is currently active. SDLC monitor mode uses virtual route 7.2.

Syntax



Parameters



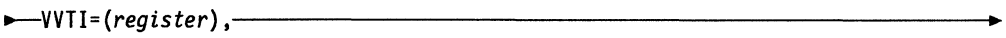
Function Specifies the register to contain the return code. The return codes are:

X'00000' A new virtual route was created.
X'00015' The virtual route was already active.
X'00012' The virtual route was not activated.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for the VVT index (VVTI).

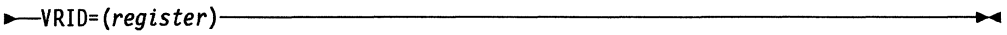


Function Specifies the register in which the VVTI is to be returned. The VVTI identifies the VRB assigned to the specified virtual route. A value of X'00000' is returned if the virtual route was not activated.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for RET.



Function Specifies the register containing the virtual route identification (VRID). The VRID consists of two values, the virtual route number (VRN) and the transmission priority field (TPF). The VRN value, 0 to 7, must be in byte 0 of the specified register. The TPF value, 0 to 2, must be in byte 1 of the specified register.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

If the VRID value is not valid, the virtual route will not be activated.

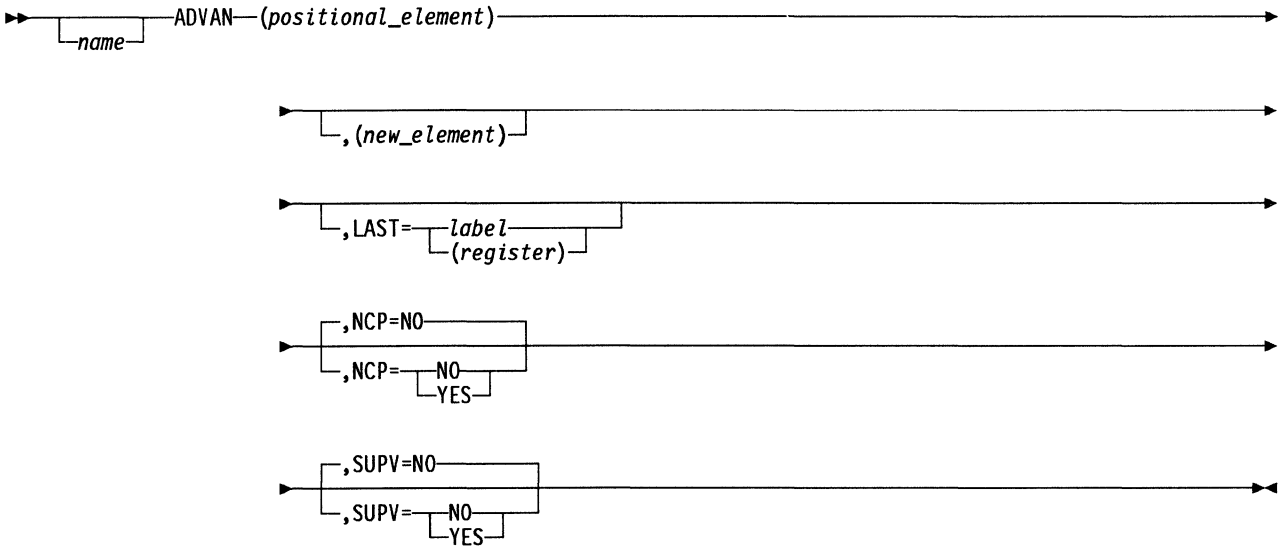
For internal virtual routes, the TPF has no effect on the flow of path information units (PIUs), except for boundary buffer pool (BPOOL) allocation priorities.

ADVAN—Advance a Pointer to Next Queue Element

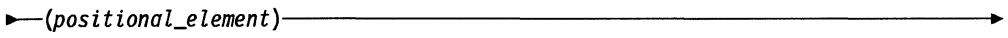
The ADVAN macro returns a pointer to the next sequential element enqueued to a specified system queue. The element is not dequeued.

Register 0 is not allowed for register parameters.

Syntax



Parameters

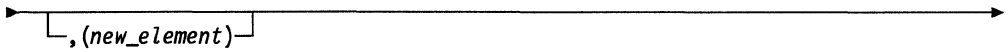


Function Specifies the register containing the address of the positional element in the queue.

Format Register notation.

Default None.

Remarks If NCP=NO, register 1 is not allowed.



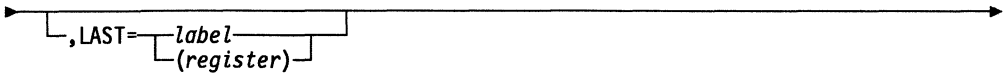
Function Specifies a register that is to receive the address of the located element.

Format Register notation.

Default The address is returned in the register containing the positional element address.

Remarks If NCP=NO, register 1 is not allowed. Register 4 is standard.

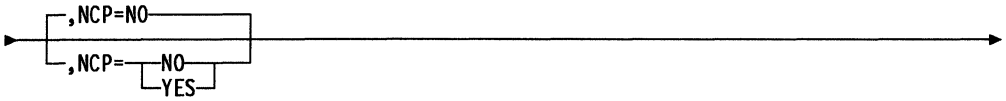
The contents of the register specified are set to 0 if the positional element is the last element on the queue.



Function Specifies an address to be branched to if the positional element is the last element on the queue.

Format Register or label notation.

Default None.

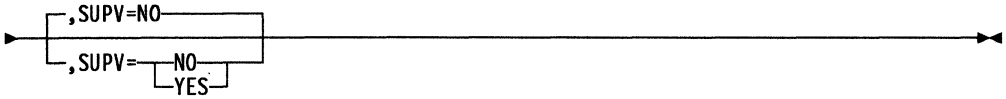


Function Specifies the type of code to be generated based on the issuer's storage protection key. NCP=YES specifies that inline code is to be generated. NCP=NO specifies that an SVC is to be generated.

Format YES or NO.

Default NO.

Remarks Specify NCP=NO when your code is in the user area of storage with a protection key not equal to 0.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

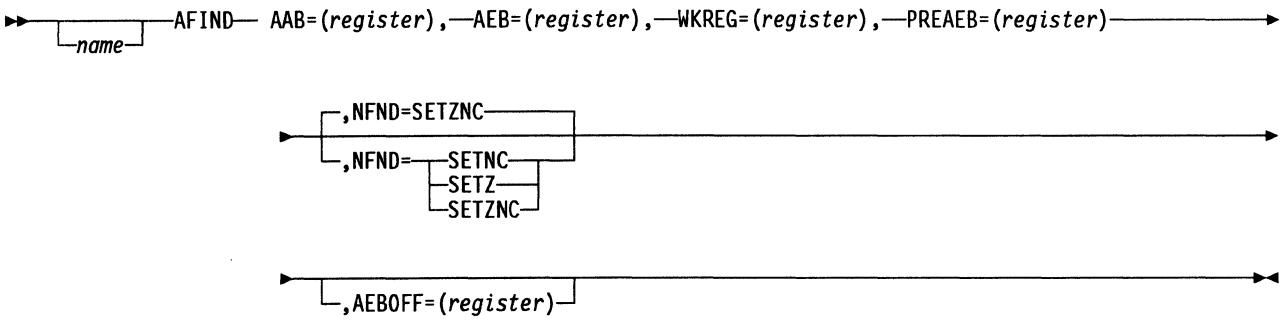
Default NO.

AFIND—Scan ACHAIN and Return the Address of the Next ACHAIN Element

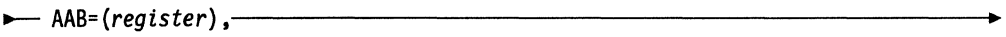
The AFIND macro scans an ACHAIN and returns the address of the element in the chain that immediately precedes the element specified.

Register 0 is not allowed for register parameters.

Syntax



Parameters

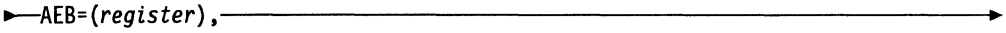


Function Specifies the register that contains the address of the achain anchor block (AAB) to which the element is chained.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AEB, WKREG, or PREAEB.



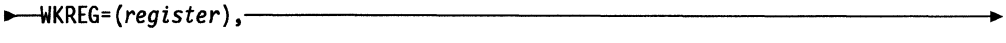
Function Specifies the register containing the address of the achain element block (AEB) for which the preceding element is to be found.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AAB, WKREG, or PREAEB.

The AAB can have both a first and last control block pointer, or it can have only a first control block pointer.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

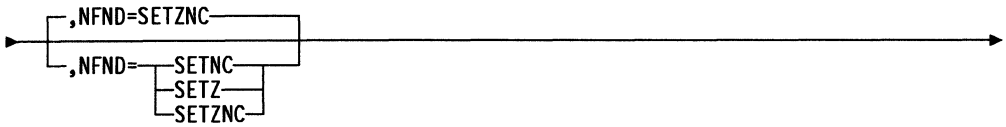
Remarks The register specified must not be the same as the register specified for AAB, AEB, or PREAEB.

Function Shows the register in which the address of the preceding element is to be returned.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AAB, AEB, or WKREG.



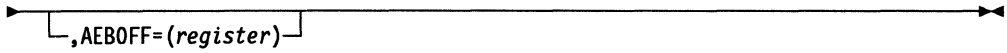
Function Specifies whether the C latch, the Z latch, or both are to indicate whether the specified element (AEB) was found in the ACHAIN.

Format SETZ, SETNC, or SETZNC.

Default SETZNC.

Remarks: For SETZNC or SETZ, the Z latch is set to 1 if the AEB is not found or to 0 if the AEB is found.

For SETZNC or SETNC, the C latch is set to 0 if the AEB is not found in the ACHAIN or to 1 if the AEB is found.



Function Specifies a register that contains the offset to the AEB within the control block to be found. If you do not specify AEBOFF, the offset to the AEB is 0, indicating that the embedded AEB is the first field in the found control block.

Format Register notation.

Default No offset.

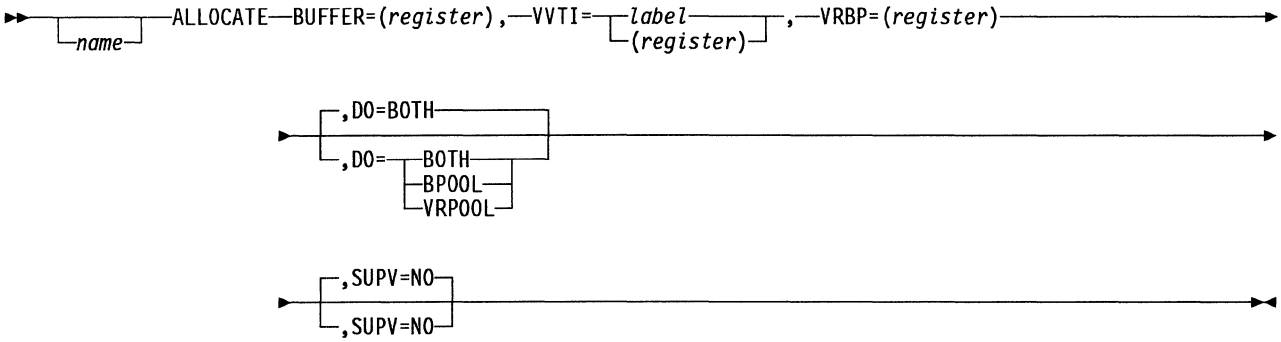
Remarks After AFIND is finished, PREAEB contains the address of the preceding control block in the chain rather than the address of the preceding AEB in the chain.

ALLOCATE—Associate Buffers with a Virtual Route

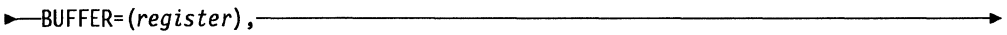
The ALLOCATE macro marks buffers (PIUs) received on a virtual route as being in the boundary buffer pool (BPOOL), VRPIU pool, or both, and associates the buffers with a virtual route. The buffers are marked with the virtual route vector table index (VVTI) of the virtual route on which they are received. The VVTI is placed in the buffer header.

Register 0 is not allowed for register parameters.

Syntax



Parameters

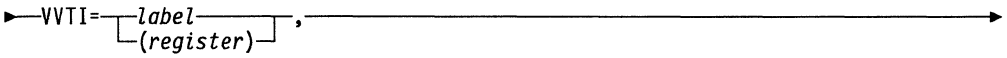


Function Specifies the register containing the address of the buffers (PIUs) to be allocated.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

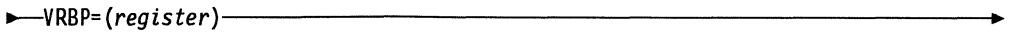


Function Specifies the VVTI of the virtual route with which the buffers (PIUs) are to be associated.

Format Register or label notation.

Default None.

Remarks Register 1 is not allowed.

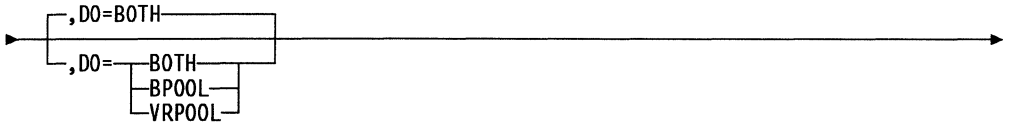


Function Specifies the register containing the address of the virtual route control block (VRB) for the virtual route on which the buffers (PIUs) are received.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.



Function Specifies the pool to which the PIU should be allocated. BPOOL specifies that the PIU should be allocated only to the boundary buffer pool. VRPOOL specifies that the PIU should be allocated only to the inbound virtual route PIU pool. BOTH specifies that the PIU should be allocated to both.

Format BPOOL, VRPOOL, or BOTH.

Default BOTH.

Function Specifies that the issuer is running in level 5.

Format NO.

Default NO.

ANDIF—Provide a Logical AND for an IF Macro

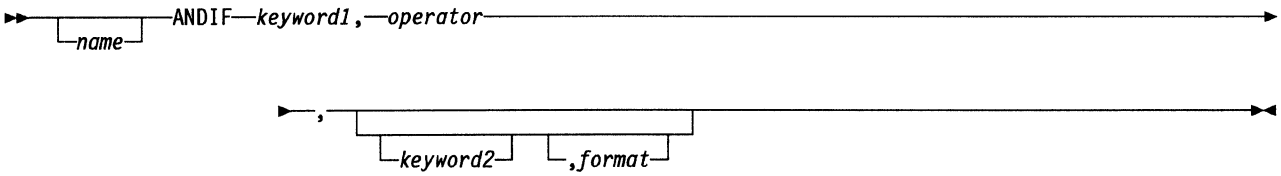
The ANDIF macro, used with the comparison format of the IF macro, provides a logical AND decision capability for an IF-THEN-ELSE program structure.

The ANDIF macro may follow an IF, an ORIF, or another ANDIF macro. However, an IF macro must always begin the sequence. Also, because of unpredictable results, ensure that no program code is between the ANDIF macro and an immediately preceding or following program-structuring macro.

Register 0 is not allowed for register parameters.

Note: For efficiency, clarity, and better structure, use the IF macro with the AND or OR keyword rather than using the ANDIF macro.

Syntax



For a description of the keywords, see the IF macro description.

Example

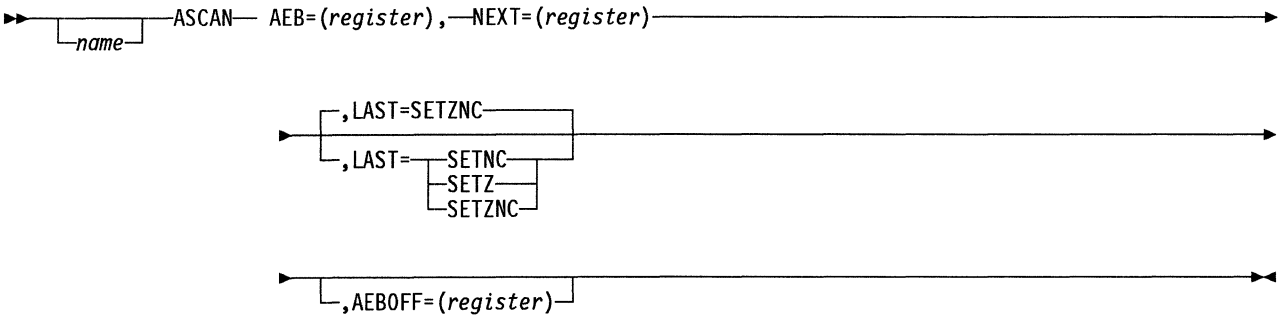
```
IF RGTEMP,EQ,X'40',I
ANDIF RGSTAT,EQ,X'88',I
  THEN x
    -
    -
  ELSE y
    -
    -
ENDIF
```

ASCAN—Return the Next Element Address in an ACHAIN

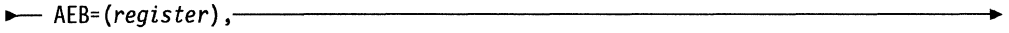
The ASCAN macro returns the address of the next element following the specified element in an ACHAIN. If the element is not in a chain or is the last element in the chain, an address of 0 is returned.

Register 0 is not allowed for register parameters.

Syntax



Parameters

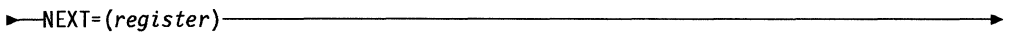


Function Specifies the register containing the address of the achain element block (AEB) for which the following element is to be found.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for NEXT.



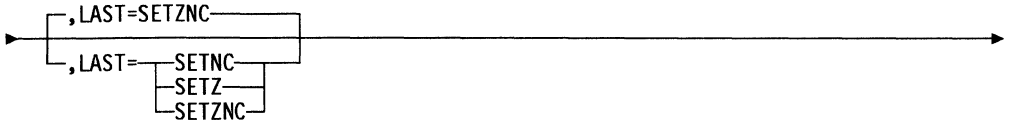
Function Shows the register to receive the address of the next AEB in the ACHAIN.

Format Register notation.

Default None.

Remarks The register is set to 0 if the specified AEB is the last AEB in the ACHAIN or if the AEB is not currently in an ACHAIN.

The register specified must not be the same as the register specified for AEB.



Function Specifies whether the C latch, the Z latch, or both are to indicate whether the specified element (AEB) is the last in a chain.

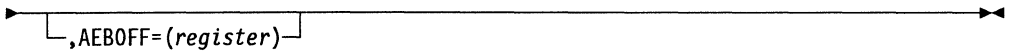
Format SETZ, SETNC, or SETZNC.

Default SETZNC.

Remarks For SETZNC or SETZ, the Z latch is set to 1 if the specified element is not the last AEB in the chain or to 0 if it is the last.

For SETZNC or SETNC, the C latch is set to 0 if the specified element is not the last AEB in the chain or to 1 if it is the last.

When the AEB is not in an ACHAIN, the latches are set the same as when the AEB is the last element.



Function Specifies a register that contains the offset to the AEB within the control block to be found. If you do not specify AEBOFF, the offset to the AEB is 0, indicating that the embedded AEB is the first field in the found control block.

Format Register notation.

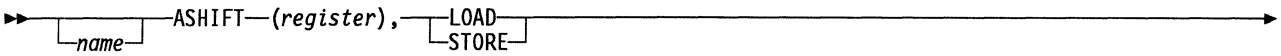
Default No offset.

Remarks After ASCAN is finished, NEXT contains the address of the next control block in the chain rather than the address of the next AEB in the chain.

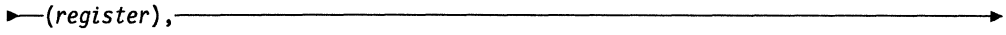
ASHIFT—Shift a Field Left or Right

The ASHIFT macro shifts a field left or right 2 bit positions.

Syntax



Parameters

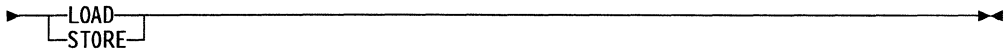


Function Specifies the register that contains the address constant to be shifted.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



Function LOAD specifies that the field is to be shifted to the left. STORE specifies that the field is to be restored to a 16-bit value and, therefore, shifted to the right.

Format LOAD or STORE.

Default None.

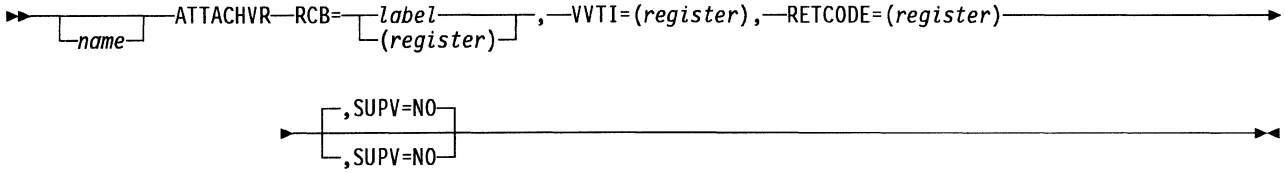
ATTACHVR—Associate a Session with a Virtual Route

The ATTACHVR macro associates a session with a virtual route. The resource connection block (RCB) of the resource associated with the session is chained to the virtual route vector table (VVT) entry for the virtual route with which the session is associated.

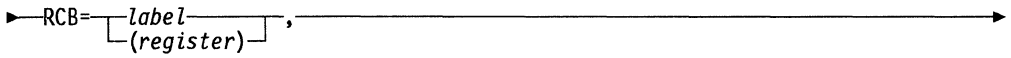
Note that you can use this macro only in level 5.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the address of the RCB associated with the virtual route.

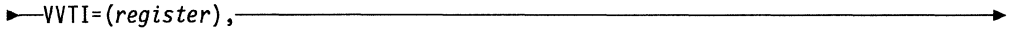
Format Register or label notation.

Default None.

Remarks The register specified must not be the same as the register specified for the virtual route vector table index (VVTI).

Neither register 1 nor register 6 is allowed.

If the RCB's VVTI field is not 0 on input to ATTACHVR, ATTACHVR will fail with a return code of X'0004'



Function Specifies the register containing the VVTI to be stored in the RCB.

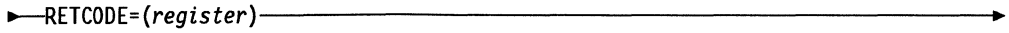
Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for RCB.

Neither register 1 nor register 6 is allowed.

The VVTI must be right-justified in the register.



Function Specifies a register to contain the return code. The return codes are:

X'0000' The macro executed successfully.

X'0004' RCB is already associated with a virtual route, indicated by a nonzero VVTI field in the RCB.

X'0006' RCB is not associated with this virtual route, but is associated with another.

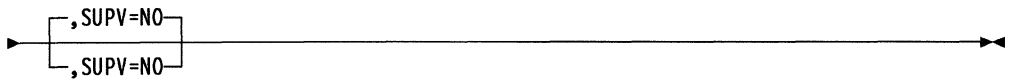
X'0008' The virtual route is inoperative.

Format Register notation.

Default None.

Remarks The register specified can be the same as the register specified for the VVTI or RCB keyword.

Register 6 is not allowed.



Function Specifies that the issuer is running in level 5.

Format NO.

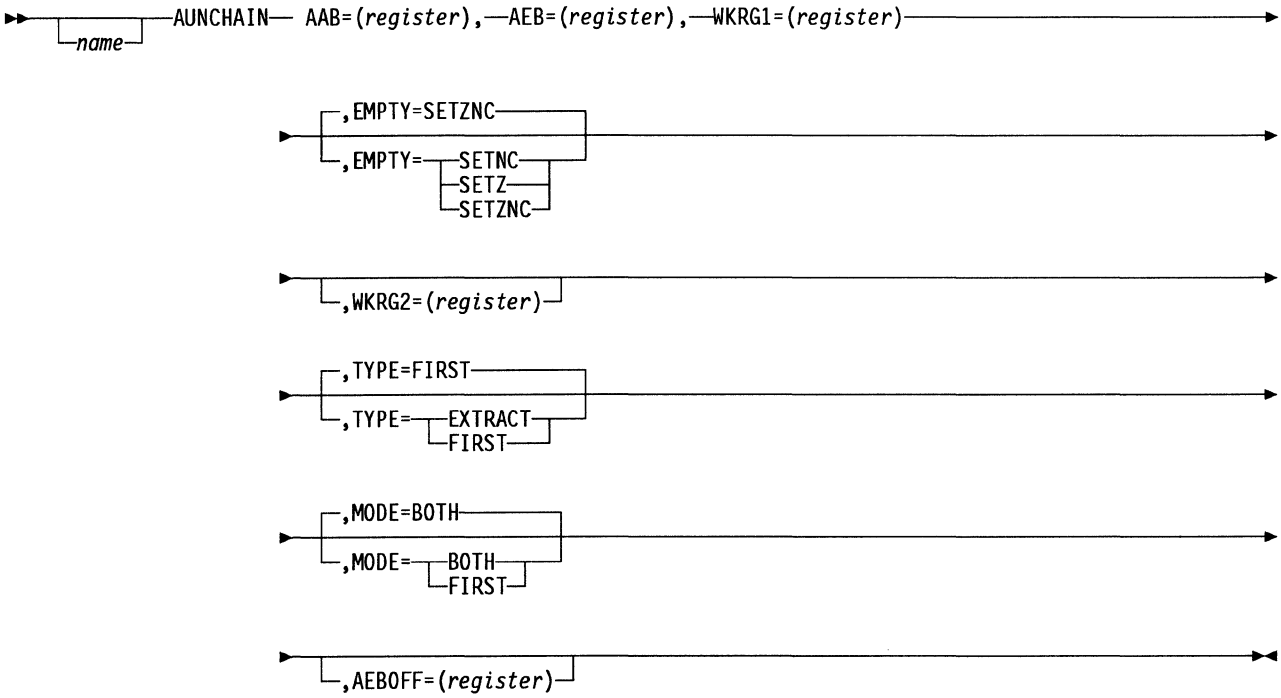
Default NO.

AUNCHAIN—Remove an Element from an ACHAIN

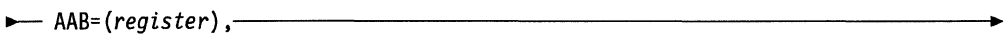
The AUNCHAIN macro removes an element from an ACHAIN. The element can be removed from any position in the chain.

Register 0 is not allowed for register parameters.

Syntax



Parameters

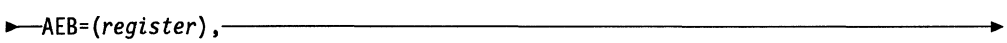


Function Specifies the register containing the address of the achain anchor block (AAB) from which the element is to be removed.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for the achain element block (AEB), WKRG1, or WKRG2.



Function Specifies a register that contains the address of the element to be unchained, or the address to which the unchained element is to be returned.

If TYPE=EXTRACT, AEB specifies the register that contains the address of the AEB to be unchained. If TYPE=FIRST or if TYPE is omitted, AEB specifies the register in which the address of the unchained element is to be returned.

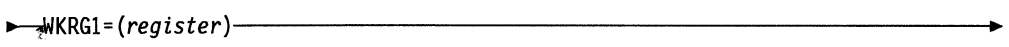
Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AAB, WKRG1, or WKRG2.

The register remains unchanged if TYPE=FIRST or if TYPE is omitted and the ACHAIN is empty.

AUNCHAIN zeroes the chain pointer field in the element that is unchained.

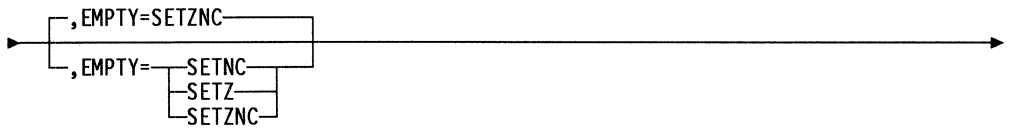


Function Specifies a work register, the contents of which may be altered during execution of the macro. If TYPE=EXTRACT and the AEB specified is not found on the ACHAIN, or if TYPE=FIRST and there are no path information units (PIUs), this register will contain a return code of X'04'; otherwise, the register value will be 0.

Format Register notation.

Default None.

Remarks The register specified must not be the same as the register specified for AAB, AEB, or WKRG2.



Function Specifies whether the C latch, the Z latch, or both are to indicate whether the ACHAIN becomes empty after the specified element (AEB) is unchained.

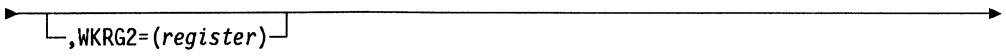
Format SETZ, SETNC, or SETZNC.

Default SETZNC.

Remarks For SETZNC or SETZ, the Z latch is set to 0 if the chain does not become empty or to 1 if the chain becomes empty.

For SETZNC or SETNC, the C latch is set to 1 if the chain does not become empty or to 0 if the chain becomes empty.

If the AEB specified is not on the ACHAIN, the latches are set as if the chain were empty.

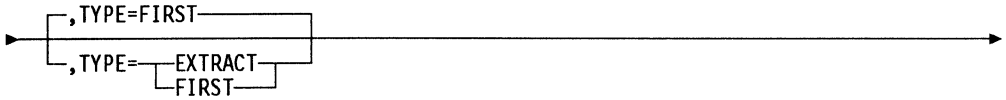


Function Specifies a work register, the contents of which may be altered during execution of the macro. Code this keyword only when TYPE=EXTRACT.

Format Register notation.

Default None.

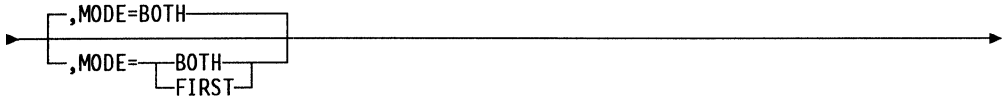
Remarks The register specified must not be the same as the register specified for AAB, AEB, or WKRG1.



Function Specifies whether the element is to be unchained from the head (FIRST) of the chain or from elsewhere (EXTRACT) in the chain.

Format FIRST or EXTRACT.

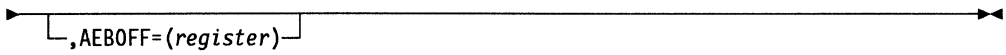
Default FIRST.



Function Specifies whether the AAB contains both a first and a last control block pointer or just a first control block pointer. When MODE=BOTH, the AAB has pointers to both the first and last control blocks. When MODE=FIRST, the AAB contains only a pointer to the first control block.

Format BOTH or FIRST.

Default BOTH.



Function Specifies a register that contains the offset to the AEB within the control block to be found. If you do not specify AEBOFF, the offset to the AEB is 0, indicating that the embedded AEB is the first field in the found control block.

Format Register notation.

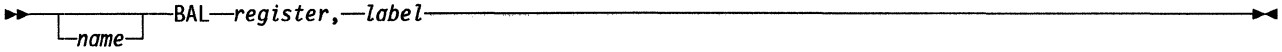
Default No offset.

Remarks After the control block has been unchained, the AAB chain pointers point to the beginning of the a-chained control blocks and not necessarily to the embedded AEBs. The AEB pointers point to the beginning of the next control block and not necessarily to the next AEB.

BAL—Branch and Link

The BAL macro generates a branch and link instruction to be used by the controller.

Syntax



Parameters

Function Specifies the register used to store the return address.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.

Function Specifies the location to be given control.

Format Label notation.

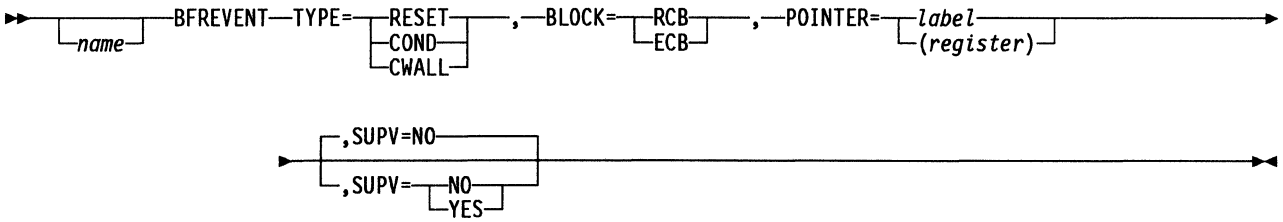
Default None.

BFREVENT—Place an ECB on a Buffer Event Queue

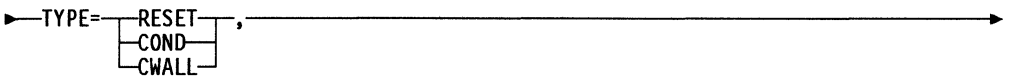
The BFREVENT macro places an event control block (ECB) on either the slowdown event queue or the CWALL event queue. The task specified in the ECB is triggered when the specified level of buffers is available.

Note: If SUPV=YES, you must provide a valid save area.

Syntax



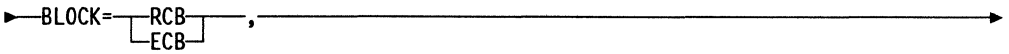
Parameters



Function Specifies whether to trigger the task when NCP is not in slowdown (TYPE=COND) or when the buffer level is above the CWALL value (TYPE=CWALL). If TYPE=RESET, the buffer event is canceled.

Format RESET, COND, or CWALL.

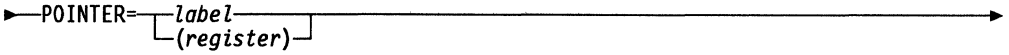
Default None.



Function Specifies whether the POINTER keyword is the address of the ECB or of the resource connection block (RCB) containing the ECB.

Format RCB or ECB.

Default None.



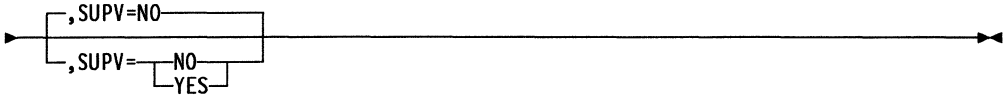
Function Specifies the address of the RCB or ECB.

Format Register or label notation.

Default None.

Remarks If SUPV=YES, label notation is not valid.

Register 0 is not allowed.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

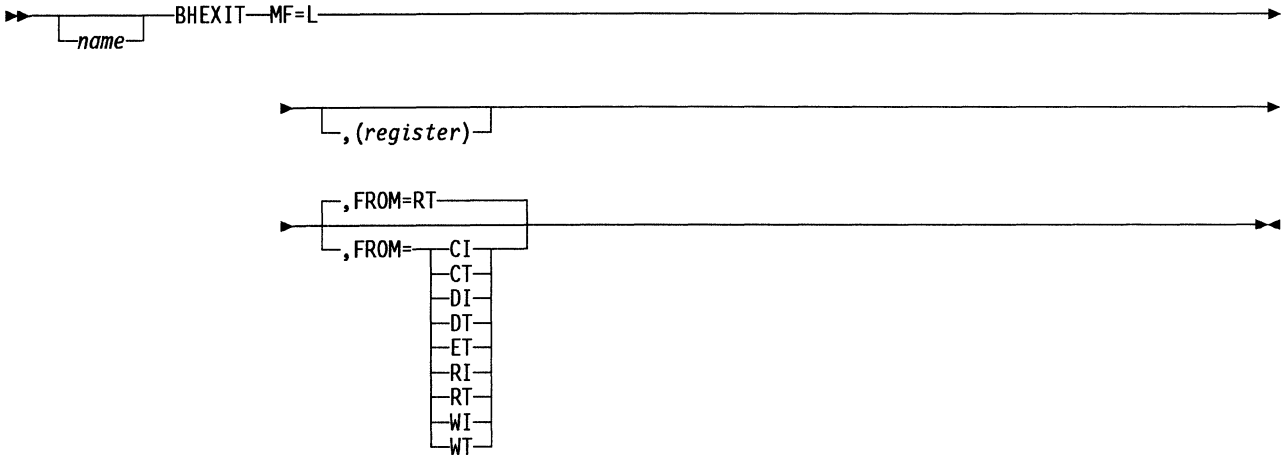
BHEXIT—Execute a Block Handling Routine

The BHEXIT macro is used by the initiator and terminator subtasks to execute block handling routines (BHRs) at point 2. The generated code tests to see whether a BHR extension exists for the device associated with the device base control block (DVB) address, and if so, branches to the BHR check routine.

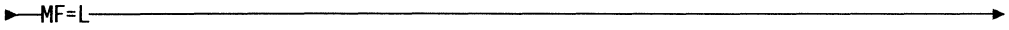
The macro has three forms: list (MF=L), execute (MF=E), and stand-alone. The list form of the macro sets up the registers for the exit but does not do the actual exit; this form alters only the contents of register 1(0). The execute form of the macro does the actual exit; it alters the contents of registers 1, 6, and 7. The stand-alone form of the macro combines the functions of the list and the execute forms; it also alters the contents of registers 1, 6, and 7.

List (MF=L) Form

Syntax



Parameters

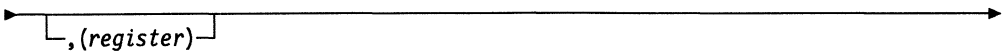


Function Specifies the list form of the macro. The list form sets up the registers for the exit, but does not do the actual exit.

Format L specifies list.

Default If you omit this keyword, the stand-alone form is used.

Remarks This keyword alters the contents of register 1(0).

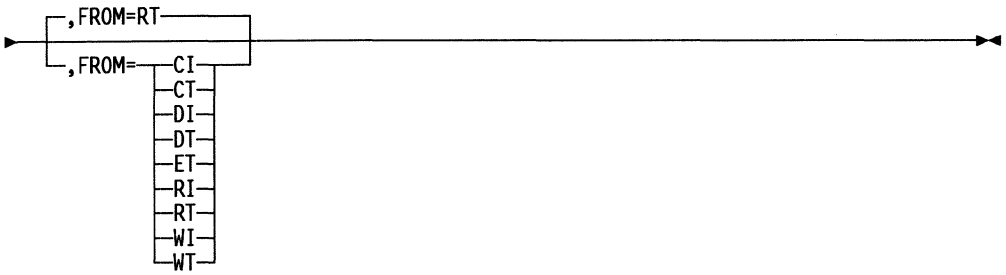


Function Specifies the register containing the DVB address.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



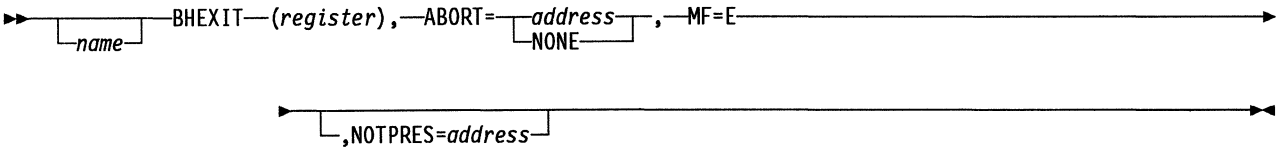
Function Specifies the type of I/O subtask issuing the BHEXIT macro.

Format Any one of R (read), W (write), C (contact), D (disconnect), and E (error), combined with I (initiator) or T (terminator).

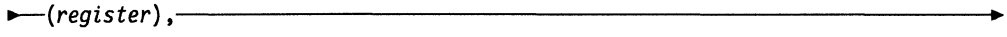
Default RT (read terminator), including display service-seeking terminator, common-read terminator, and read-entry point of chain terminator.

Execute (MF=E) Form

Syntax



Parameters

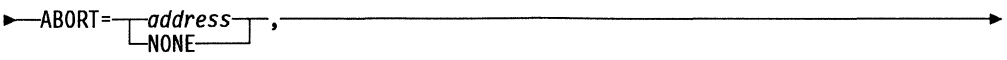


Function Specifies the register containing the DVB address.

Format Register notation.

Default None.

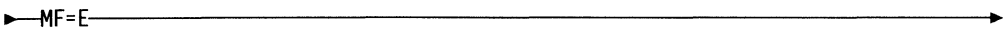
Remarks Register 0 is not allowed.



Function Specifies the address to be branched to if the BHR issues an ABORT macro with a nonzero completion code, or if the BHR removes the block control unit from the queue.

Format NONE or label notation.

Default None.

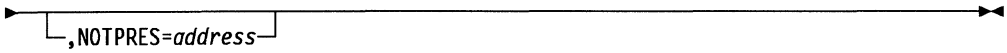


Function Specifies the execute form of the macro. The execute form does the actual exit.

Format E specifies execute.

Default If you omit this keyword, the stand-alone form is used.

Remarks This keyword alters the contents of registers 1, 6, and 7.



Function Specifies an address to be branched to if a BHR extension does not exist.

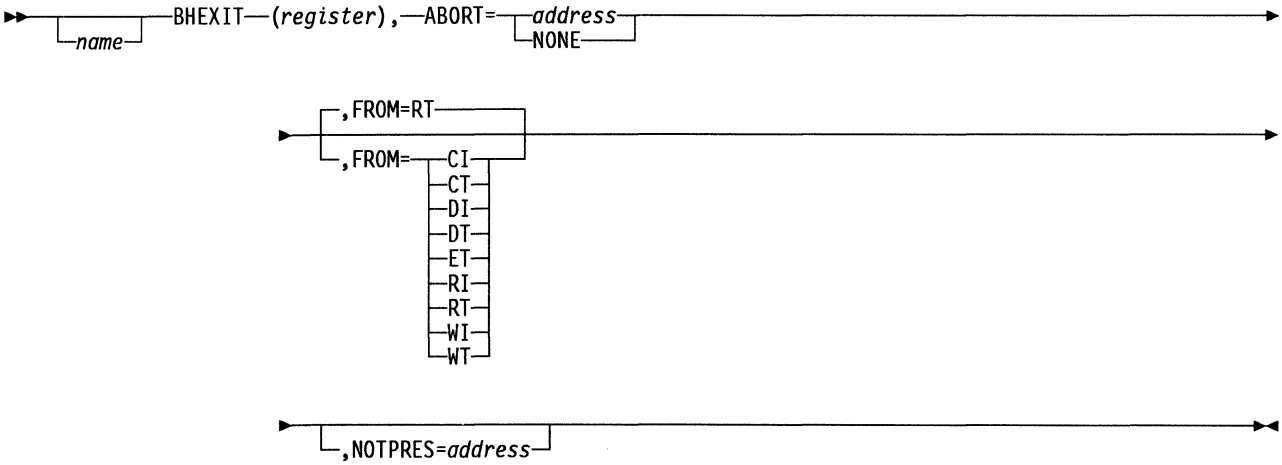
Format Label notation.

Default Branches to the next sequential instruction following the BHEXIT macro expansion.

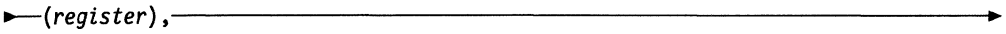
Stand-Alone Form

The stand-alone form of the macro combines the functions of the list and execute forms. It alters the contents of registers 1, 6, and 7.

Syntax



Parameters

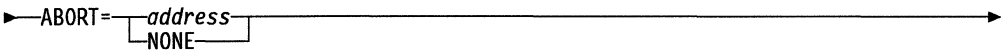


Function Specifies the register containing the DVB address.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.

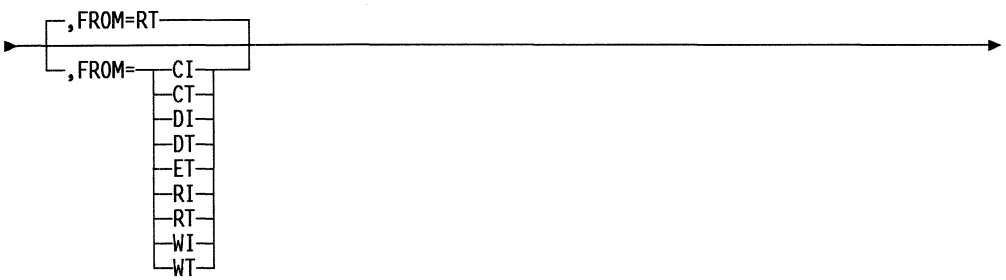


Function Specifies the address to be branched to if the BHR issues an ABORT macro with a nonzero completion code, or if the BHR removes the block control unit from the queue.

Format NONE or label notation.

Default None.

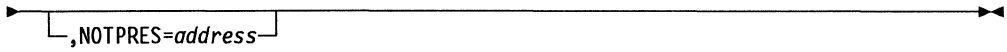
Remarks Specify ABORT=NONE when FROM=RI, WI, CI, or DI.



Function Specifies the type of I/O subtask issuing the BHEXIT macro.

Format Any one of R (read), W (write), C (contact), D (disconnect), and E (error), combined with I (initiator) or T (terminator).

Default RT (read terminator), including display service-seeking terminator, common-read terminator, and read-entry point of chain terminator.



Function Specifies an address to be branched to if a BHR extension does not exist.

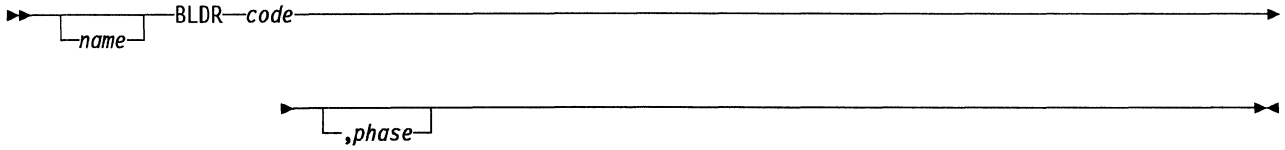
Format Label notation.

Default Branches to the next sequential instruction following the BHEXIT macro expansion.

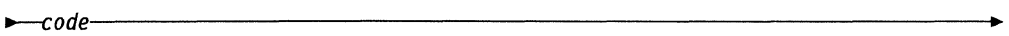
BLDR—Build a System Response

I/O subtasks use the BLDR macro to build a phase 1, 2, or 3 system response in register 1(0) in program level 5. The macro combines the specified response code with either the current phase as contained in the system response field of the BTU (BCUSRES) or the specified phase given in the macro. The BLDR macro uses equates that are defined by the XXCXTRES macro.

Syntax



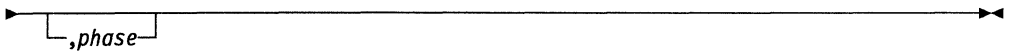
Parameters



Function Specifies the numeric code for the desired response. System responses define the codes to be used. Refer to the section on BTU responses in *NCP and EP Reference Summary and Data Areas*, Volume 2.

Format Absolute notation.

Default None.



Function Specifies the desired phase of the response.

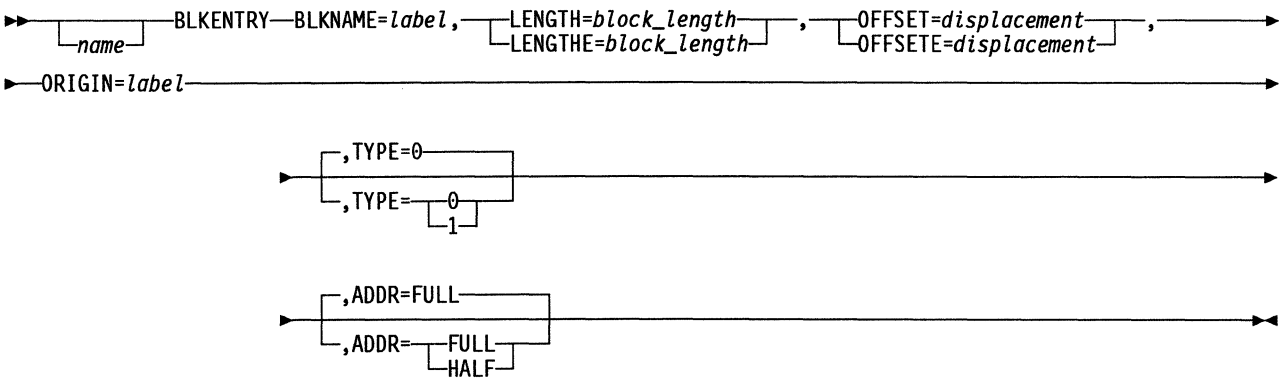
Format 1, 2, or 3.

Default The phase currently contained in the BTU system response field is used.

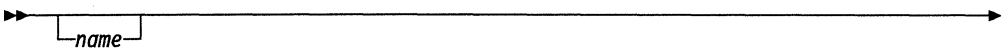
BLKENTRY—Describe a Block or a Table

The BLKENTRY macro describes a block or a table that is pointed to, as a dependent, by a parent block previously described in the block dump table (BDT). The parent block could be described in either the GRPENTRY or another BLKENTRY macro. The BLKENTRY and GRPENTRY macros can be used by IBM special products or user-written code as an interface to the dump formatter. They provide control block pointers, lengths, and names so that IBM special products or user-written code control blocks can appear in a formatted NCP dump.

Syntax



Parameters

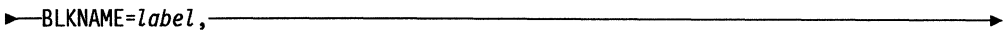


Function Referred to in the ORIGIN keyword of another (and dependent) BLKENTRY macro.

Format Label notation.

Default None.

Remarks The label (name) is the name given to a macro that describes a parent block that points to a following dependent block. In the macro that describes the dependent block, the ORIGIN keyword contains the parent block macro label to complete the chained relationship.

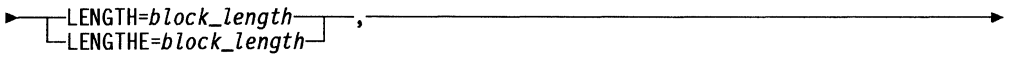


Function Specifies the name of the block to be printed in the formatted dump; the name can be up to 5 characters long.

Format Label notation; character string from 1 to 5 bytes long.

Default None.

Remarks The name is truncated if it is longer than 5 characters.

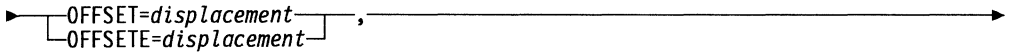


Function Specifies the block length.

Format Decimal notation (LENGTH), or symbolic notation (LENGTHE); from 0 to 252 bytes.

Default None.

Remarks Since symbolic expressions are not resolved during the macro expansion phase, cross-checking to ensure that pointers in this control block do not exceed the length of the control block is not done.

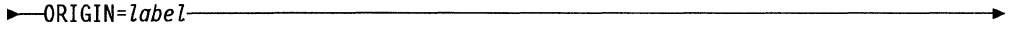


Function This is the offset location of the pointer within the parent block whose describing macro is referred to in the ORIGIN keyword. The pointer contains the address of the block being described in this BDT entry.

Format Decimal notation (OFFSET), or symbolic notation (OFFSETE); from 0 to 252 bytes.

Default None.

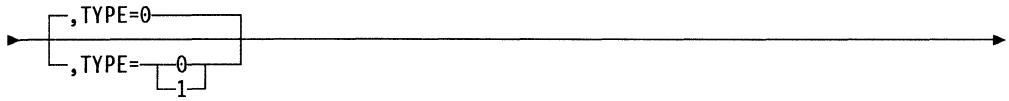
Remarks Since symbolic expressions are not resolved during the macro expansion phase, cross-checking to ensure this offset does not exceed the length of the parent control block is not done.



Function Specifies the label (name) of another BLKENTRY or the GRPENTRY macro that describes a parent block that points to the dependent block being described in this BDT entry.

Format Any name of a block previously described.

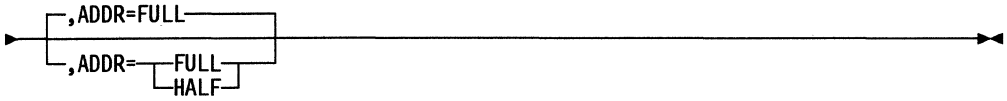
Default None.



Function Specifies whether a table or a block is to be described.

Format 1 for a table; 0 for a block.

Default 0.



Function Specifies whether the address of the block or table to be dumped is a halfword or a fullword.

Format HALF for a halfword; FULL for a fullword (4 bytes).

Default FULL.

Branch Macros—Branch Conditionally to a Specified Location

All branch macros generate a series of conditional branches. The result is a branch to a specified location if a certain condition is met. The condition is the result of a compare operation or the result of an arithmetic or logical operation.

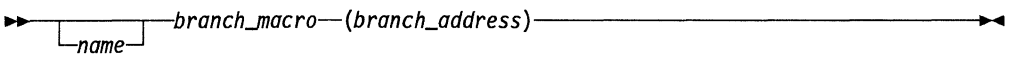
The BAL branch macro, which is described on page 44, is not included in this section because it is formatted differently than the macros discussed here.

Table 5 shows the branch macros that have the same format.

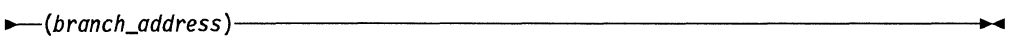
Table 5. Branch Macro Conditions

Macro	Condition for Branching
BCR	The C condition latch is on after an arithmetic or logical operation. The branch address must be specified in register notation.
BH	A is greater than B after comparing A and B.
BM	The result is negative after an arithmetic or logical operation.
BMZ	The result is negative or 0 after an arithmetic or logical operation.
BNC	The C condition latch is not on after an arithmetic or logical operation.
BNDH	The result is a branch-to-address within the low 64KB of storage.
BNE	A is not equal to B after comparing A and B.
BNH	A is not greater than B after comparing A and B.
BNL	A is not less than B after comparing A and B.
BNZ	The Z condition latch is not on after an arithmetic or logical operation.
BP	The result is positive after an arithmetic or logical operation.
BPZ	The result is positive or 0 after an arithmetic or logical operation.
BZR	The Z condition latch is on after an arithmetic or logical operation. The branch address must be specified in register notation.

Syntax



Parameters



Function Specifies the location to be given control if the condition is met.

Format Register or label notation.

Default None.

Remarks Register 0 is not allowed.

BTDELETE—Delete a Node from a Binary Tree

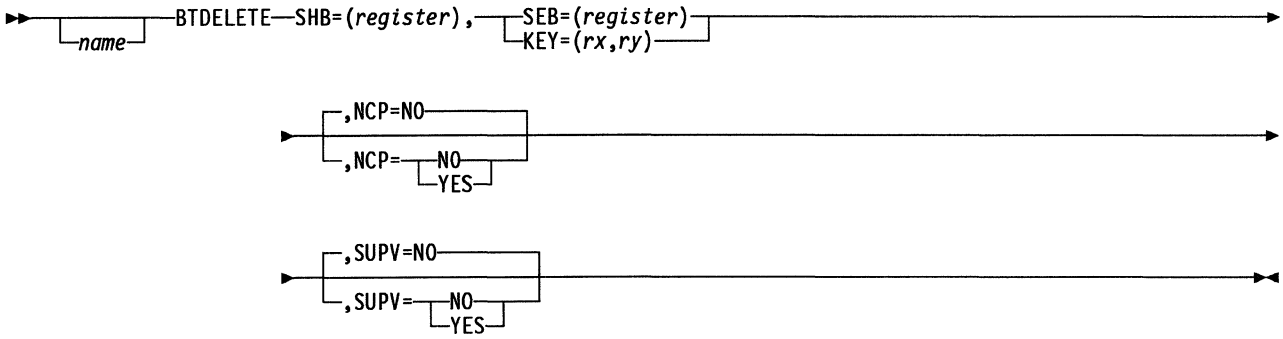
The BTDELETE macro deletes a node or search element control block (SEB) from a binary search tree.

You can use register 7 as an input register, but the contents may not be preserved. Register 6 must point to a valid save area. Register 0 is not allowed.

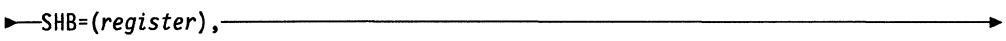
On exit, register 1 contains one of the following return codes:

X'00000' The SEB was successfully deleted from the tree.
X'00001' An SEB with the specified key was not found. This return code indicates a severe problem with the tree that must be corrected.
X'00002' The specified SEB is not in a search tree.

Syntax



Parameters

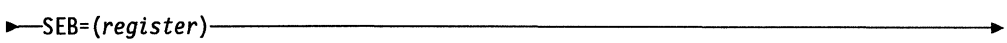


Function Specifies the register containing the pointer to the search header block (SHB) that anchors the tree.

Format Register notation.

Default None.

Remarks The register specified cannot be used for any other keyword.



Function Specifies the register containing the pointer to the SEB.

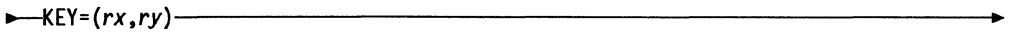
Format Register notation.

Default None.

Remarks You must specify either an SEB pointer or provide a search key, but not both.

You must code SEB rather than KEY if checking is required to determine whether the SEB is in a search tree before deletion processing begins (indicated by a return code of X'00002').

The register specified cannot be used for any other keyword.



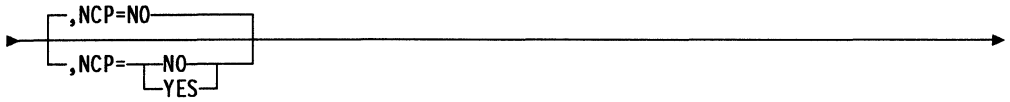
Function Specifies the two registers to contain the 4-byte key of the SEB to be deleted. The *rx* register contains the first 2 bytes, and the *ry* register contains the second 2 bytes.

Format Register notation.

Default None.

Remarks You must specify either an SEB pointer or provide a search key, but not both.

The register you specify must not be the same as the register specified for any other keyword.

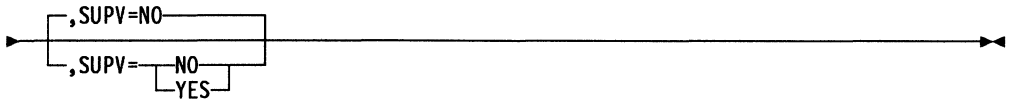


Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES specifies that the issuer is running in NCP storage protection key 0; inline code is generated. NCP=NO specifies that the issuer is running in NCP storage protection key 1; SVC code is generated.

Format YES or NO.

Default NO.

Remarks Specify this parameter only when SUPV=NO.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

BTECHECK—Check Whether a Binary Tree Is Empty

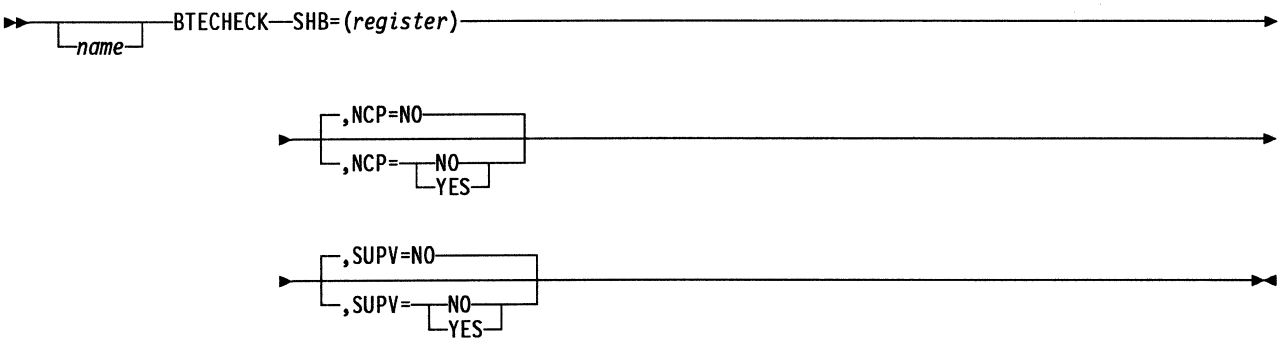
The BTECHECK macro checks a binary tree to see if it is empty.

On exit, register 1 contains one of the following return codes:

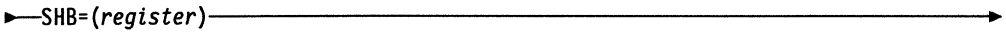
X'00000' The binary tree is empty.
X'00001' The binary tree is not empty.

Register 6 must point to a valid save area. Register 0 is not allowed.

Syntax



Parameters

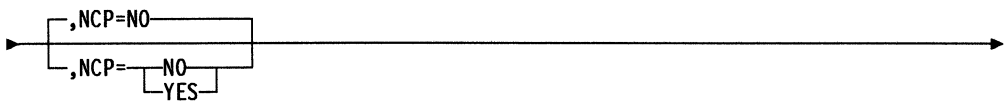


Function Specifies the register containing the pointer to the search header block (SHB) that anchors the tree.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword.

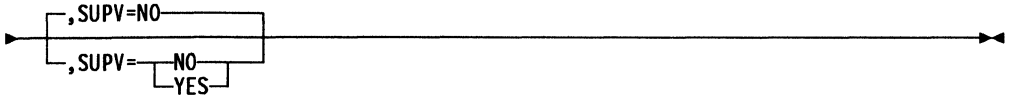


Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES specifies that the issuer is running in NCP storage protection key 0; inline code is generated. NCP=NO specifies that the issuer is running in NCP storage protection key 1; SVC code is generated.

Format YES or NO.

Default NO.

Remarks Specify this parameter only when SUPV=NO.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

BTINSERT—Insert a Node into a Binary Search Tree

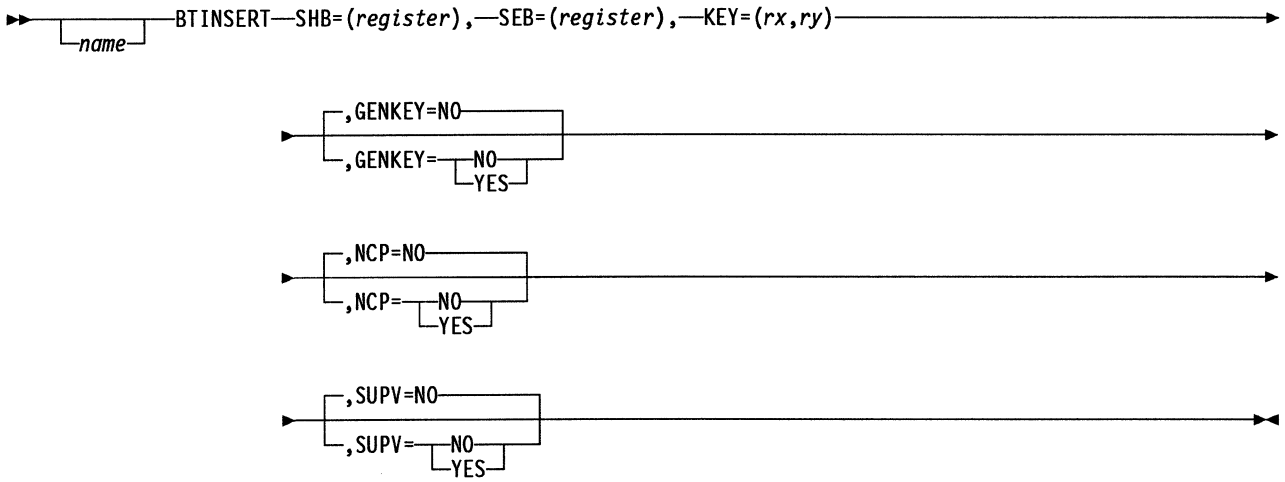
The BTINSERT macro inserts a node search element control block (SEB) into a binary search tree.

Use register 7 as an input register, keeping in mind that this is a volatile storage area. Register 6 must point to a valid save area. Register 0 is not allowed.

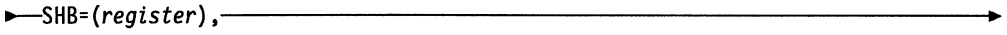
On exit, register 1 contains one of the following return codes:

X'00000' The SEB was successfully inserted to the tree.
X'00001' An SEB with the specified key was already in the tree. This return code indicates a severe problem with the tree that must be corrected.
X'00002' The search tree header control block (SHB) future key field was never initialized.

Syntax



Parameters

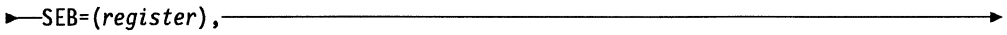


Function Specifies the register containing the pointer to the SHB that anchors the tree.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword.

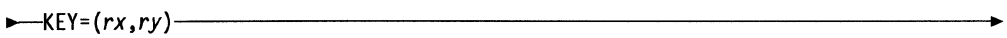


Function Specifies the register containing the pointer to the SEB to be inserted.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword.



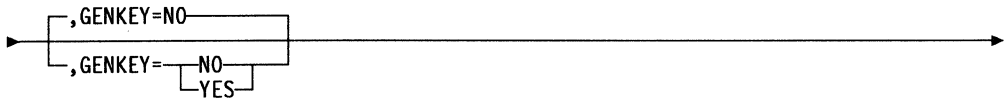
Function Specifies which registers are to be put in the SEB. If GENKEY=NO, KEY specifies the registers containing the search key to be put in the SEB. If GENKEY=YES, KEY indicates that the registers will contain an NCP-generated key value that is put in the SEB on return from BTINSERT. The *rx* register contains the high 2 bytes of the key, and the *ry* register contains the low 2 bytes.

Format Register notation.

Default None.

Remarks The two registers specified can only be used for this keyword.

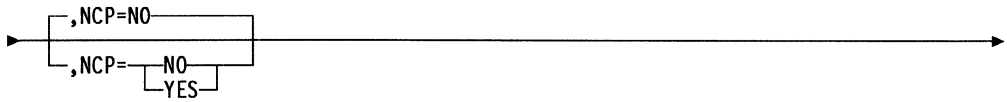
You cannot use register 1 if GENKEY=YES is specified.



Function Specifies whether NCP is to generate the search key for this SEB. If GENKEY=YES, NCP will generate a search key and assign it to the SEB.

Format YES or NO.

Default NO.

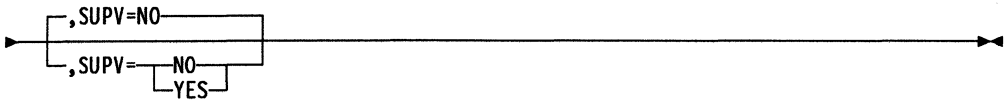


Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES specifies that the issuer is running in NCP storage protection key 0; inline code is generated. NCP=NO specifies that the issuer is running in NCP storage protection key 1; SVC code is generated.

Format YES or NO.

Default NO.

Remarks Specify this parameter only when SUPV=NO.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

BTSEARCH—Search a Binary Tree

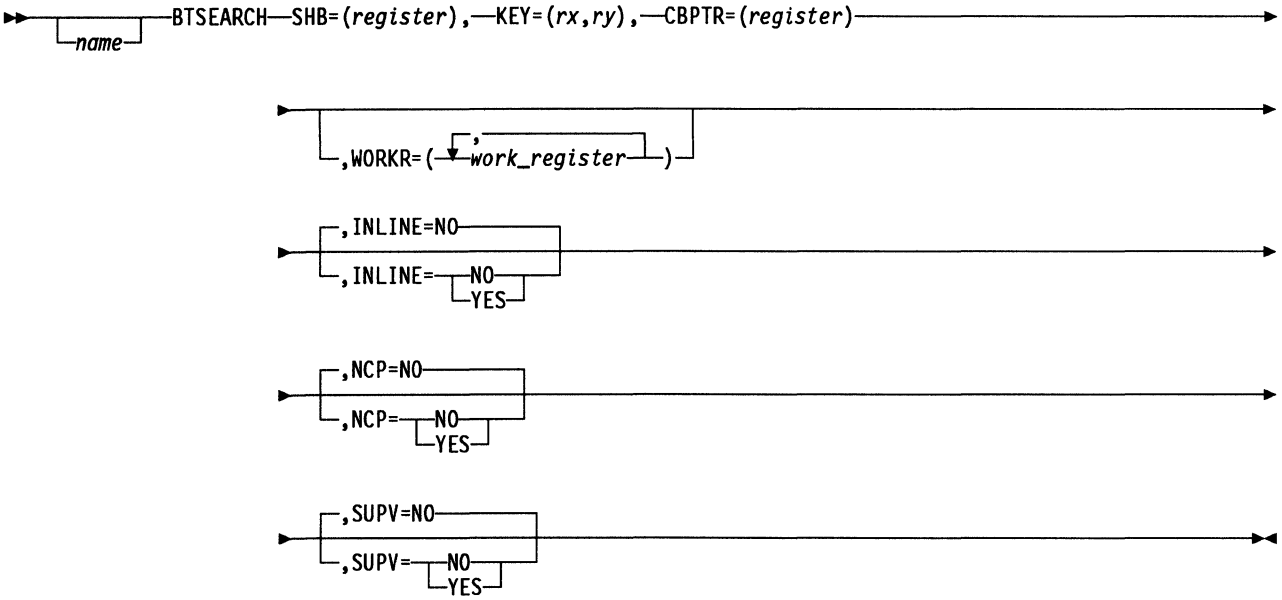
The BTSEARCH macro finds a node search element control block (SEB) in a binary search tree.

Use register 7 as an input register, keeping in mind that this is a volatile storage area. Register 6 must point to a valid save area. Register 0 is not allowed.

On exit, register 1 contains one of the following return codes:

X'00000' An SEB with a key equal to KEY was found.
X'00001' No SEB with the specified key was found.

Syntax



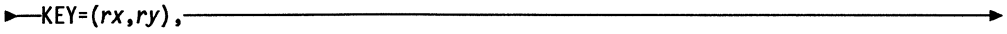
Parameters

Function Specifies the register containing the pointer to the search header block (SHB) that anchors the tree.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword. You cannot use register 6. If you specify INLINE=YES, you cannot use register 1.

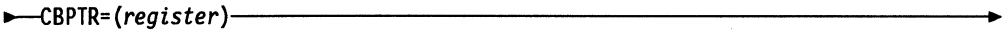


Function Specifies the two registers to contain the 4-byte key of the SEB to be found. The *rx* register contains the first 2 bytes, and the *ry* register contains the second two bytes. KEY is set to 0 if no matching SEB is found.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword. You cannot use register 6. If you specify INLINE=YES, you cannot use register 1.

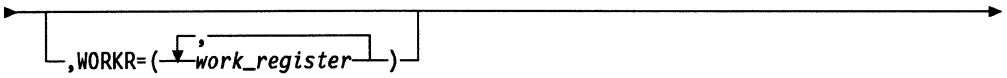


Function Specifies the register to contain a pointer to the SEB's outer control block if the SEB is found. The register is set to 0 if no SEB is found.

Format Register notation.

Default None.

Remarks The register you specify can be the same as the register specified for SHB or KEY. You cannot use register 1 or register 6.

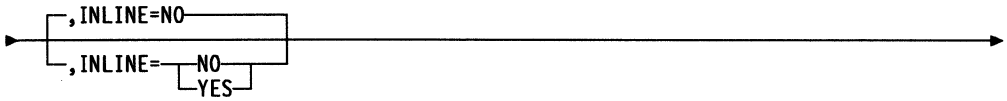


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default No work registers.

Remarks You can code any or all of the following registers: 1, 2, 3, 4, 5, or 7. You can code WORKR only when INLINE=YES.

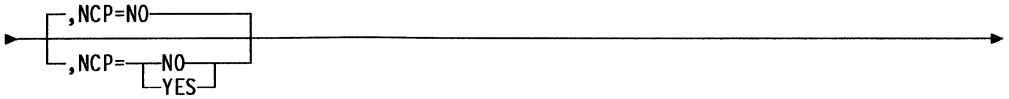


Function Specifies whether inline linkage code is generated.

Format YES or NO.

Default NO.

Remarks You can specify this parameter only if you do not code NCP. If you code INLINE, you must include the XBTSPLST DSECT. Specifying INLINE=YES along with WORKR is recommended for performance reasons; however, you cannot use register 1 for SHB, KEY, or CBPTR.

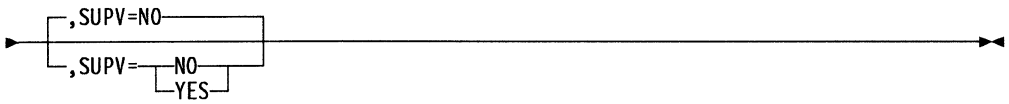


Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES specifies that the issuer is running in NCP storage protection key 0; inline code is generated. NCP=NO specifies that the issuer is running in NCP storage protection key 1; SVC code is generated. Note that specifying INLINE=YES is recommended for performance reasons when SUPV=YES, unless register 1 must be used for SHB, KEY, or CBPTR.

Format YES or NO.

Default NO.

Remarks Do not specify NCP if you code SUPV=NO and INLINE. Specifying INLINE=YES rather than NCP=YES is recommended for performance reasons, unless register 1 must be used for SHB, KEY, or CBPTR.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

BUFCHK—Check if Address Is in Buffer Pool

For NCP V7R1 or V7R1F and later versions, the BUFCHK macro checks to see if an address is within the range of the buffer pool.

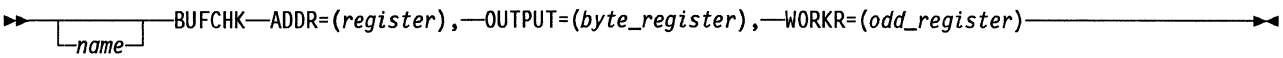
Do not specify the same register for more than one keyword.

Register 0 is not allowed for register parameters.

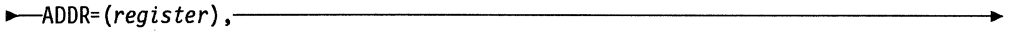
This macro generates inline code.

Note: You must invoke the XDA DSECT macro XCXTXDA and the XDB DSECT macro XCXTXDB in the routine that issues the BUFCHK macro.

Syntax



Parameters

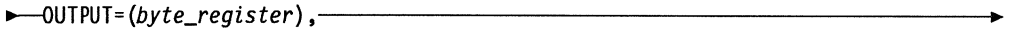


Function Specifies the register containing the address that should be checked to see if it is within the buffer pool range.

Format Register notation.

Default None.

Remarks Register 6 is not allowed.

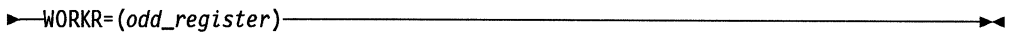


Function Specifies a byte register that will contain the following output:

X'00' The address was within the buffer pool.
X'01' The address was not within the buffer pool.

Format Byte register notation.

Default None.



Function Specifies an odd-numbered work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks Must be an odd-numbered register.

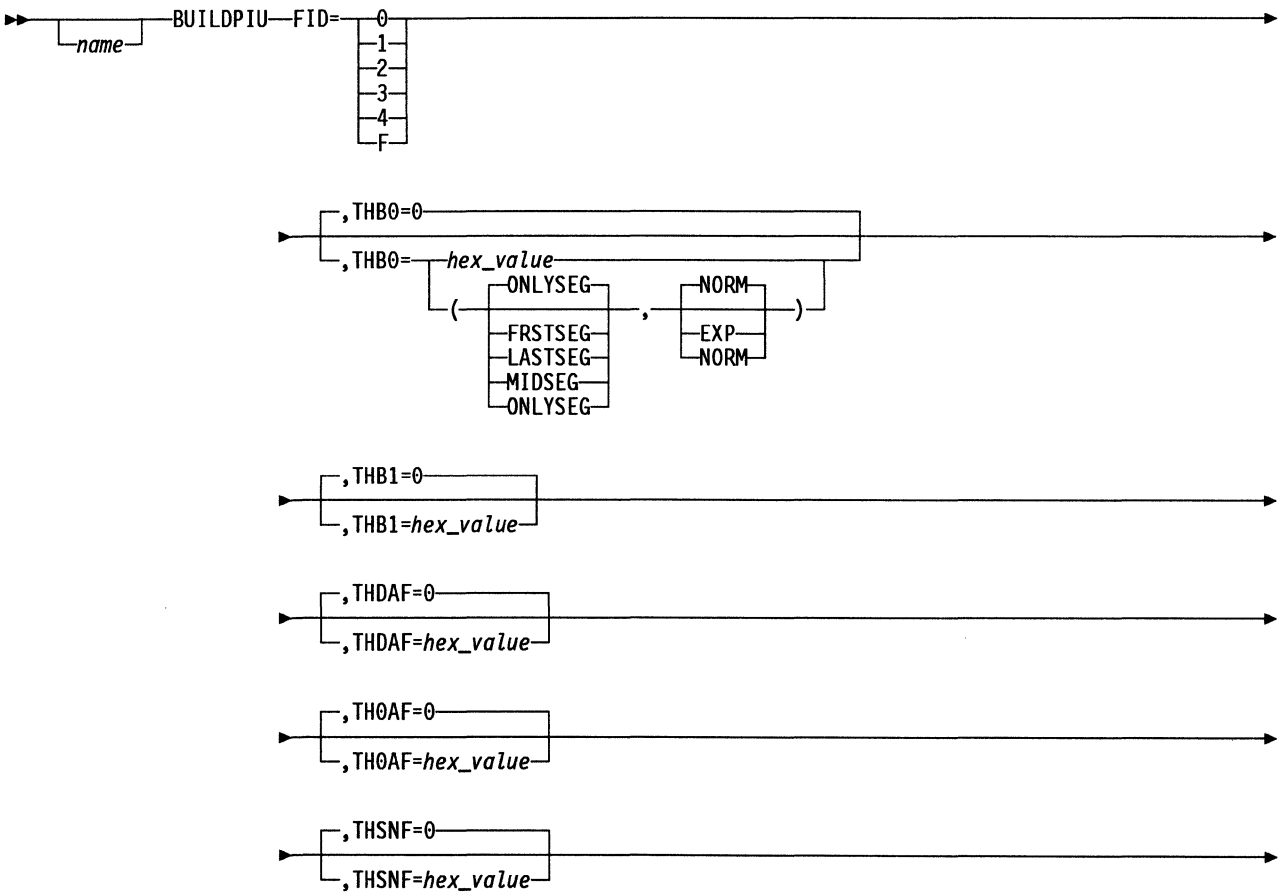
BUILDPIU—Build a Prototype PIU

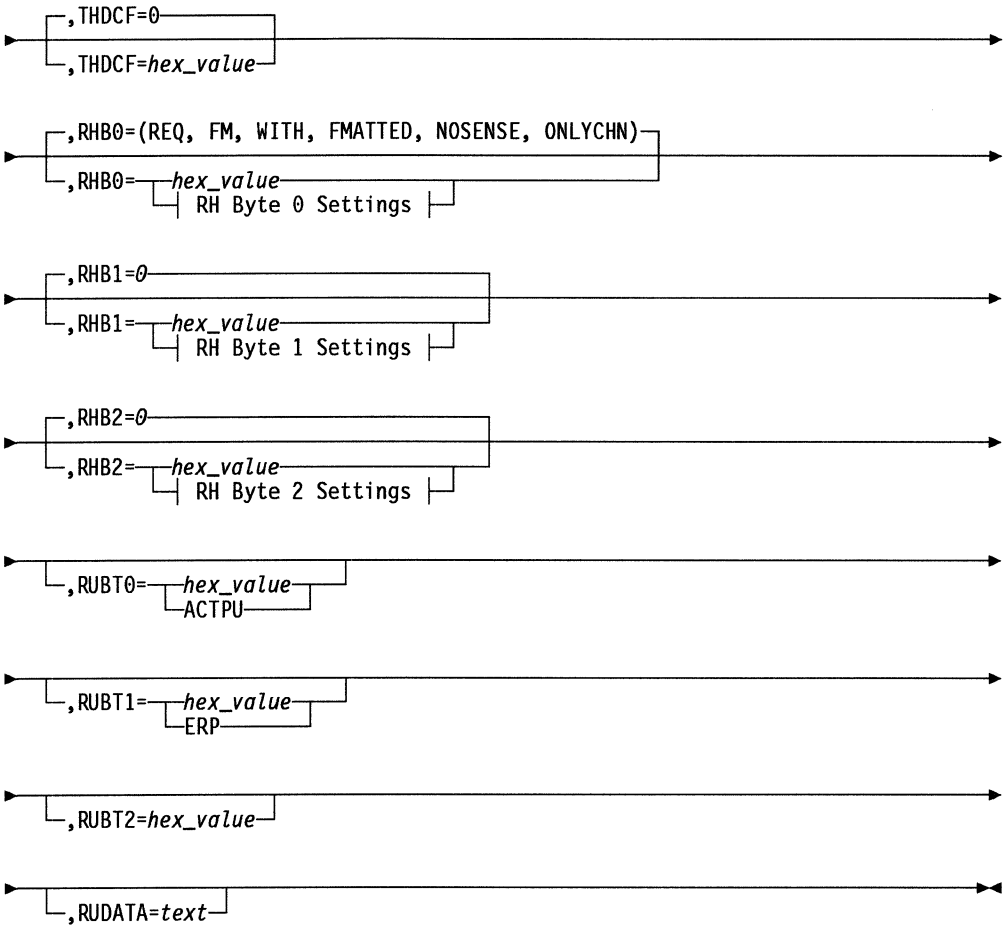
The BUILDPIU macro builds a prototype path information unit (PIU) in an inline buffer, not in a buffer from the buffer pool. To build a PIU with the BUILDPIU macro, use a sequence similar to the following:

```
      LA        3,name                 Puts BUILDPIU name in R3
*
*  Fill in UnOFFSET and UnDATCNT fields.
*
      COPYPIU   (3),(4),...,LEASE=YES  R3 points to the inline buffer and
                                       R4 points to a new buffer
      B         label
name  BUILDPIU  keywords
label EQU       *
```

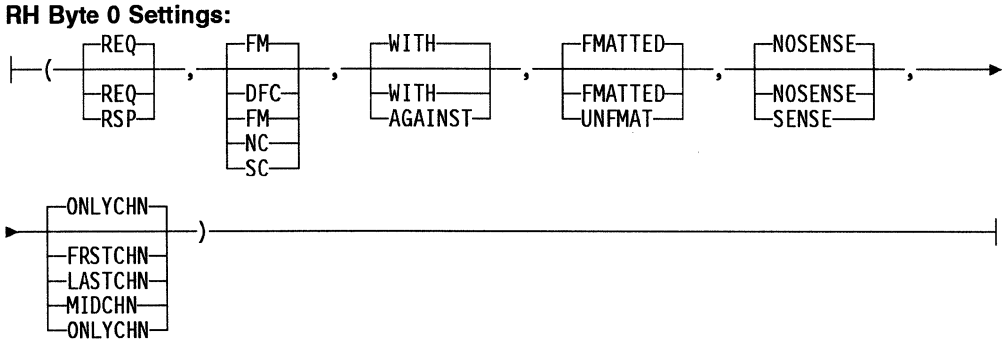
The COPYPIU macro with LEASE=YES leases a buffer from the buffer pool and copies the prototype PIU into the buffer. Fields such as OAF, DAF, and SNF should be updated with the appropriate values before the copied PIU is transmitted. The buffer's ECB is initialized appropriately; all other fields between the buffer prefix and THB0 are set to 0.

Syntax

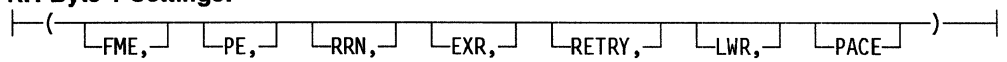




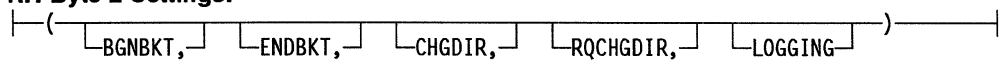
RH Byte 0 Settings:



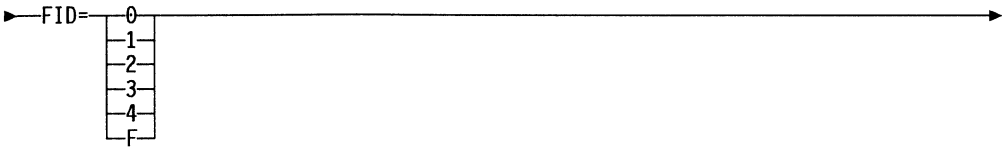
RH Byte 1 Settings:



RH Byte 2 Settings:



Parameters

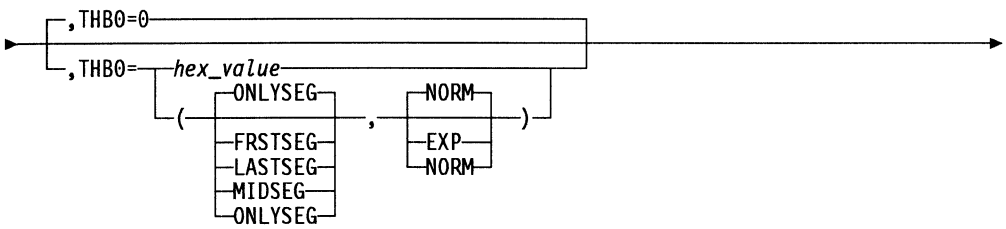


Function Specifies the PIU format identifier (FID) type to be built.

Format Absolute notation.

Default None.

Remarks If the absolute notation is not in the range of 0 to 4 or if the value or keyword is omitted, no PIU is built and an MNOTE is issued.

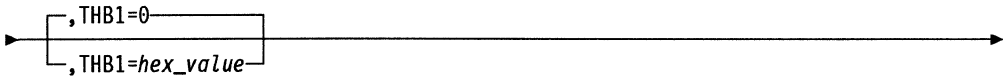


Function Specifies the segment and data flow conditions for the PIU to be built (in byte 0 of the transmission header), or the value of byte 0 of the new PIU.

Format FRSTSEG, LASTSEG, MIDSEG, or ONLYSEG, followed by either EXP or NORM, both enclosed in parentheses and separated by a comma. Use absolute hexadecimal notation for byte 0 of the transmission header.

Default X'00', ONLYSEG, or NORM.

Remarks FRSTSEG, LASTSEG, MIDSEG, and ONLYSEG refer to segments of a PIU too large for a cluster controller or device buffer (not to be confused with an NCP buffer).

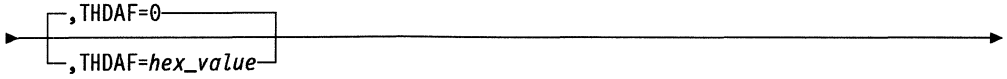


Function Specifies the second byte (byte 1) of the new PIU.

Format Absolute hexadecimal notation.

Default X'00'.

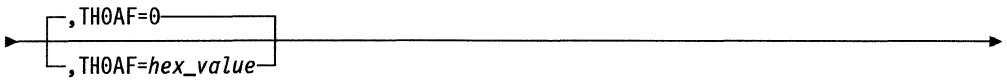
Remarks Not valid for FID type 3.



Function Specifies the destination address field (DAF) of the new PIU.

Format Absolute hexadecimal notation.

Default X'0000' for all, except FID type 2, for which it is X'00'.

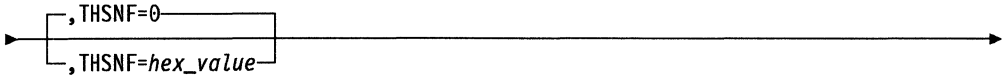


Function Specifies the origin address field (OAF) of the new PIU.

Format Absolute hexadecimal notation.

Default X'0000' for all except FID type 2, for which it is X'00'.

Remarks Not valid for FID type 3.



Function Specifies the sequence number field (SNF) of the new PIU.

Format Absolute hexadecimal notation.

Default X'0000'.

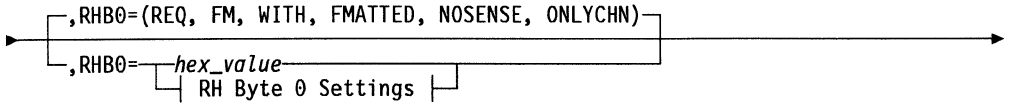
Remarks Not valid for FID type 3.

Function Specifies the data count of the new PIU. This count is equal to the sum of the length of the request/response header (RH) and the request/response unit (RU) fields of the new PIU.

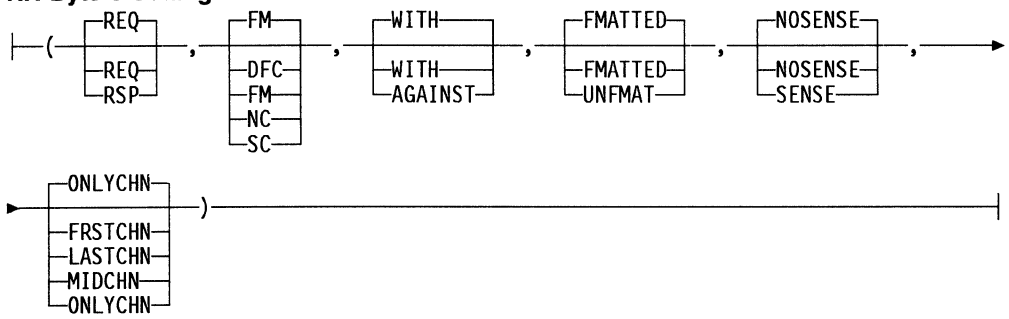
Format Absolute hexadecimal notation.

Default X'0000'.

Remarks Not valid for FID types 2 or 3.



RH Byte 0 Settings:



Function Specifies byte 0 of the RH field.

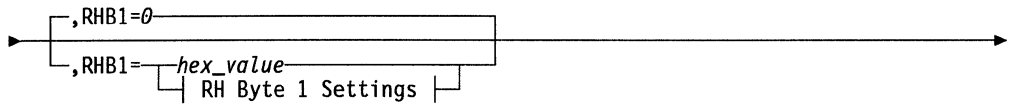
Format Absolute hexadecimal notation.

Default REQ, FM, WITH, FMATTED, NOSENSE, and ONLYCHN.

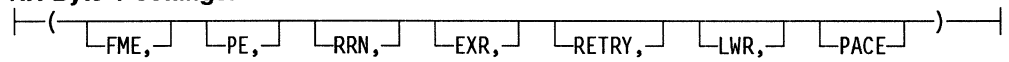
Remarks The meanings of the keywords are:

REQ	Request PIU
RSP	Response PIU
SC	Session control
NC	Network control
FM	Function management data
DFC	Data flow control
WITH	With flow
AGAINST	Against flow
FMATTED	Formatted data
UNFMAT	Unformatted data
SENSE	Sense data included
NOSENSE	Sense data not included.

FRSTCHN, LASTCHN, MIDCHN, and ONLYCHN refer to elements in a series of PIUs.



RH Byte 1 Settings:



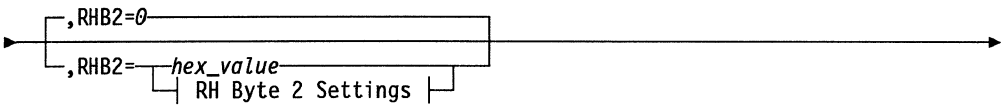
Function Specifies the second byte of the RH field of the new PIU.

Format Absolute hexadecimal notation.

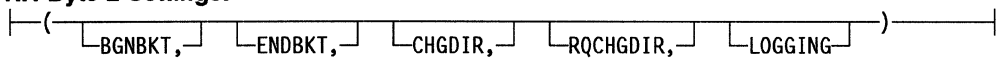
Default X'00'.

Remarks The meanings and values of the keywords are:

FME	Function management END or DR1 requested or sent, value: X'80'
PE	Path error requested or sent, value: X'40'
RRN	Relative-record number or DR2 requested or sent, value: X'20'
EXR	Exception response, value: X'10'
RETRY	Retry or BUSY requested or sent, value: X'08'
LWR	Larger window requested, value: X'04'
PACE	PACE requested or sent, value: X'01'.



RH Byte 2 Settings:



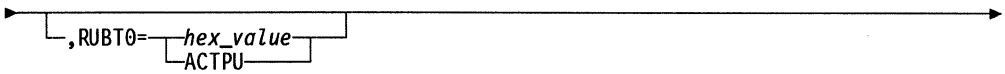
Function Specifies the third byte of the RH field of the new PIU.

Format Absolute hexadecimal notation.

Default X'00'.

Remarks The meanings of the keywords are:

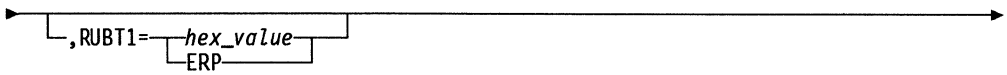
BGNBKT	Begin bracket
ENDBKT	End bracket
CHGDIR	Change direction
RQCHDIR	Request change direction
LOGGING	Logging.



Function Specifies the first byte of the RU.

Format Absolute hexadecimal notation. ACTPU results in a value of X'11'.

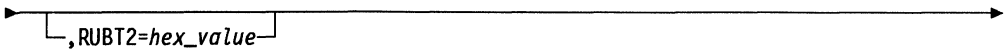
Default X'00'.



Function Specifies the second byte of the RU field.

Format Absolute hexadecimal notation.

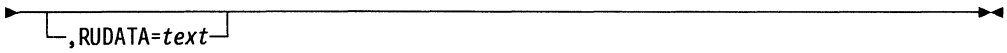
Default X'00'.



Function Specifies the third byte of the RU field.

Format Absolute hexadecimal notation.

Default X'00'.



Function Specifies the text field.

Format Absolute hexadecimal notation.

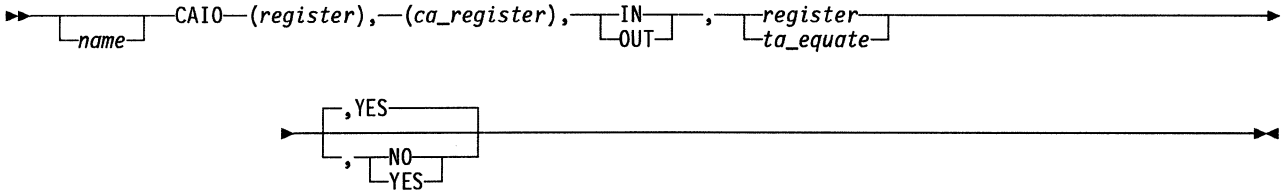
Default X'00'.

Remarks BUILDPIU will not ensure that the PIU that is built is within proper buffer boundaries.

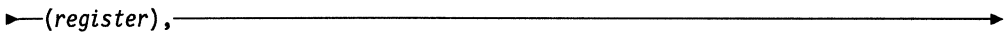
CAIO—Generate Channel Adapter Input or Output

The CAIO macro generates a channel adapter input or output instruction to be used by the communication controller.

Syntax



Parameters

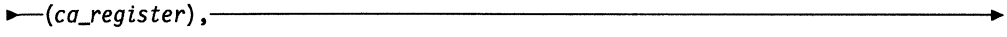


Function Specifies any of the eight general-purpose registers 0 through 7.

Format Label notation.

Default None.

Remarks Labels for registers are defined in the System Register Equates in the XSYSEQU DSECT macro.



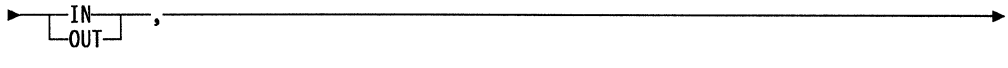
Function Specifies the channel adapter external register to be read in or written out.

Format Label notation.

Default None.

Remarks Labels for channel adapter external registers appear in the System Register Equates in the XSYSEQU DSECT macro.

Register 0 is not allowed.



Function Specifies the type of operation to be carried out as input or output. IN indicates an input or a read operation. OUT indicates an output or a write operation.

Format IN or OUT.

Default None.

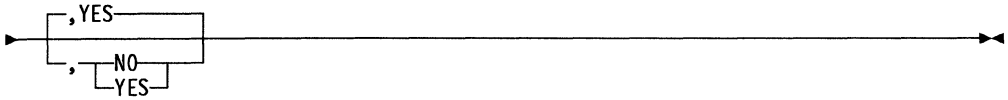
Function Passes the high byte of the tag address data, including the bus bit. This is passed either in the high byte of an odd-numbered register or as a byte equate.

Format Label notation if a register; byte equate notation if tag address (TA) equate.

Default None.

Remarks If the high byte of the tag address is passed in a register, you must use an odd-numbered register and the high byte of the tag address must be in the high byte of the register.

Register 0 is not allowed.



Function Specifies whether the adapter input/output halfword (IOH) trace is to be called. YES indicates that the trace is to be called. NO indicates that it is not to be called.

Format YES or NO.

Default YES.

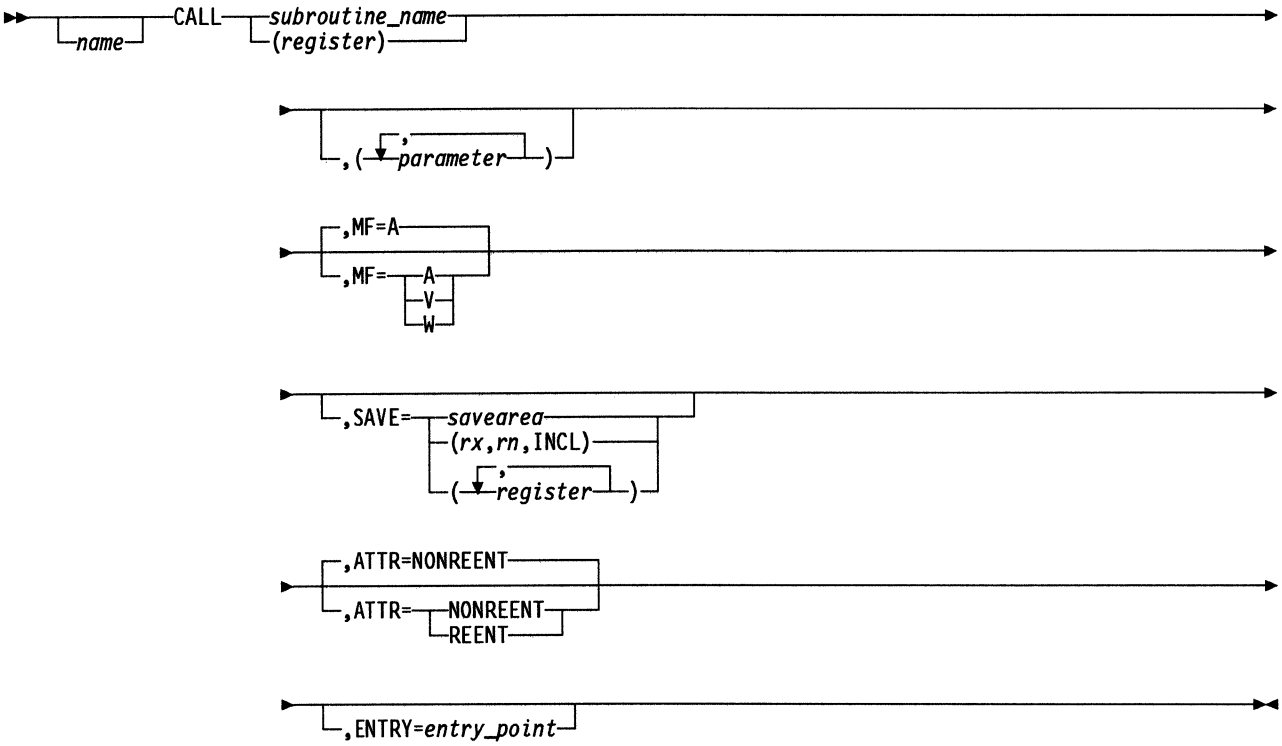
CALL—Transfer Control to a Subroutine

The CALL macro transfers control to a specified subroutine. The macro structures a save area or refers to a previously defined save area if the call is to a nonreentrant subroutine; or the macro allocates a dynamic save area if the call is to a reentrant subroutine. The macro saves the specified registers.

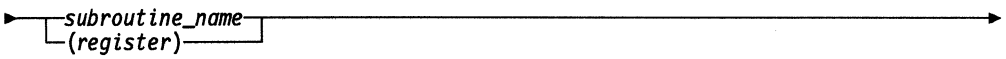
Register 0 is not allowed for register parameters.

Note: Use the SUBRTN macro to define the entry point of routines to be called by CALL.

Syntax



Parameters



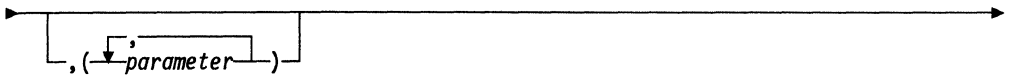
Function Specifies the name of the subroutine to be invoked.

Format Register or label notation.

Default None.

Remarks Register 1 is not allowed.

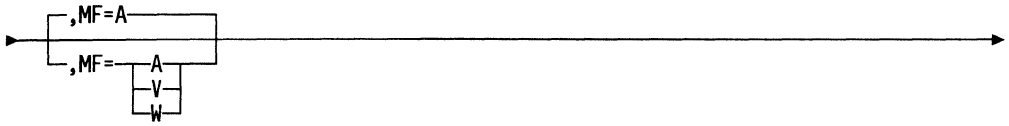
If you use label notation, the symbol must be the same as the name field of the SUBRTN macro that defines the subroutine.



Function Specifies the parameters to be passed as arguments from the calling program to the subroutine.

Format Symbolic names, coded within parentheses and separated by commas.

Default None.

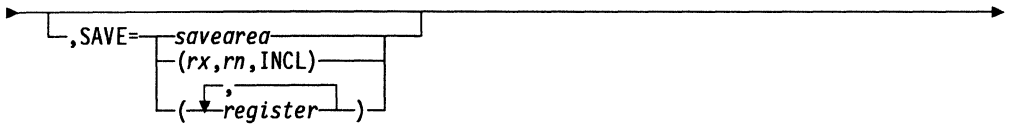


Function Specifies whether the subroutine being called is a strong external (V) reference, a local (A) reference, or a weak external (W) reference.

Format V, A, or W.

Default A.

Remarks If you specify the subroutine name in register notation, this keyword is ignored.



Function Specifies either the symbolic name of a previously defined save area (defined by a SAVEAREA macro) or the registers that are to be saved before invoking the subroutine.

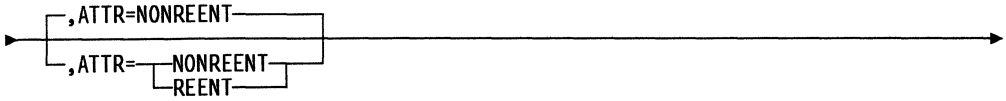
Format Label notation or registers.

Default No registers are saved.

Remarks If ATTR=REENT, you cannot specify this keyword in label notation.

In a series in which registers *rx* and *rn* specify the limits of the registers to be saved, the limits must be in ascending sequence, and INCL must be coded. In a random grouping, each register (*rx,ry,...rn*) must be specified.

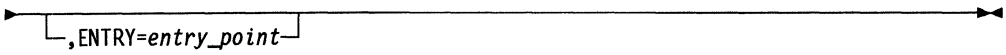
Register 1 cannot be saved.



Function Specifies whether the subroutine being invoked is reentrant or nonreentrant.

Format REENT or NONREENT.

Default NONREENT.



Function Specifies an optional entry point for the subroutine.

Format Label notation.

Default None.

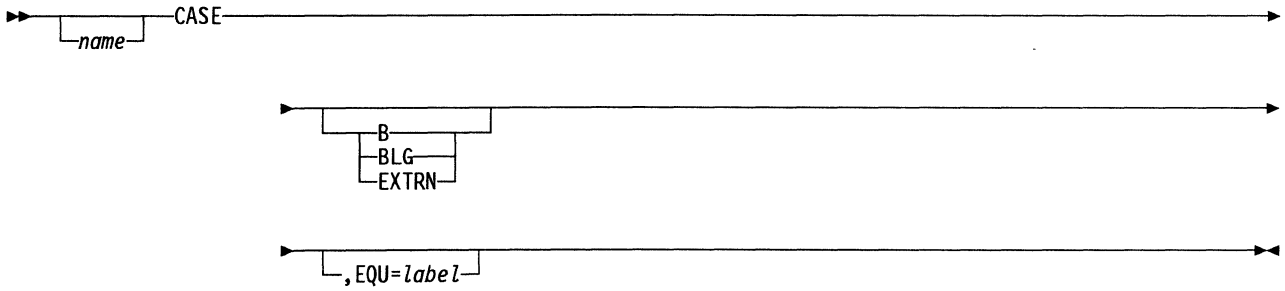
Remarks If you specify the subroutine name in register notation, this keyword is not valid.

CASE—Begin a Case Program Structure

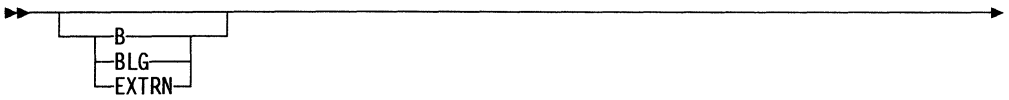
The CASE macro is used with the CASEIF, CASENTRY, CASEXIT, and ENDCASE macros to form a case program structure. The CASE macro specifies the starting point of a case routine, which may be entered from one or more CASEIF macros following a CASENTRY macro. CASEIF macros following a different CASENTRY macro cannot enter this particular case routine unless the EQU keyword is used. The end point of the case routine is specified by the ENDCASE macro.

Note: A CASE macro not equated to another CASE macro may have several CASE macros equated to it. An equated CASE macro must immediately follow the CASE macro to which it is equated before any instructions or the ENDCASE macro for the original CASE. CASE macros equated to another CASE macro share a single ENDCASE macro.

Syntax



Parameters



Function Overrides the return function specified by the RTRN keyword of the CASENTRY macro. This parameter is valid only when B or BLG or the default has been specified by the RTRN keyword of the CASENTRY macro.

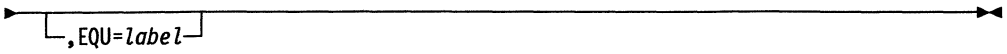
Format B specifies that a B instruction is to be used to return to CASEXIT.

BLG specifies that a BLG instruction is to be used to return to CASEXIT.

EXTRN specifies that the CASE routine is in another CSECT.

Default The return function specified by the RTRN keyword of the associated CASENTRY macro is used.

Remarks If you use the EQU keyword, do not specify this parameter.



Function Equates this CASE macro with an immediately preceding CASE macro that has an identical return function. The return function is specified either by the return parameter of the CASE macro or by the CASENTRY RTRN keyword. The equated CASE macros can be entered from the same or different CASENTRY macros. If an equated CASE macro is entered from different CASENTRY macros, the RTRN keyword of the CASENTRY macros must specify (EXIT, SUPV), RESTORE, or (BR, register).

Format To equate a CASE macro to another, give this keyword a value equal to the label of the appropriate CASE macro. That CASE macro must not also be equated to another CASE macro.

Default No equate.

CASEIF—Test for a Condition in a Case Program Structure

The CASEIF macro is used with the CASENTRY, CASEXIT, CASE, and ENDCASE macros to form a case program structure. The CASEIF macro tests for a specified condition and, if the test is true, branches to the specified case routine associated with the current CASENTRY and CASEXIT macro pair. After the case routine is executed, control returns to the point established by CASEXIT, except when the RTRN keyword on the CASENTRY macro specifies (EXIT, SUPV), RESTORE, SYSXIT, OR (BR,register). The CASEIF macro has the same functions as the IF macro. See *NCP and SSP Customization Guide* for additional information.

See the description of the IF macro on page 176 for the keywords for these formats:

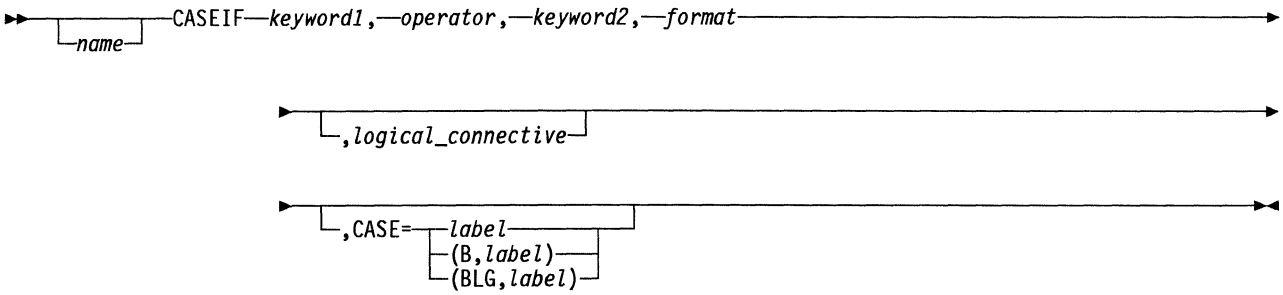
- Test CL, ZL
- Branch-on-bit
- Test-under-mask
- Comparison.

Also refer to the description of the IF macro, comparison format, on page 181, for a description of the function, format, default, and notes for the following CASEIF parameters. If none of these parameters is specified, the CASEIF macro unconditionally branches to the specified case routine.

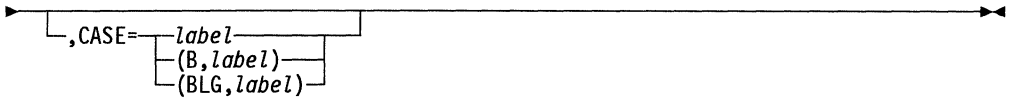
- *keyword1*
- *operator*
- *keyword2*
- *format*
- *logical connective.*

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the label of a case routine and, optionally, whether entry is through use of a B instruction or a BLG extended mnemonic. When you specify the label alone, the branch instructions used in the CASEIF

macro test are used to enter the case routine. If you do not specify this keyword, you can specify a logic operator to create a complex test for entry to the case routine specified on the last CASEIF macro in such a test.

label specifies the label of the case to be executed using the default branch.

(B,label) specifies that a B instruction is to be used to branch to the case routine specified by label.

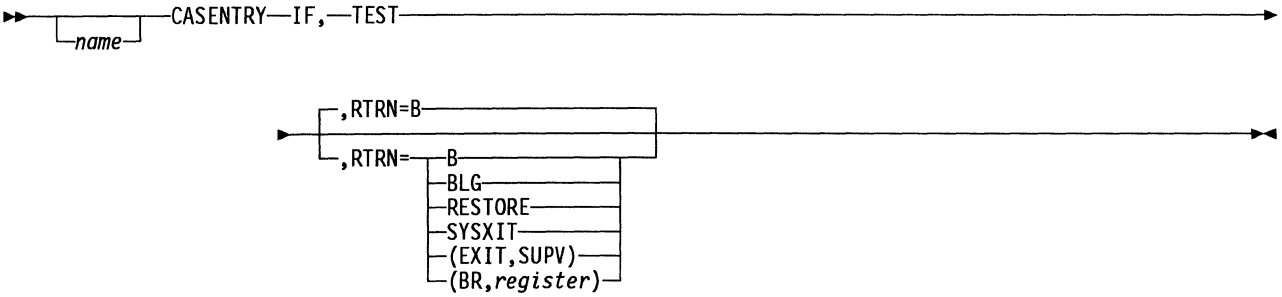
(BLG, label) specifies that a BLG extended mnemonic is to be used to branch to the case routine specified by label.

Remarks You must specify the CASE and logic operator keywords under a single CASEIF macro. The last CASEIF macro in a complex multiple test must have the CASE keyword specified and the logic keyword blank; you cannot have both of these keywords blank.

CASENTRY—Begin Case Selection Criteria

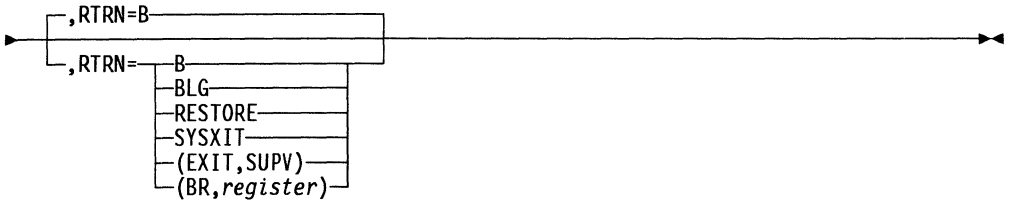
The CASENTRY macro is used with the CASEIF, CASEXIT, CASE, and ENDCASE macros to form a case program structure. The case program structure standardizes the structure of decision logic and makes the program easier to read and maintain. CASENTRY establishes the starting point of CASEIF macro tests used to enter case routines defined by CASE and ENDCASE macros.

Syntax



IF and TEST are required dummy keywords.

Parameters



Function Specifies the return function used when an ENDCASE macro is encountered.

Format B specifies that the B instruction will be used at the end of the case routines.

BLG specifies that the BLG extended mnemonic will be used at the end of the case routines.

RESTORE specifies a RESTORE macro with AREA=NO. Return is through the save area return address. The save area pointer is assumed to be in register 6. Note that CASENTRY does not save the return address for the case structure.

SYSXIT shows that no further processing is required in the current level 5 routine after processing the CASE macros.

(EXIT,SUPV) specifies that no further processing is required in the current interrupt level after processing the case routines. This is not allowed in program level 5. Note that this format makes your code unstructured and must be used carefully.

(BR,*register*) specifies that the return is with a BR instruction, and the return address is in the register specified by *register*.

Default B.

Remarks CASENTRY IF,TEST precedes one or more CASEIF macros that select the case routine to be executed. Each CASE macro has a label, and the appropriate case routine is entered from those CASEIF macros that specify the case routine label.

The CASENTRY macros can be nested and may occur within a CASE (between CASE and ENDCASE).

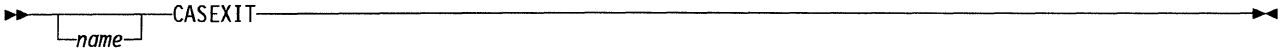
The B or BLG return functions can be overridden by the return type coded on the CASE macro.

Register 0 is not allowed for the register parameter.

CASEXIT—End Case Selection Criteria

The CASEXIT macro is used with the CASEIF, CASENTRY, CASE, and ENDCASE macros to form a case program structure. The CASEXIT macro establishes the end point of CASEIF testing. The end point determines the particular case routine that should be executed. After a case routine is executed, control is returned to the end point specified by CASEXIT, except when the RTRN keyword on the CASENTRY macro specifies (EXIT, SUPV), SYSXIT, RESTORE, or (BR, register).

Syntax



CHAIN—Add a Buffer to a Buffer Chain

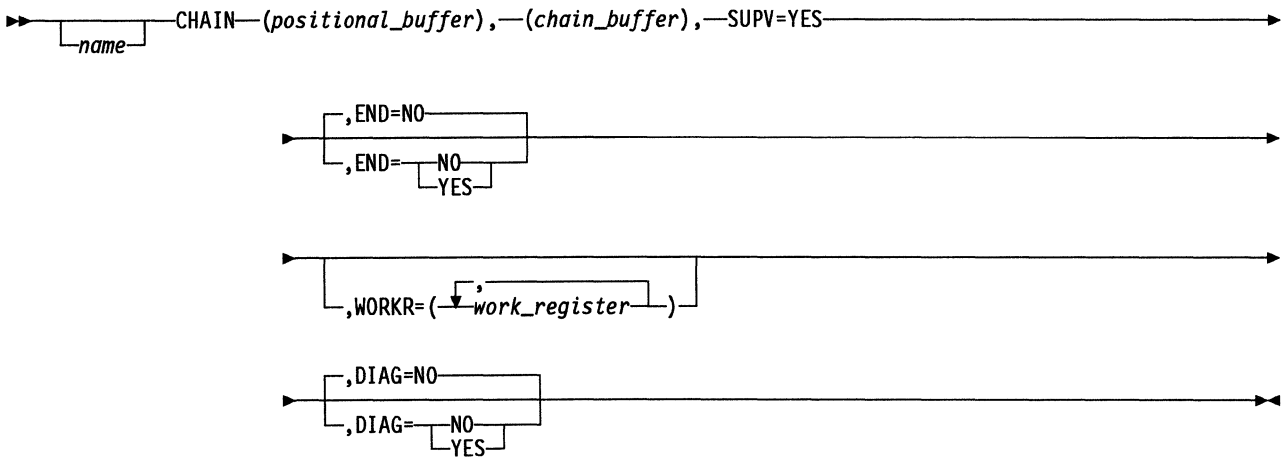
The CHAIN macro connects a buffer to a specified position in an existing buffer chain. This macro has two distinct formats: SUPV=YES denotes that the issuer is operating in an interrupt level, and SUPV=NO denotes that the issuer is operating in level 5.

The macro is sometimes used to connect a chain of buffers to the end of an existing buffer chain. The CHAIN macro cannot, however, add a chain of buffers to the middle of an existing chain.

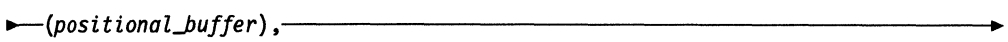
Register 0 is not allowed for register parameters.

SUPV=YES Format (Generates Inline Code)

Syntax



Parameters



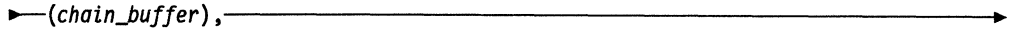
Function Specifies the register containing the address of the buffer to which the new buffer is to be chained.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

This value is not necessarily a pointer to the beginning of a path information unit (PIU), but is a pointer to the actual buffer to which the new buffer should be chained.



Function Specifies the register containing the address of the buffer to be chained.

Format Register notation.

Default None.

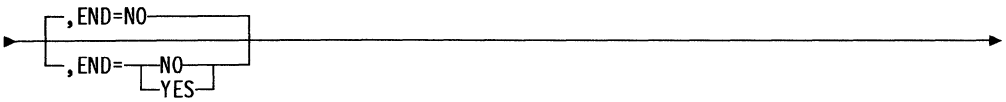
Remarks Register 1 is not allowed

If END=NO, this chained buffer must be a single buffer.

Function Specifies that the issuer is running in an interrupt level.

Format YES.

Default None.



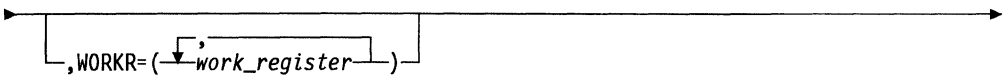
Function Specifies whether the new buffer is a single buffer or a buffer chain that is to be chained to the end of an existing chain.

END=YES specifies that the new buffers will be chained at the end of an existing chain, which must be pointed to by *positional buffer address register*. END=NO specifies that a single buffer is to be chained following a user-specified buffer in the existing chain. The user-specified buffer can be at any position.

Format YES or NO.

Default NO.

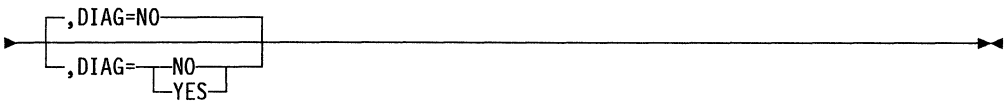
Remarks If END=YES, *positional buffer address register* must point to the last buffer of the chain to which the new buffers are being added.



Function Specifies work registers, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.



Function Specifies whether the buffer address and location will be logged in the ABN diagnostic area if an abend occurs.

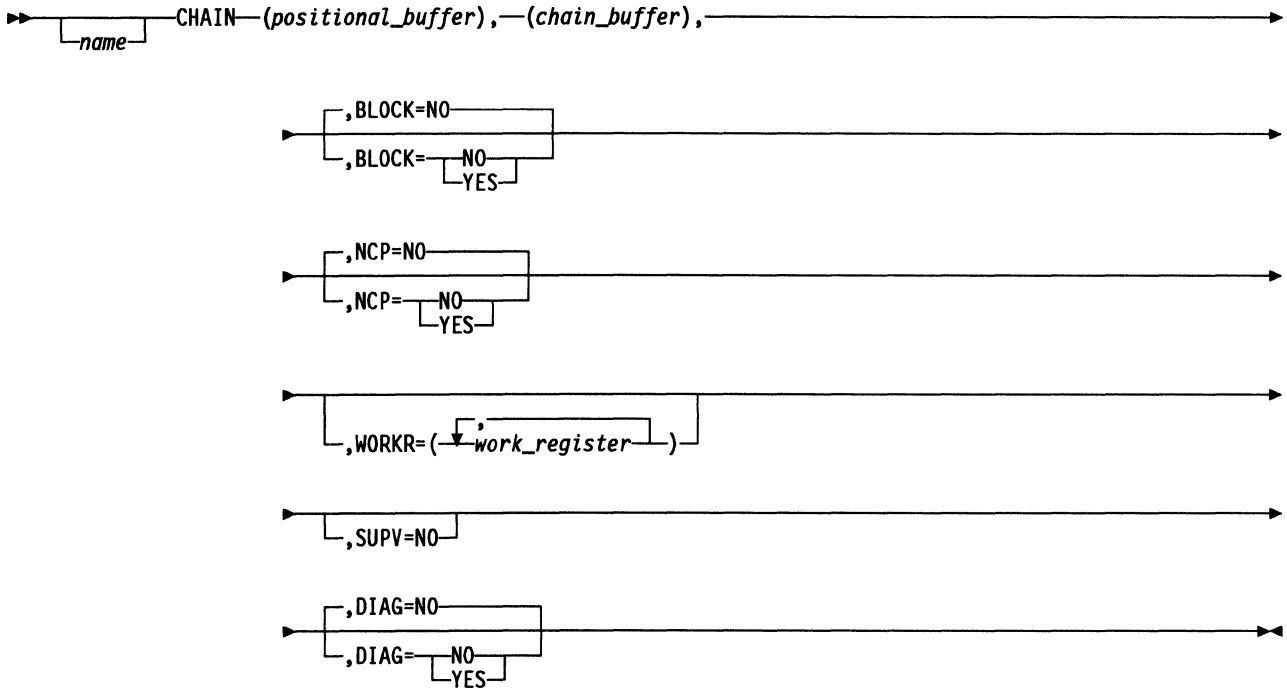
Format YES or NO

Default NO

Remarks If YES is specified, the assembly must include dsect XCXTABN.

SUPV=NO Format

Syntax



Parameters

Function Specifies the register containing the address of the buffer to which the new buffer is to be chained.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

This value is not necessarily a pointer to the beginning of a PIU, but is a pointer to the actual buffer to which the new buffer should be chained.

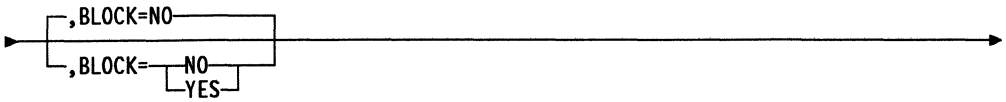
Function Specifies the register containing the address of the buffer to be chained.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

If BLOCK=NO, this chained buffer must be a single buffer.

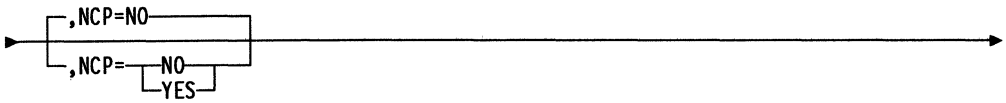


Function Specifies whether the new buffer is a single buffer or a buffer chain that is to be chained to the end of an existing chain.

BLOCK=YES specifies that the new buffer is to be chained at the end of an existing chain. BLOCK=NO specifies that a single buffer is to be chained following a user-specified buffer in the existing chain. The user-specified buffer can be at any position.

Format YES or NO.

Default NO.

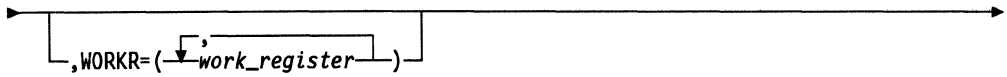


Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES indicates that inline code is generated. NCP=NO indicates that an SVC is generated.

Format YES or NO.

Default NO.

Remarks If NCP=YES, user-written code must be running in storage protection key 0.



Function Specifies work registers, the contents of which may be altered during execution of the macro.

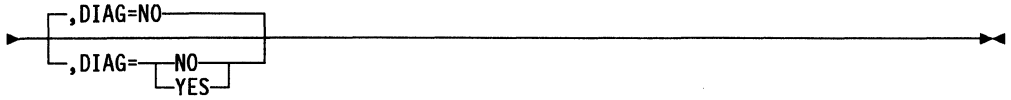
Format Register notation.

Default None.

Function Specifies that the issuer is running in level 5.

Format NO.

Default None.



Function Specifies whether the buffer address and location will be logged in the ABN diagnostic area if an abend occurs.

Format YES or NO

Default NO

Remarks If YES is specified, the assembly must include dsect XCXTABN. If NCP=NO is specified, DIAG=YES is ignored.

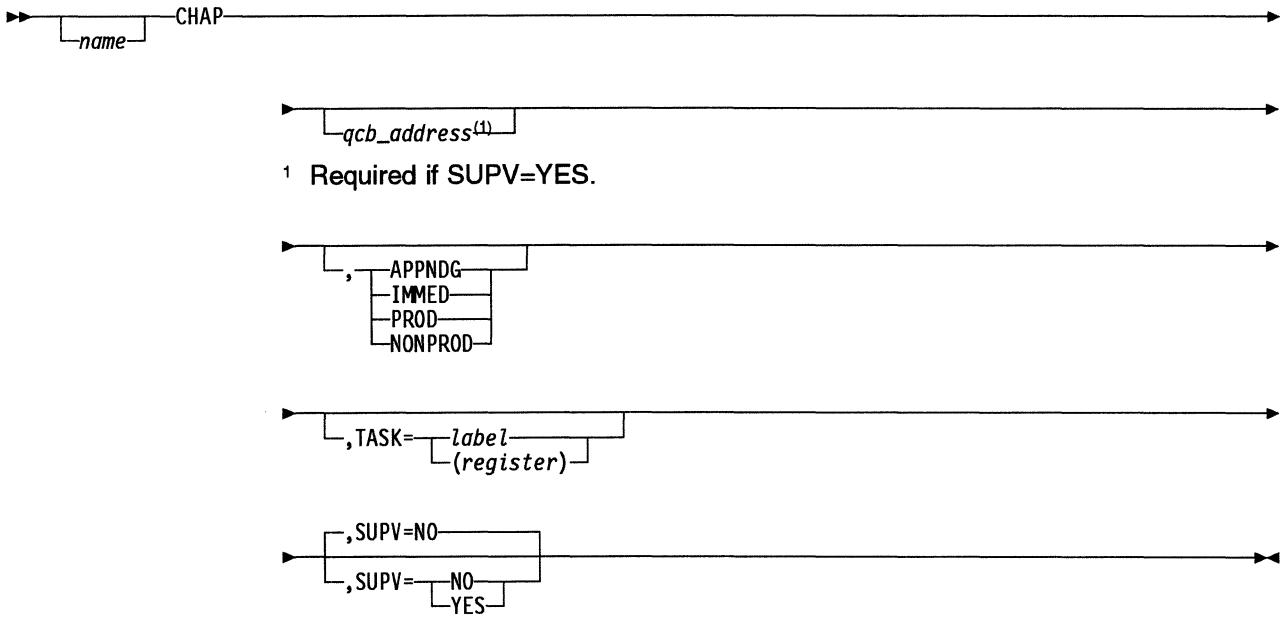
CHAP—Change the Dispatching Priority of a Task

The CHAP macro changes the dispatching priority of a specified task to *appendage*, *immediate*, *productive*, or *nonproductive* and replaces the task address in a specified queue control block (QCB).

If the task is *not* in the pending state when the CHAP macro is issued, its QCB is flagged so that the subsequent activation of the task is appropriately scheduled. If the task is in the pending state, its QCB is flagged and the QCB of the task is removed from its current dispatching queue and put on the appropriate dispatching queue.

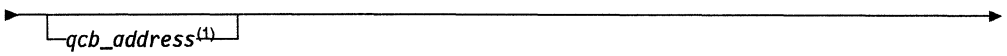
Register 0 is not allowed for register parameters.

Syntax



1 Required if SUPV=YES.

Parameters



1 Required if SUPV=YES.

Function Specifies the address of the input or the pseudo-input QCB governing the task whose priority is to be changed.

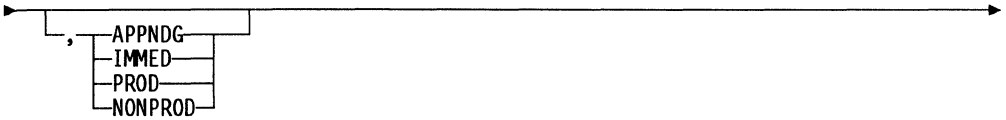
Format Register or label notation.

Default If SUPV=YES, there is no default. If SUPV=NO, the QCB that activated the issuing task is the default.

Remarks Register 1 is not allowed.

If SUPV=YES, register 2 is standard and register 6 is not allowed. The contents of register 1 are destroyed.

If a task is changing its own priority and the QCB address is omitted, the address of the QCB that started the task is used.

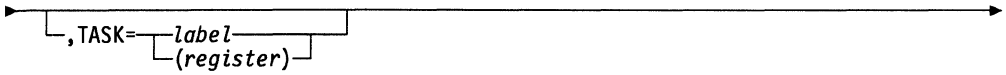


Function Specifies the change to the dispatching priority as shown below:

APPNDG	Appendage
IMMED	Immediate
PROD	Productive
NONPROD	Nonproductive.

Format APPNDG, IMMED, PROD, or NONPROD.

Default The dispatching priority of the specified task remains unchanged.



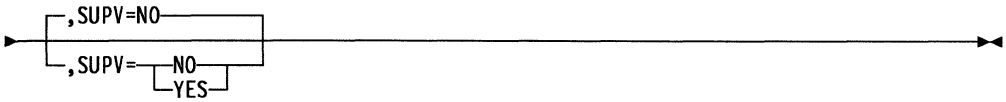
Function Specifies a replacement for the task address in the specified QCB.

Format Register or label notation.

Default None.

Remarks Register 1 is not allowed. Register 2 is standard. If SUPV=YES, register 6 is not allowed and register 3 is standard.

The task address can be any valid task in any valid QCB.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

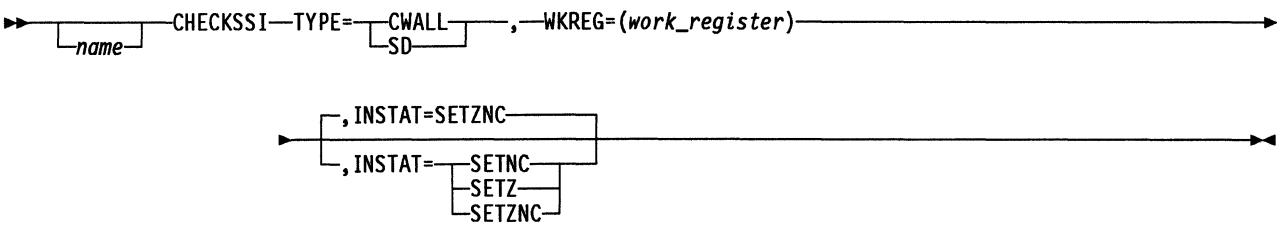
CHECKSSI—Check the Status of the NCP Buffer Pool

The CHECKSSI macro checks the status of the NCP buffer pool to determine whether NCP is in slowdown, or whether the number of buffers remaining is below the CWALL value.

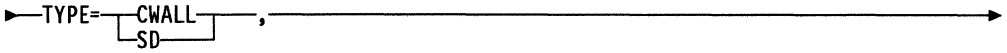
The CHECKSSI macro generates inline code.

Note: You must invoke the XXCXTXDB macro in the module that issues the CHECKSSI macro.

Syntax



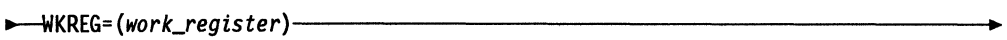
Parameters



Function Specifies whether the buffer availability check is for the *in slowdown* state (SD) or the *below CWALL* state (CWALL).

Format SD or CWALL.

Default None.

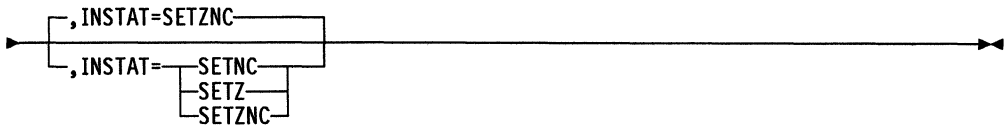


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Absolute register notation.

Default None.

Remarks Register 0 is not allowed. Specify an odd-numbered register only.



Function Specifies whether the C latch, the Z latch, or both are to indicate the state of buffer availability.

Format SETZ, SETNC, or SETZNC.

Default SETZNC.

Remarks For SETZ or SETZNC, the Z latch is set to 1 if the state checked for exists. For SETNC or SETZNC, the C latch is set to 0 if the state checked for exists.

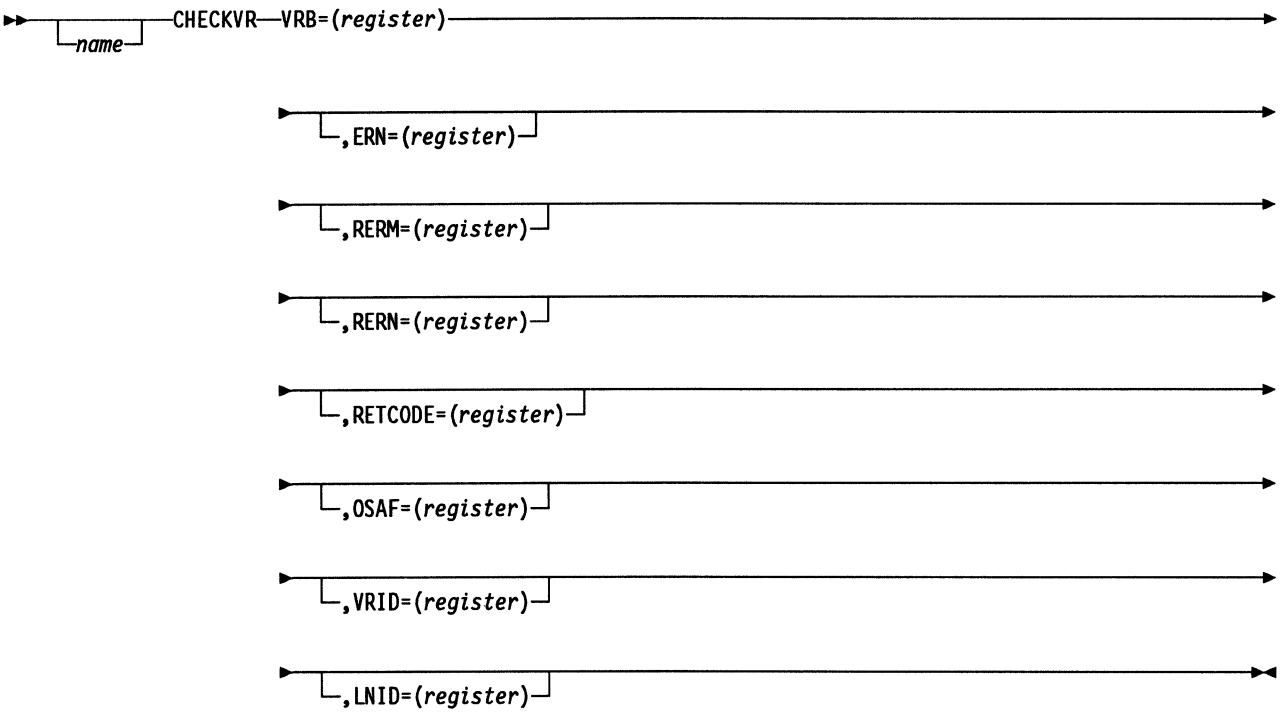
CHECKVR—Return VRB Fields Using the Virtual Route Control Block

The CHECKVR macro uses the virtual route control block (VRB) address to return fields in the VRB. Any combination of outputs can be requested.

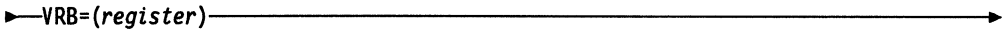
Register 0 is not allowed for register parameters.

Note: You must invoke the VRB DSECT macro XCXTVRB in the routine that issues the CHECKVR macro.

Syntax



Parameters

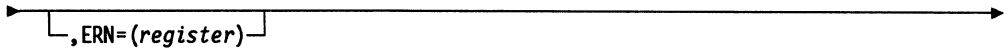


Function Specifies the VRB address of the virtual route to be checked.

Format Absolute register notation.

Default None.

Remarks The register you specify for the VRB address must not be the same as the register specified for any other keyword.



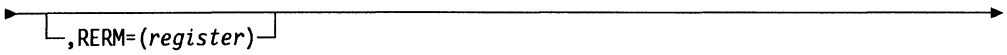
Function Specifies the explicit route number (ERN) associated with this VRB.

Format Absolute register notation.

Default None.

Remarks The register must be a byte register.

The byte register you specify must not be the same as the register specified for LNID, RERM, RERN, or VRID. The register you specify must not be the same as any of the fullword registers specified for any other keyword.



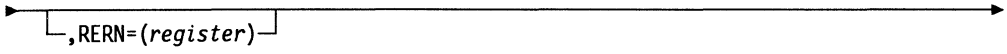
Function Specifies the reverse explicit route mask associated with this VRB.

Format Absolute register notation.

Default None.

Remarks The register must be a fullword register.

The register you specify must not be the same as the register specified for ERN, LNID, RERN, or VRID. The register specified must not be the same as any of the fullword registers specified on any of the other keywords of this macro.

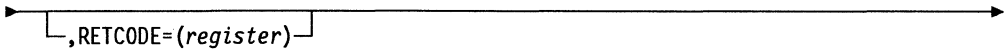


Function Specifies the reverse ERN associated with this VRB.

Format Absolute register notation.

Default None.

Remarks The register must be a byte register. The register you specify must not be the same as the register specified for ERN, LNID, or VRID. The register you specify must not be the same as any of the fullword registers specified on any of the other keywords of this macro.



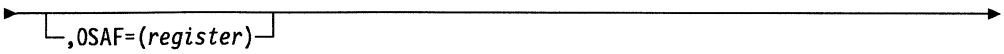
Function Specifies the register to receive the 2-byte flag field of the VRB. Refer to the description of the VRB in *NCP and EP Reference Summary and Data Areas*, Volume 1, for bit definitions.

Format Absolute register notation.

Default None.

Remarks The register must be a fullword register.

The register you specify for RETCODE must not be the same as the register specified for any other keyword.



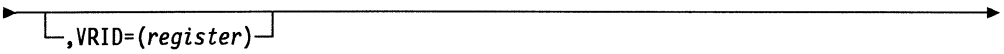
Function Specifies the register to receive the subarea address of the other end of the virtual route.

Format Absolute register notation.

Default None.

Remarks The register must be a fullword register.

The register you specify for OSAF must not be the same as the register specified for any other keyword.



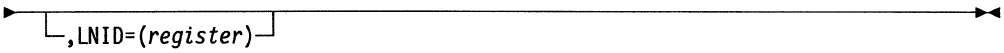
Function Specifies the virtual route identifier (VRID) associated with the required virtual route.

Format Absolute register notation.

Default None.

Remarks The register you specify must be a byte register.

The byte register you specify must not be the same as the register specified for ERN, LNID, or RERN. The register you specify must not be any of the fullword registers specified on any of the other keywords of this macro.



Function Specifies the local network identification (LNID) of the network in which the virtual route resides.

Format Absolute register notation.

Default None.

Remarks The register must be a byte register.

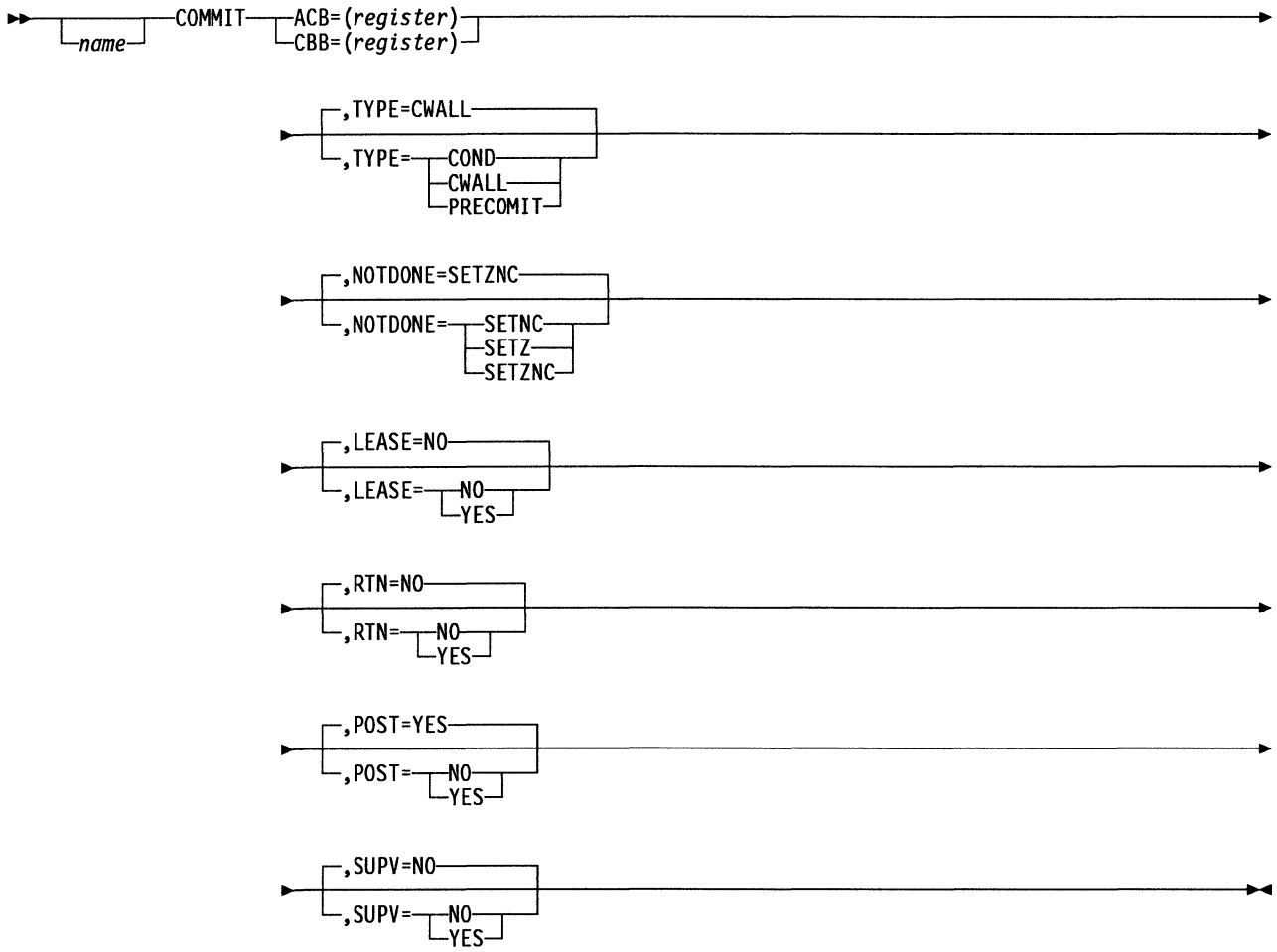
The register you specify must not be the same as the register specified for ERN, RERN or VRID. The register you specify must not be any of the fullword registers specified on any of the other keywords of this macro.

COMMIT—Commit Buffers

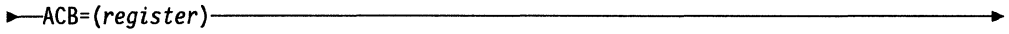
The COMMIT macro determines whether enough buffers are available to schedule a poll and receive the resulting input.

Note: This macro uses but does not destroy register 2 and, if NOTDONE is specified, register 3. Register 6 must point to a save area. Register 0 is not allowed.

Syntax



Parameters

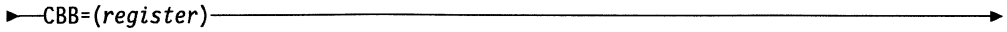


Function Specifies the address of the adapter control block (ACB) containing the buffer commitment request.

Format Register notation.

Default None.

Remarks ACB and CBB may not be coded together.

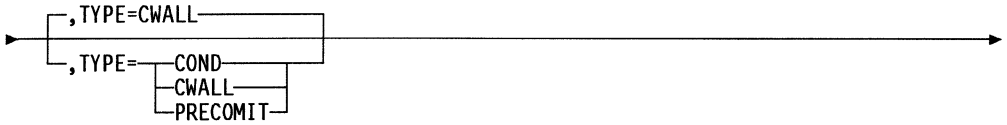


Function Specifies the address of the committed buffers block (CBB) containing the buffer commitment request.

Format Register notation.

Default None.

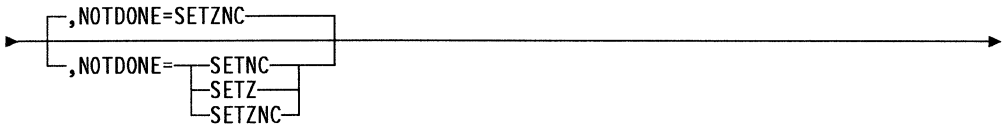
Remarks ACB and CBB may not be coded together.



Function Specifies whether the commitment is conditional (COND) or unconditional (CWALL) or whether a check on a commitment in progress (PRECOMIT) is being made.

Format COND, CWALL, or PRECOMIT.

Default CWALL.



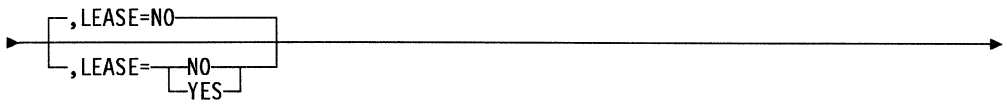
Function Specifies whether the C latch, the Z latch, or both are to indicate whether the request was satisfied.

Format SETZ, SETNC, or SETZNC.

Default SETZNC.

Remarks For SETZ or SETZNC, the Z latch is set to 1 if the commitment is unsatisfied or to 0 if satisfied. For SETNC or SETZNC, the C latch is set to 0 if the commitment is unsatisfied or to 1 if satisfied.

This keyword is valid only when SUPV=NO.

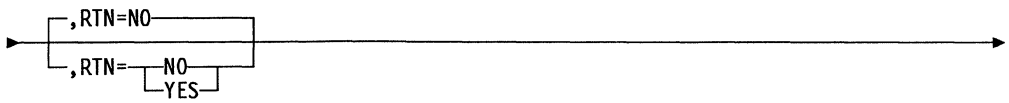


Function Specifies whether this macro should lease the buffers and chain them to the CBB when the commitment request is satisfied.

Format YES or NO.

Default NO.

Remarks LEASE=YES is valid only when CBB is coded.

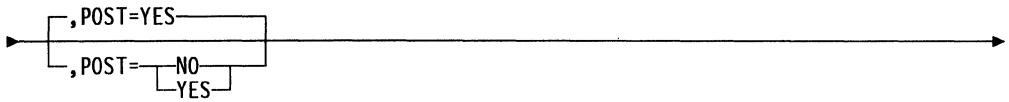


Function Specifies whether a routine address has been stored in the CBB. This routine will get control if the commitment request is posted and satisfied later.

Format YES or NO.

Default NO.

Remarks RTN=YES is not valid if ACB is coded or if POST=NO.

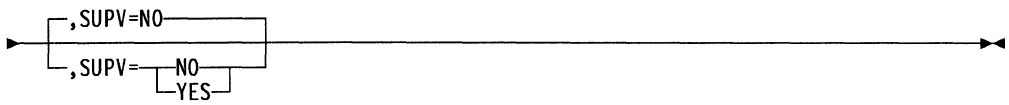


Function Specifies whether to put an unsatisfied commitment request on the appropriate ACHAIN of the commit request block (CRB).

Format YES or NO.

Default YES.

Remarks This keyword is ignored if TYPE=PRECOMIT.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

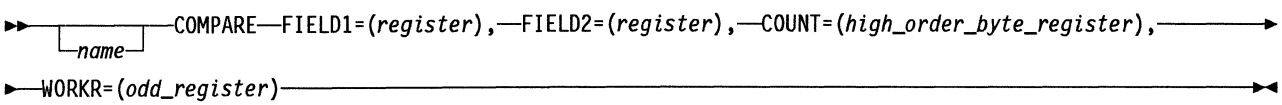
COMPARE—Compare Data in Two Storage Locations

The COMPARE macro allows you to test data in two storage locations to determine whether the contents of the specified locations are equal or, if unequal, which is greater. After execution of the macro, the result register will contain X'00' if the data is equal or X'01' or X'02' if the data is unequal. For comparison, the storage contents are considered to be unsigned binary integers; however, storage fields containing the EBCDIC encoding of alphanumeric characters may also be compared, and the field with the greater alphabetical value (Z is greater than A) will have the greater value. The DSECT XCXTEQU is necessary for the proper expansion of the COMPARE macro.

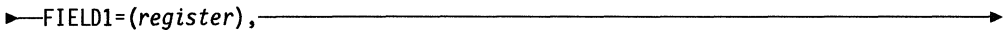
Do not use the COMPARE macro between portions of a connected IF macro (IF R1,EQ,R2,AND...) because the COMPARE macro includes an IF macro, and therefore will assemble incorrectly.

Register 0 is not allowed for register parameters.

Syntax



Parameters



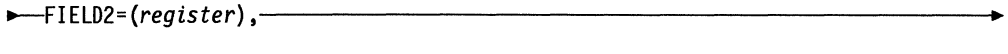
Function Specifies the register that contains a pointer to the beginning of the first item for comparison.

Format Absolute register notation.

Default None.

Remarks A full register is required.

Contents of this register are changed by the macro.



Function Specifies the register that contains a pointer to the beginning of the second item for comparison.

Format Absolute register notation.

Default None.

Remarks A full register is required.

Contents of this register are changed by the macro.

```
►──COUNT=(high_order_byte_register),──────────────────────────────►
```

Function Specifies the register that contains the number of bytes to be compared (the number of bytes in each field). After the macro is executed, this register contains the result of the comparison, as shown in the following list:

- X'00' if the field contents for comparison are equal
- X'01' if the contents of FIELD1 are numerically less than the contents of FIELD2
- X'02' if the contents of FIELD2 are numerically less than the contents of FIELD1.

Format Absolute register notation with high-order byte specification. (You must include 0.)

Default None.

Remarks The high-order byte of an odd-numbered register is required.

The maximum field length is 255.

Overlapping fields are handled normally.

```
►──WORKR=(odd_register)──────────────────────────────────────────►◄
```

Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Absolute register notation.

Default None.

Remarks An odd-numbered full register is required.

The work register must not be the same as the field or count register.

CONVRT—Convert an Explicit Route Mask or Number

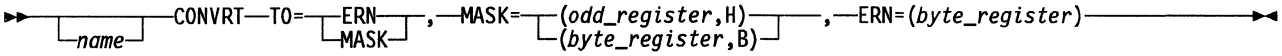
The CONVRT macro generates inline code. It converts an explicit route mask to an explicit route number (ERN) or an ERN to an explicit route mask. For example:

- B'10000000 00000000' converts to X'00'.
- B'01000000 00000000' converts to X'01'.
- B'00100000 00000000' converts to X'02'.
- X'00' converts to B'10000000 00000000'.
- X'01' converts to B'01000000 00000000'.
- X'02' converts to B'00100000 00000000'.

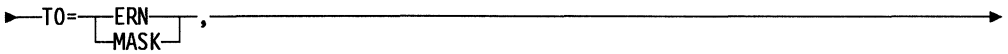
The left-most bit set to 1 is the only one considered by the macro. No conversion is done on a mask of all zeros. If you code (*odd register,H*), an ERN greater than 15 is converted to a mask of all zeros. If you code (*byte register,B*), an ERN greater than 7 is converted to a mask of all zeros.

Register 0 is not allowed for register parameters.

Syntax



Parameters

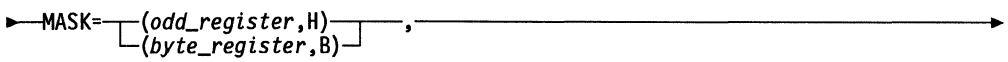


Function Specifies the direction of conversion. TO=ERN specifies mask-to-ERN conversion. TO=MASK specifies ERN-to-mask conversion.

Format ERN or MASK.

Default None.

Remarks If TO=MASK, the ERN must be in byte 0 of the register specified by the ERN keyword.



Function Specifies the register that contains the mask value or specifies the register in which the mask value is to be returned. Use the (*odd register*,**H**) keyword if you are converting an ERN greater than 7, because you will require a halfword register. Use the (*byte register*,**B**) keyword if you are converting an ERN of 7 or less, because you will require only a byte register.

Format Absolute or label register notation.

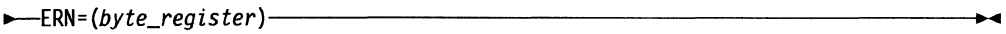
Default None.

Remarks If TO=ERN and the mask register contains 0, no conversion is made.

If TO=MASK and MASK=(*odd register*,**H**), a mask value of 0 is returned if the ERN is greater than 15.

If TO=MASK and MASK=(*byte register*,**B**) a mask value of 0 is returned if the ERN is greater than 7.

If TO=ERN, and the mask register does not contain 0, the mask register elements are changed by the CONVRT macro. If TO=ERN and the mask register contains 0, no conversion is made.



Function Specifies the byte register that contains the ERN or in which the ERN is to be returned.

Format Absolute or label register notation.

Default None.

Remarks The byte register specified cannot be a byte of the MASK register.

If TO=ERN and the MASK register contains 0, the ERN register is not modified.

If TO=MASK, the ERN must be in a high byte register (byte 0).

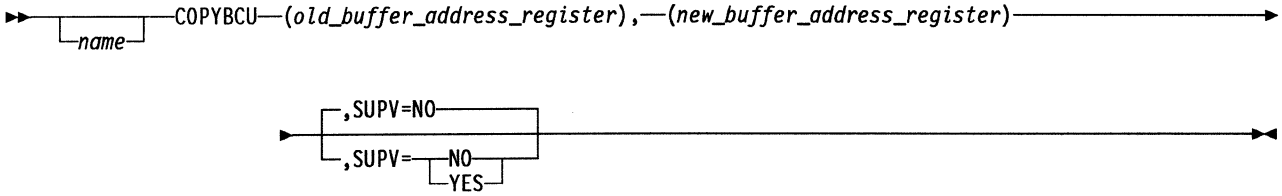
If TO=MASK, the ERN register contents are changed by the CONVRT macro, with the exception that the byte register specified cannot be a byte of the mask register.

COPYBCU—Copy the Block Control Unit Work Area

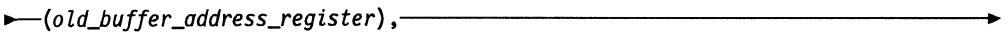
The COPYBCU macro causes the supervisor to copy the block control unit work area and the 14 bytes of basic transmission unit (BTU) control information from one buffer into a second buffer. That is, it copies 26 bytes, beginning at a displacement of +20, into the original buffer.

Register 0 is not allowed for register parameters.

Syntax



Parameters



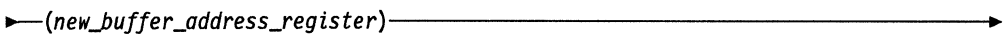
Function Specifies the register containing the address of the buffer containing the BTU to be copied.

Format Register format.

Default None.

Remarks Register 1 is not allowed.

If SUPV=YES, register 6 is not allowed, and register 3 is standard.



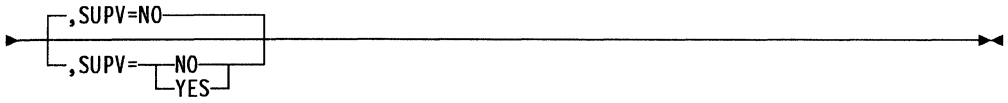
Function Specifies the register containing the address of the buffer into which the BTU is to be copied.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

If SUPV=YES, register 6 is not allowed, and register 4 is standard.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

COPYPIU—Copy a PIU from One Buffer to Another

If the LEASE keyword is coded NO, the COPYPIU macro copies a FID0 or FID1 path information unit (PIU) from one buffer into another. The fields copied are:

- Resource type field from the unit information block (UIB)
- UIB status
- Transmission header (TH)
- Request/response header (RH)
- Request/response unit (RU) if the RU keyword is specified.

All the bytes to be copied must be in the buffer pointed to by the old buffer address. After the fields are copied, the new PIU buffer prefix offset field points to the beginning of the transmission header; the PIU prefix count field shows the length of the TH+RH+RU; and the TH count field shows the length of the RH+RU. The event control block (ECB) block control unit (BCU) status field is set to 0 if specified by the ECBINIT keyword.

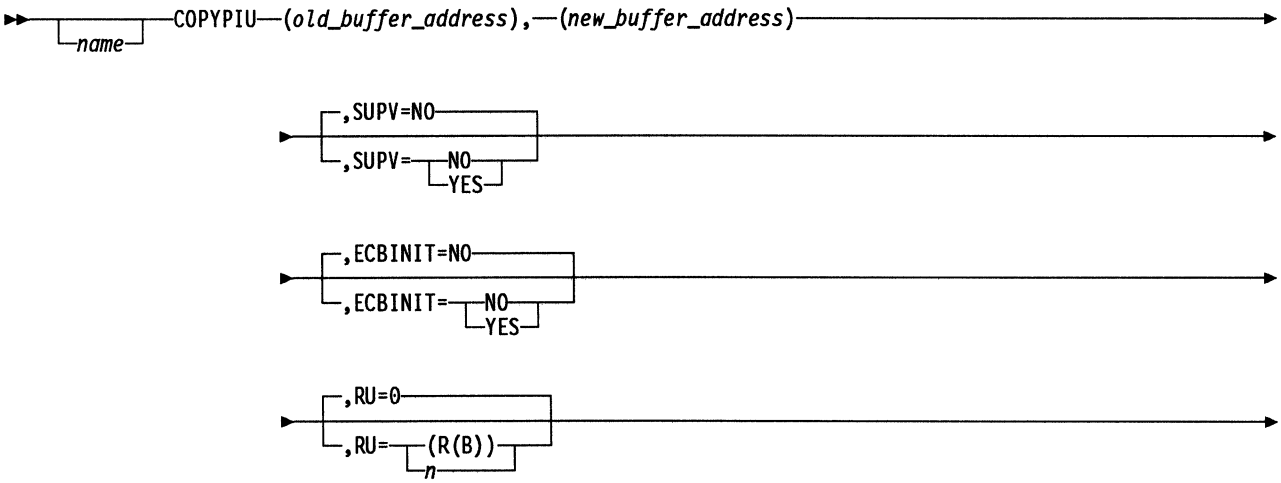
If the LEASE keyword is coded YES, the service routine supplies the new buffers necessary to make a copy. The old PIU may be more than one buffer and does not have to be in the buffer pool. Omit the RU keyword because the complete RU will be copied.

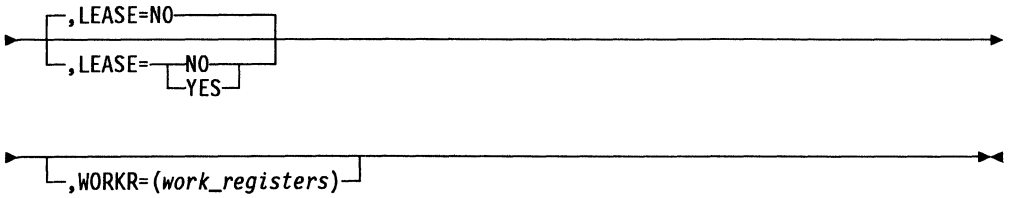
When LEASE=YES, the COPYPIU macro overwrites the first buffer of the original PIU with a pointer to the new PIU. Fields in bytes 0 to 15 are overlaid. Bytes 16 to 19 contain a pointer to the new PIU.

Register 0 is not allowed for register parameters.

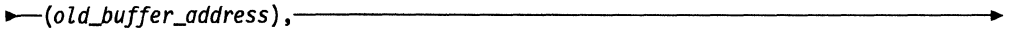
Note: If LEASE=NO, the COPYPIU macro produces code that may alter the contents of register 1.

Syntax





Parameters



Function Specifies the register containing the buffer from which the PIU is to be copied.

Format Register notation.

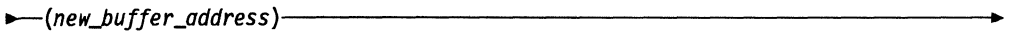
Default None.

Remarks Register 3 is standard.

Register 1 is not allowed.

If SUPV=YES, register 6 is not allowed.

Old and new buffers must not be specified by the same register.



Function Specifies the register containing the buffer into which the PIU is to be copied.

Format Register notation.

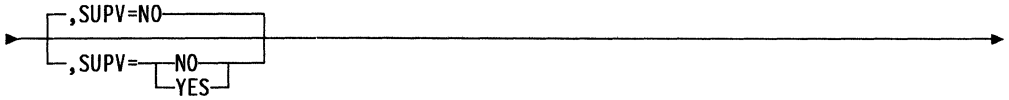
Default None.

Remarks Register 4 is standard.

Register 1 is not allowed.

If SUPV=YES, register 6 is not allowed.

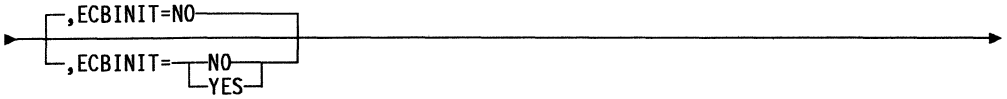
Old and new buffers must not be specified by the same register.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

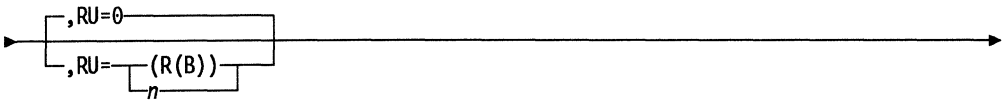


Function Specifies whether the contents of the ECB status field is to be set to 0.

Format YES or NO

Default: NO, if LEASE=NO.

Remarks If LEASE=YES, omit this keyword.



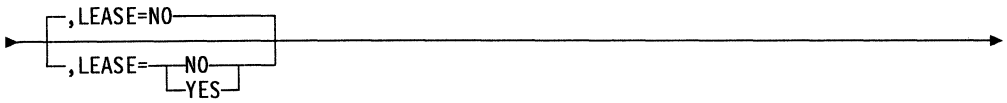
Function Specifies the number of bytes in the RU to be copied.

Format *n* indicates the number of bytes (decimal) to be copied; (R(B)) is byte register notation indicating where the length value is located; and 0 indicates that the RU is not to be copied.

Default 0, if LEASE=NO.

Remarks If LEASE=YES, omit this keyword.

Register 1 (byte 0) is the preferred register and the contents are destroyed if not specified.



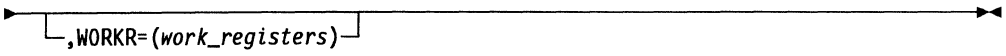
Function Shows whether the necessary buffers are to be leased by the service routine, or if the new buffer is supplied by the user.

Format YES or NO.

Default NO.

Remarks If LEASE=YES and there are not enough buffers in the free buffer pool to make a copy, nothing is copied and the new buffer address is 0. If LEASE=NO, both the old and the new buffer addresses must be in the buffer pool.

If LEASE=NO, only one buffer is copied.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Registers 1, 2, 3, 4, 5, and 7 (up to a maximum of six registers) can be specified using register notation. Do not use equated values.

Default None.

Remarks Specifying WORKR=1, 3, 4, or 7 can save controller machine cycles.

WORKR=2 or WORKR=5 is ignored.

CXTSVX—Build a Save Area Chain

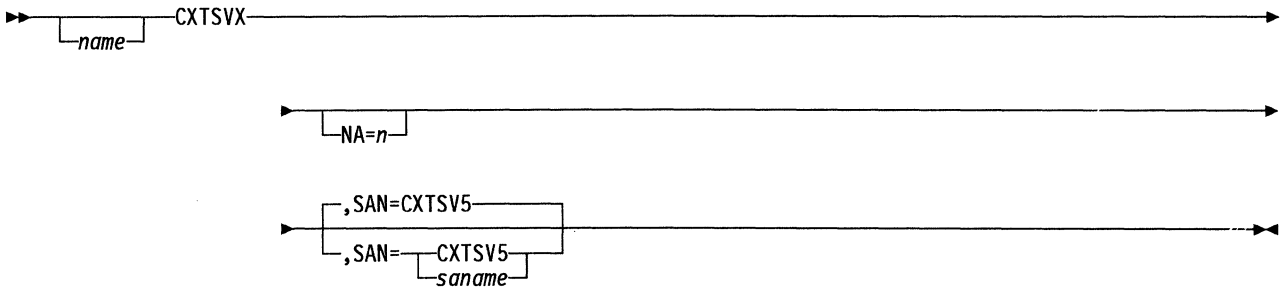
The CXTSVX macro builds a save area chain. It provides the save area chain structure for registers saved by the ROUTINE macro. The CXTSVX macro also provides the save area for the return linkage to the instruction following the PERFORM macro in the PERFORM, ROUTINE, RETURN macro sequence.

Code the CXTSVX macro in program level 4 supervisor routines to save storage by creating common save areas and then providing linkage to the program level 5 background routines that use them.

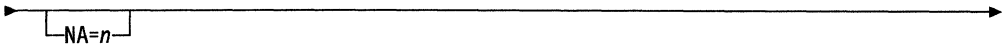
The level 4 dispatcher initially sets up register 6 in program level 5 to point to the first save area at CXTSV5. Register 6 must be maintained by any module using the PERFORM, ROUTINE, RETURN macro sequence, its counterpart macro sequence LINK, SAVE, RESTORE, and the CASENTRY RTRN=RESTORE macro. Register 6 is updated to point to the correct save area by these macros unless overridden.

The save area chain structure built by the CXTSVX macro can also be used by the LINK, SAVE, RESTORE macro sequence. The save areas are in storage that has a storage-protection key of 1; they are accessible to user routines.

Syntax



Parameters

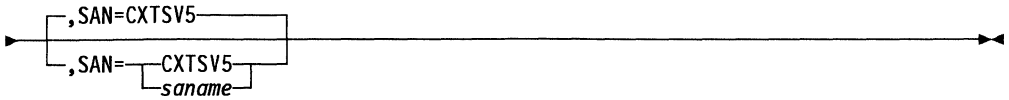


Function Specifies the number of save areas to be built in the save area chain.

Format Decimal values 3 to 255.

Default 7.

Remarks The value *n* is not checked. A value of 3 or less will result in three save areas in the chain.



Function Provides an entry-point name for the save area chain.

Format Six valid characters.

Default CXTSV5.

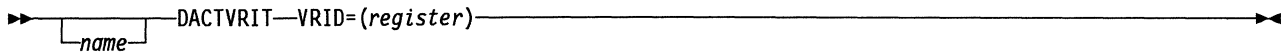
DACTVRIT—Deactivate an Internal Virtual Route

The DACTVRIT macro generates a supervisor call instruction (SVC). It is used to deactivate an internal virtual route.

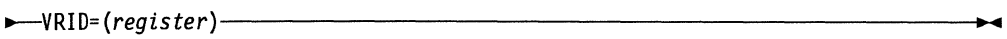
Note that you can use this macro only in level 5.

Warning: When a virtual route is deactivated, any sessions that are still active on the virtual route are deactivated. Therefore, users who are writing their own routines are encouraged not to share virtual routes. (Refer to the description of the ACTVRIT macro.)

Syntax



Parameters



Function Specifies the virtual route identification number (VRID). The VRID has two values, the virtual route number (VRN) and the transmission priority field (TPF). The VRN value can be 0 to 7 and must be right-justified in byte 0 of the specified register. The TPF value can be 0 to 2 and must be right-justified in byte 1 of the specified register.

Format Absolute register notation.

Default None.

Remarks Registers 0 and 1 are not allowed.

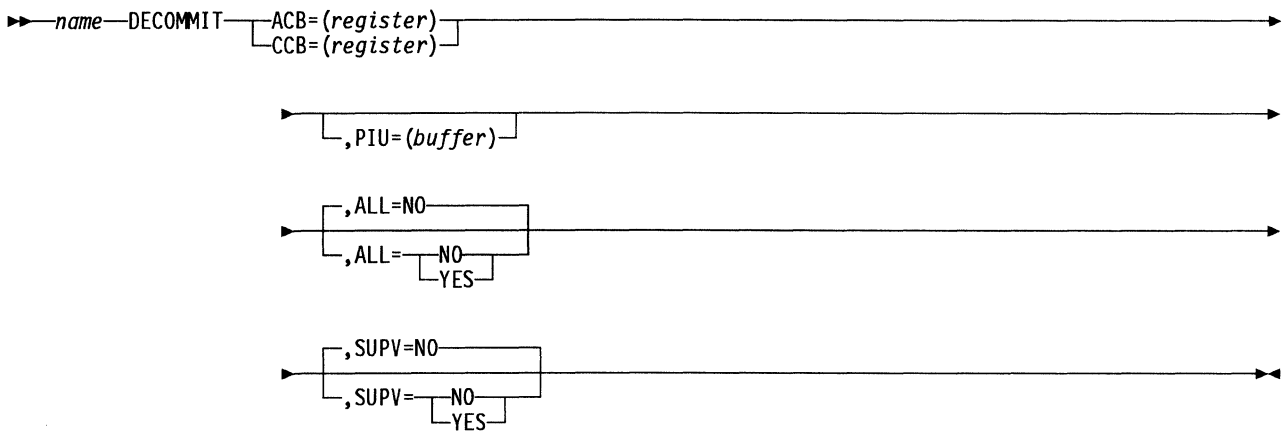
No checking is done to see that the VRN and TPF values are valid.

DECOMMIT—Request That a Buffer Commitment Be Unassigned

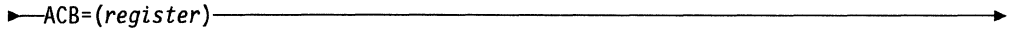
The DECOMMIT macro requests that a buffer commitment be unassigned.

Register 0 is not allowed for register parameters.

Syntax



Parameters

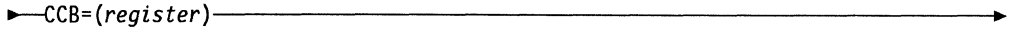


Function Specifies the address of the adapter control block (ACB) containing the buffer commitment request to be decommitted.

Format Register notation.

Default None.

Remarks ACB and CBB may not be coded together.

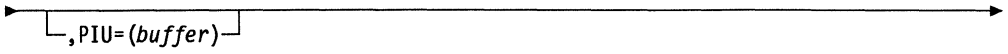


Function Specifies the address of the committed buffers block (CBB) containing the buffer commitment request to be decommitted.

Format Register notation.

Default None.

Remarks ACB and CBB may not be coded together.

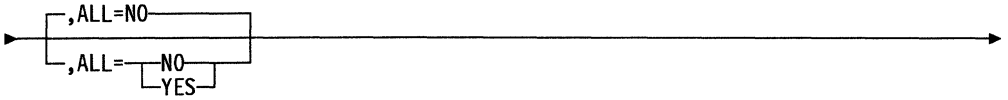


Function Specifies the address of the input buffer chain. The number of buffers to be decommitted is equivalent to the number of buffers in the chain.

Format Register notation.

Default None.

Remarks The receive buffer count is incremented by the number of buffers in the chain.



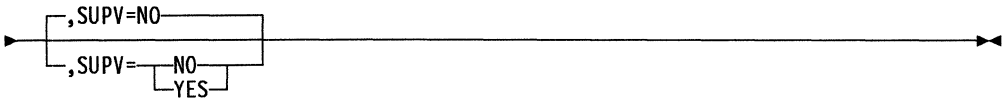
Function Specifies whether the complete current buffer commitment is to be unassigned.

Format YES or NO.

Default NO.

Remarks If you specify YES, the next commitment size is calculated.

You must specify YES if you omit the path information unit (PIU) keyword.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

DEFMSK—Generate a Bit-Mask Symbol and a Conditional Branch Symbol

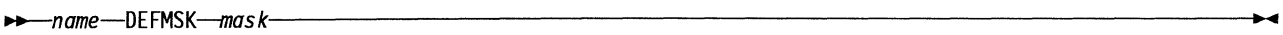
The DEFMSK macro generates a bit-mask symbol and a conditional-branch symbol. The mask value shows the position of the bit in an 8- or 16-bit field, starting with 0. For the bit-mask symbol, the DEFMSK macro inserts bit 1 in the byte or halfword bit position that corresponds to the value of the mask. For example, the bit-mask symbol has a configuration of B'0001 0000' for a bit position of 3 and B'0000 0000 0010 0000' for a bit position of decimal 10. The conditional-branch symbol has the value of the specified mask.

The DEFMSK macro generates both symbols when the mask is specified by decimal or IRNBITn notation, but generates only the bit-mask symbol when the mask is specified by hexadecimal (X'FE', for example) or binary notation (B'1111 1110', for example).

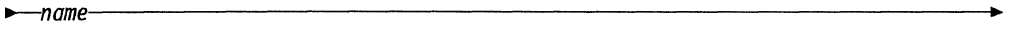
The conditional-branch symbol is used to code the M field of the branch on bit (BB) instruction, the P field of the branch on bit extended (BBE) assembler extended mnemonic code, and in the IF and CASEIF macros.

The bit-mask symbol is used to code the I field of the test register under mask (TRM) instruction and as a bit set/reset mask.

Syntax



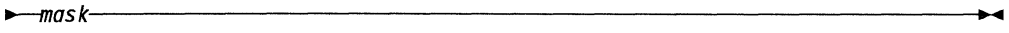
Parameters



Function The DEFMSK macro generates the bit-mask symbol by concatenating the character *M* to the end of the specified name. The DEFMSK macro generates the conditional-branch symbol by concatenating the character *B* to the end of the specified name.

Format Any symbol of 7 or fewer characters.

Default None.



Function Specifies the value of the mask.

Format Decimal values 0 through 15, IRNBITn where *n* can be any decimal value 0 to 15, binary notation, or hexadecimal notation.

Default None.

Remarks This keyword may be followed by up to 255 characters of comments, separated from the mask by a comma, and enclosed in single quotation marks.

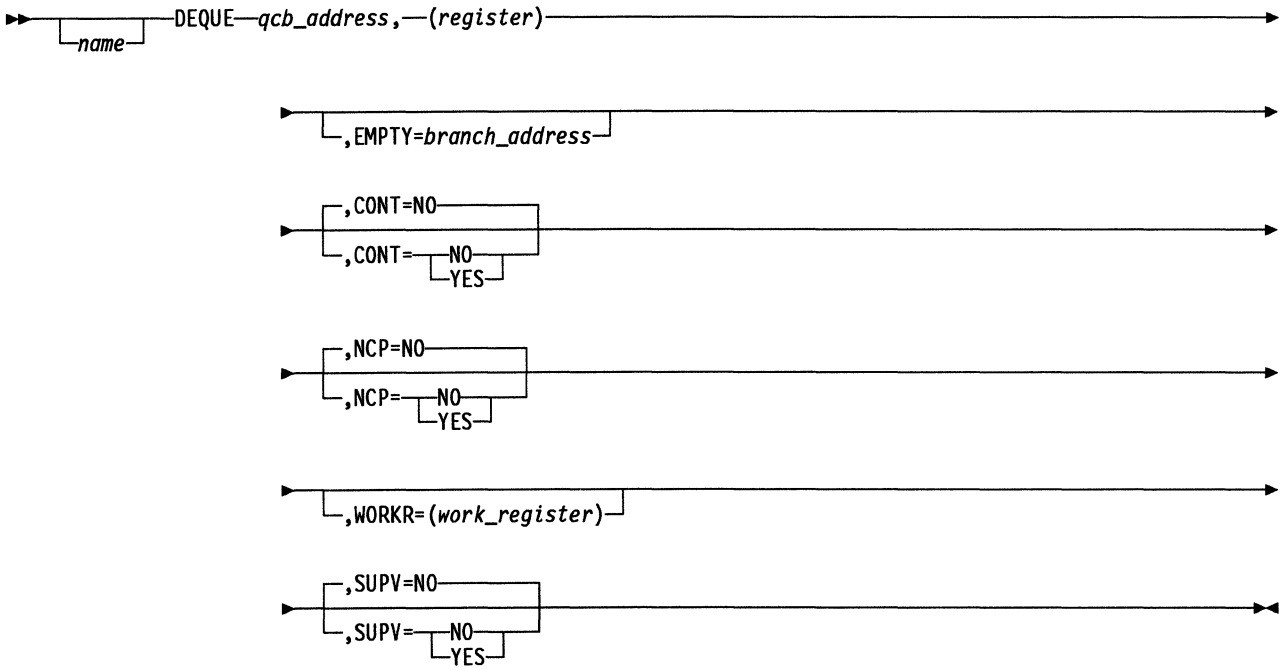
DEQUE—Locate and Unchain the First Element in a Queue

The DEQUE macro locates and unchains the first element, which can be the block control unit (BCU), the path information unit (PIU), or the queue control block (QCB), in a specified system queue and returns a pointer to the element.

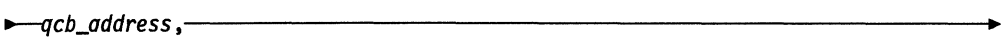
Register 0 is not allowed for register parameters.

Note: Use the EXTRACT macro to detach any element, except the first, in a system queue.

Syntax



Parameters



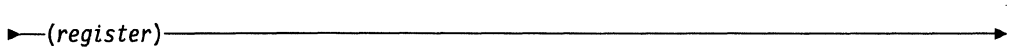
Function Specifies the address of the QCB governing the queue from which the element is to be removed.

Format Register or label notation.

Default If SUPV=YES, there is no default. If SUPV=NO, the QCB that activated the issuing task is the default.

Remarks Register 1 is not allowed.

If SUPV=YES, register 2 is standard, and register 6 is not allowed.



Function Specifies the register containing the address of the element to be located and dequeued.

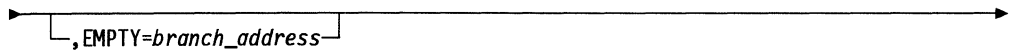
Format Register notation.

Default None.

Remarks Register 1 is not allowed. Register 3 is standard.

If SUPV=YES, register 6 is not allowed.

The contents of the register are set to 0 if the specified queue is empty.



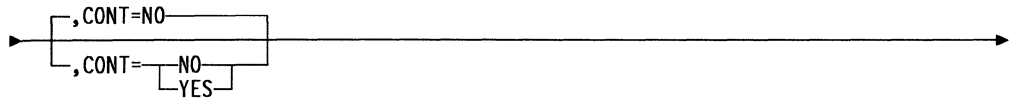
Function Specifies the address to get control if the queue is empty.

Format Register or label notation.

Default The instruction following the macro is given control.

Remarks Register 1 is not allowed.

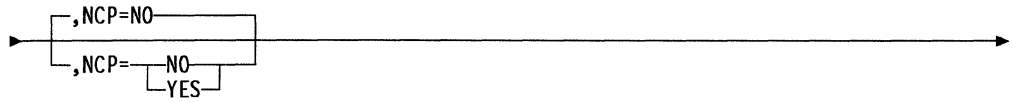
If SUPV=YES, register 6 is not allowed.



Function Specifies whether the queue is a contention queue (one that can be simultaneously manipulated by two or more interrupt levels).

Format YES or NO.

Default NO.



Function Specifies whether the routine is to be generated inline.

Format YES or NO.

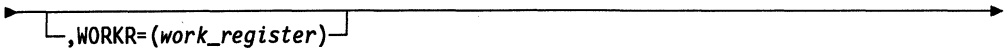
Default NO.

Remarks If NCP=YES, you must specify WORKR.

If NCP=YES, specify CONT=NO (either by specification or by default).

If NCP=YES, the module must copy the DSECTs for the QCB and the event control block (ECB).

If NCP=YES and SUPV=NO, you must specify the QCB address.

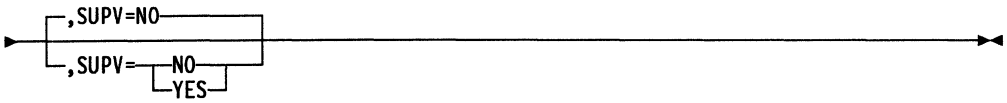


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register specified must be an odd-numbered register.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

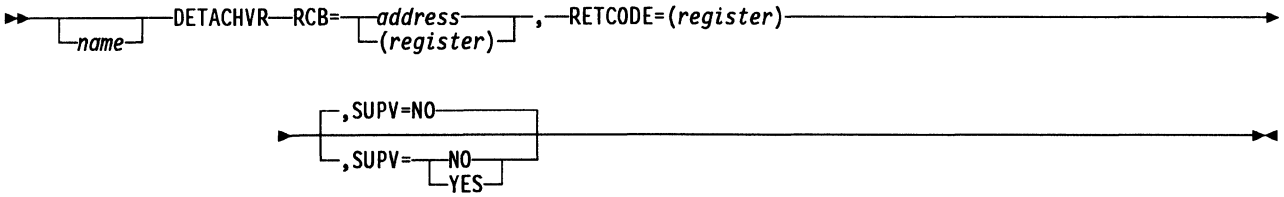
Default NO.

DETACHVR—Detach a Resource Control Block from a Virtual Route

The DETACHVR macro terminates the association between a resource connection control block (RCB) and a virtual route.

Register 0 is not allowed for register parameters.

Syntax



Parameters

Function Specifies the register containing the address of the RCB to be disassociated from its virtual route.

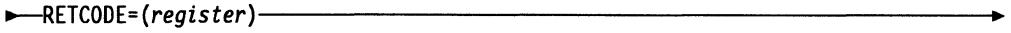
Format Register or the label or name of the RCB itself.

Default None.

Remarks The register specified for RCB must not be the same as the register specified for RETCODE.

For best performance, you should specify register 2 when SUPV=YES.

Neither register 1 nor register 6 is allowed.



Function Specifies the register in which the return code is to be placed. The return codes are:

X'0000'	The operation was successful.
X'0004'	The RCB was not associated with a virtual route, so no action was taken.
X'0006'	The RCB was not associated with this virtual route.
X'0008'	The virtual route is inoperative.

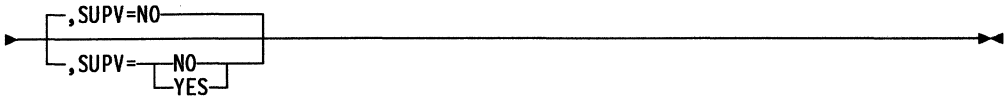
Format Register notation.

Default None.

Remarks The register you specify for RETCODE must not be the same as the register specified for RCB keyword.

When SUPV=YES, choose either register 5 or 7 for best results.

Do not specify register 6.



Function Specifies the level in which the issuer is running. SUPV=NO causes the macro to issue a supervisor request to level 4. SUP=YES causes the macro to link directly to the DETACHVR service routine.

Format YES or NO.

Default NO.

Remarks If you code SUPV=NO, the macro becomes nonreentrant.

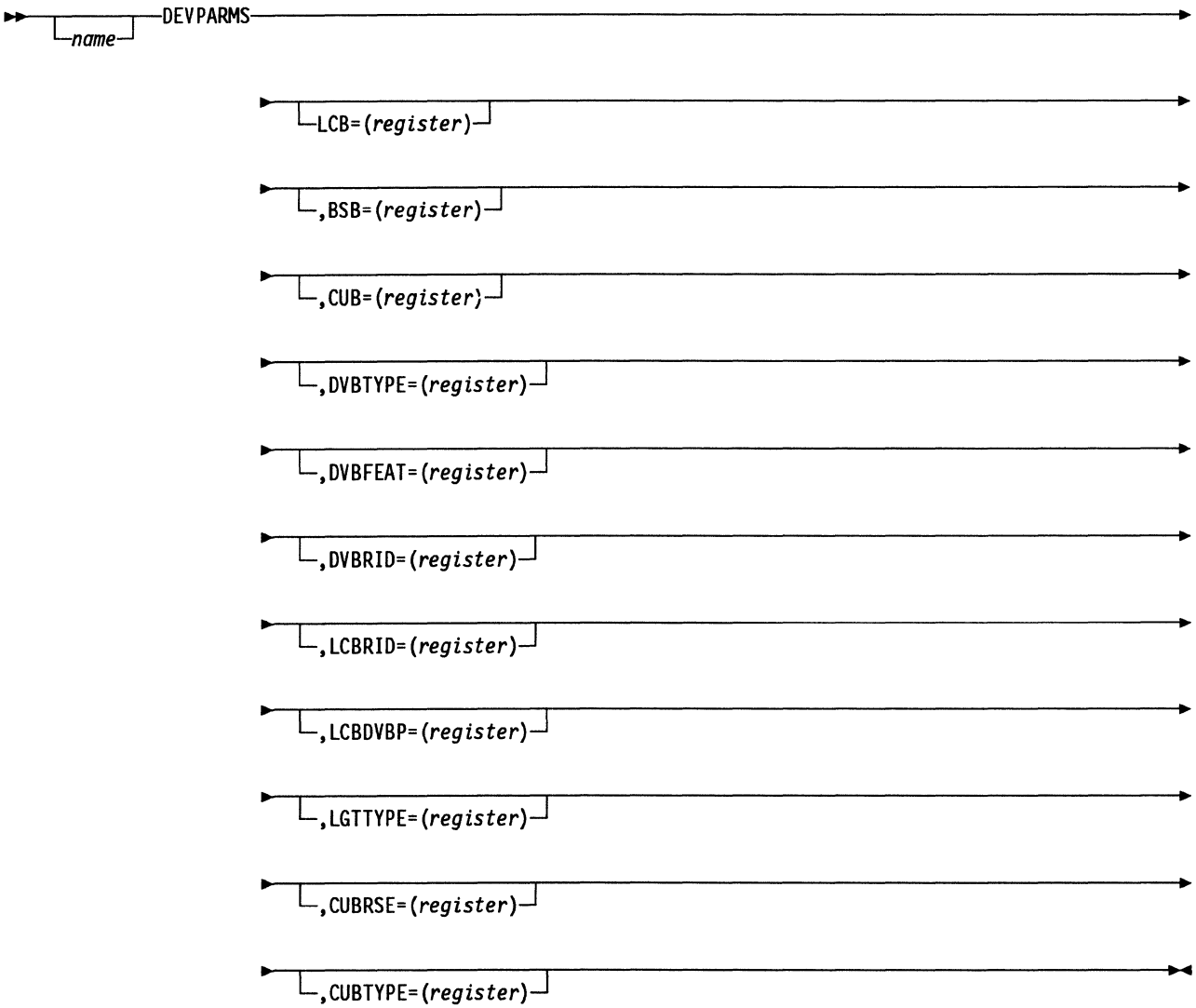
If SUPV=YES, register 6 must point to an available save area.

DEVPARMS—Provide Device Information Fields

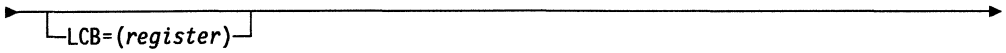
The DEVPARMS macro provides device information fields so that you are not dependent on NCP control block structure. The macro generates inline code only; there is no service routine.

Register 0 is not allowed for register parameters.

Syntax



Parameters

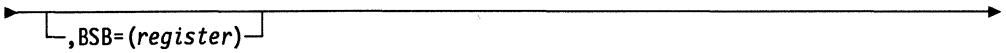


Function Specifies the register containing the address of the line control block (LCB) to be referred to.

Format Register notation.

Default None.

Remarks Specify one or more LCB, BSB, or CUB keywords. If you code BSB, use register 1, 2, 3, 4, 5, 6, or 7.

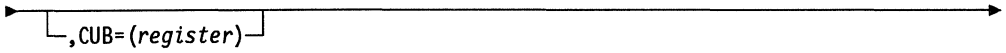


Function Specifies the register containing the address of the boundary session control block (BSB) to be referred to.

Format Register notation.

Default None.

Remarks Specify one or more LCB, BSB, or CUB keywords. If you code BSB, use register 1, 2, 3, 4, 5, 6, or 7.

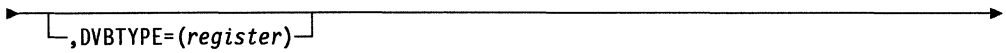


Function Specifies the register containing the address of the common physical unit block (CUB) to be referred to.

Format Register notation.

Default None.

Remarks Specify one or more LCB, BSB, and CUB keywords. If you code BSB, use register 1, 2, 3, 4, 5, 6, or 7.



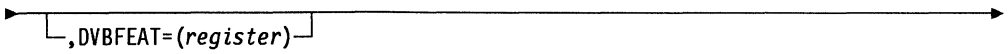
Function Specifies the odd-numbered register to be updated with the contents of the DVBTYPE field from the DVB pointed to by the LCB.

Format Register notation.

Default None.

Remarks The contents of the DVBTYPE field are returned in the low-order byte (byte 1) of the register. Use register 1, 3, 5, or 7.

This keyword is valid only if you code the LCB keyword.



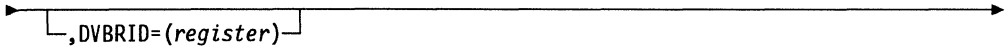
Function Specifies the register returned with the contents of the DVBFEAT1 and DVBFEAT2 fields from the DVB pointed to by the LCB.

Format Register notation.

Default None.

Remarks The contents of the DVBFEAT1 are returned in the high-order byte (byte 0) and the contents of DVBFEAT2 are returned in the low-order byte of the register. Use register 1, 2, 3, 4, 5, 6, or 7.

This keyword is valid only if you code the LCB keyword.

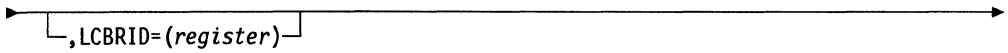


Function Specifies the register returned with the contents of the DVBRID field from the DVB pointed to by the LCB.

Format Register notation.

Default None.

Remarks This keyword is valid only if you code the LCB keyword. Use register 1, 2, 3, 4, 5, 6, or 7.

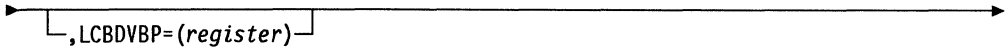


Function Specifies the register returned with the contents of the LCBRID field from the referred-to LCB.

Format Register notation.

Default None.

Remarks This keyword is valid only if you code the LCB keyword. Use register 1, 2, 3, 4, 5, 6, or 7.

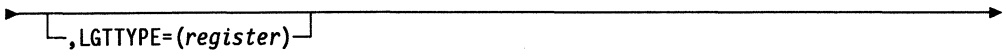


Function Specifies the register returned with the contents of the LCBDVBP field from the referred-to LCB.

Format Register notation.

Default None.

Remarks This keyword is valid only if you code the LCB keyword. Use register 1, 2, 3, 4, 5, 6, or 7.



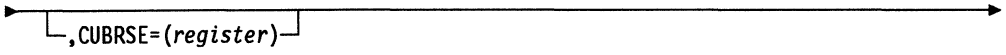
Function Specifies the odd-numbered register returned with the contents of the LGTTYPE field from the line group table (LGT). Use the LCB keyword to find the LGT.

Format Register notation.

Default None.

Remarks The contents of the LGTTYPE field are returned in the low-order byte (byte 1) of the register. Use register 1, 3, 5, or 7.

This keyword is valid only if you code the LCB keyword.

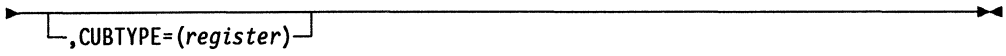


Function Specifies the register returned with the contents of the CUBRSE field from the referred-to CUB.

Format Register notation.

Default None.

Remarks This keyword is valid only if you code the CUB keyword. Use register 1, 2, 3, 4, 5, 6, or 7.



Function Specifies the odd-numbered register returned with the contents of the CUBTYPE field from the referred-to CUB.

Format Register notation.

Default None.

Remarks The contents of the CUBTYPE field are returned in the low-order byte (byte 1) of the register. Use register 1, 3, 5, or 7.

This keyword is valid only if you code the CUB keyword.

DOWHILE and DOUNTIL—Begin a Loop Program Structure

The DOWHILE and DOUNTIL macros provide the structure for repeated looping while or until specific conditions are met. They are used with the LEAVEDO and ENDO macros to form DO-loop program structures. The DOWHILE and DOUNTIL macros mark the beginning of the loop and the ENDO macro marks the end of the loop.

The DOWHILE and DOUNTIL macros specify the condition for executing the DO loop. For DOWHILE, the DO loop is executed as long as the condition is true. The condition is evaluated at the beginning of the loop, so if the condition is not true when the loop is entered for the first time, the loop is not executed. For DOUNTIL, the DO loop is executed until the condition is true. The condition is evaluated at the end of the loop, so even if the condition is true when the loop is entered for the first time, the loop is executed at least once.

The DOWHILE and DOUNTIL macros may form a multiple-step test through a logical connective keyword (with certain restrictions); however, DOWHILE and DOUNTIL macros cannot be mixed to build a test for a single DO loop. The last or only DOWHILE or DOUNTIL macro in a test designates the beginning of a DO loop function, and at least one is required for each DO loop. Loop processing is ended by branching to the next instruction following the ENDDO macro. The LEAVEDO macro can be used to end all loop processing without necessarily satisfying the specified test. The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro. DO loops can be nested, and the separate loops may be a mixture of DOWHILE and DOUNTIL types.

DOWHILE and DOUNTIL macros each have seven format types:

- The *no-operation* format includes instructions for the first DOWHILE or DOUNTIL test within the loop.
- The *test CL, ZL* format provides the same test as the corresponding IF macro format.
- The *branch-on-bit* format provides the same test as the corresponding IF macro format.
- The *test-under-mask* format provides the same test as the corresponding IF macro format.
- The *comparison* format provides the same test as the corresponding IF macro format.
- The *DO-X-by-Y* format decrements the variable *X* by *Y* (either a constant or the value in a register) each time the loop test is performed. The DO loop function is executed if the resulting value of *X* is greater than 0, and the loop is ended if *X* is 0 or negative. Note that the most efficient format is generated when *Y* is 1. This format can occur only once per DO loop, and it is restricted to the first DOWHILE or DOUNTIL in a test.
- The *branch-on-count* format decrements the contents of a register by 1 until the result becomes 0. This format has a limited branch range because of the limited range of the BCT instruction it uses. The test is assembled at the end of the DO loop (at ENDDO) for maximum cycle efficiency. This format cannot be connected logically with another DOWHILE or DOUNTIL in a test; it must be

the only DOWHILE or DOUNTIL in a DO loop. However, it may be nested within other DO loops and may have other DO loops nested within it.

Logical Connective Evaluation

There are 4 logical connectives (or logical operators): AND, OR, (AND), and (OR). The different connectives indicate how the Boolean expression is parsed. The Boolean expression is fully parenthesized before execution.

The following examples show how to place parentheses around the logical connectives (AND or OR) in multiple DOWHILE macros to perform tests containing two, three, or four parts in various logical groupings. You can code DOUNTIL macros the same way.

In these examples, *x*, *y*, and *z* represent either logical connective (AND or OR); this convention lets you match the connectives in the coding samples to the connectives in the tests they perform. A, B, C, and D represent individual tests, such as:

Coding	*Test*
`R4(7),ON`	Bit 7 of register 4 is on.
`R3(0),Z,0F`	Byte 0 of register 3 ANDed with mask `X'0F'` is zero.
`R5,LT,R7`	The value in register 5 is less than the value in register 7.

Note: Register 0 is not allowed for register parameters.

Code a two-part test as follows:

Code This...	**For This Test**
`DOWHILE A,`*x* `DOWHILE B`	A *x* B

Code three-part tests as follows:

Code This...	**For This Test**

`DOWHILE A,(`*x*`)`
`DOWHILE B,`*y*
`DOWHILE C`
	(A *x* B) *y* C

`DOWHILE A,`*x*
`DOWHILE B,(`*y*`)`
`DOWHILE C`
	A *x* (B *y* C)

`DOWHILE A,(`*x*`)`
`DOWHILE B,(`*y*`)`
`DOWHILE C`
	A *x* (B *y* C)

Code four-part tests as follows:

Code This...	For This Test
DOWHILE A,*x* DOWHILE B,*y* DOWHILE C,*z* DOWHILE D	((A *x* B) *y* C) *z* D
DOWHILE A,(*x*) DOWHILE B,*y* DOWHILE C,*z* DOWHILE D	((A *x* B) *y* C) *z* D
DOWHILE A,*x* DOWHILE B,(*y*) DOWHILE C,*z* DOWHILE D	(A *x* (B *y* C)) *z* D
DOWHILE A,*x* DOWHILE B,*y* DOWHILE C,(*z*) DOWHILE D	(A *x* B) *y* (C *z* D)
DOWHILE A,(*x*) DOWHILE B,(*y*) DOWHILE C,*z* DOWHILE D	(A *x* (B *y* C)) *z* D
DOWHILE A,(*x*) DOWHILE B,*y* DOWHILE C,(*z*) DOWHILE D	(A *x* B) *y* (C *z* D)
DOWHILE A,*x* DOWHILE B,(*y*) DOWHILE C,(*z*) DOWHILE D	A *x* (B *y* (C *z* D))
DOWHILE A,(*x*) DOWHILE B,(*y*) DOWHILE C,(*z*) DOWHILE D	A *x* (B *y* (C *z* D))
DOWHILE B,(*x*) DOWHILE C,*y* DOWHILE D,*z* DOWHILE A	A *z* ((B *x* C) *y* D) (Notice that you code the individual tests in a different order for this test.)

In the following example, the loop (the instructions between the DOWHILE macros and the ENDDO macro) is executed as long as:

- Register 2 is not equal to register 6 *and* register 4 is not equal to register 6
  or
- The Z condition latch is off.

```
LOOP1    DOWHILE R2,NE,R6,H,(AND)
         DOWHILE R4,NE,R6,H,OR
         DOWHILE ZL,OFF
           .
           .
           .
ENDLOOP1 ENDDO
```

In the following example, register 6 is decremented by 2 each time the loop is executed. The loop is executed until:

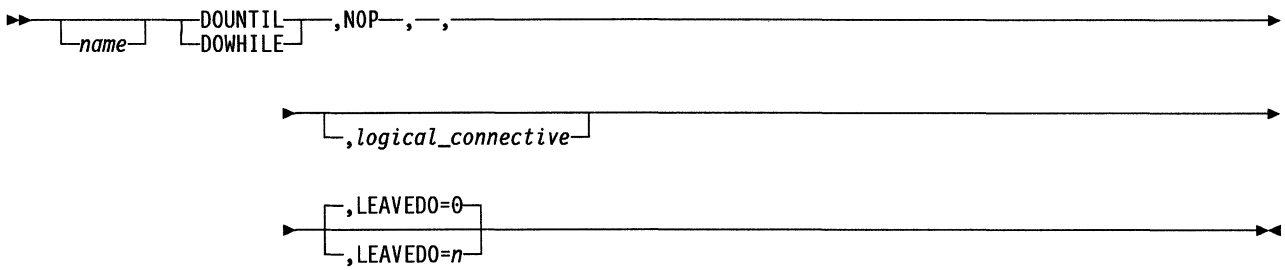
- The value of register 6 is zero
  or
- The 4 low-order bits of byte 0 of register 4 are zero *and* bit 7 of byte 0 of register 2 is on.

In addition, there are two LEAVEDO macros coded in the loop.

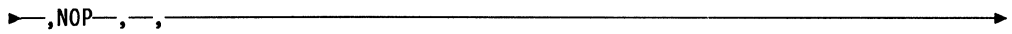
```
TWO      EQU 2
LOOP2    DOUNTIL (R6),BY,TWO,F,OR,LEAVEDO=2
         DOUNTIL R4(0),Z,0F,I,(AND)
         DOUNTIL R2(0,7),NZ,,H,
           .
           .
           .
ENDLOOP2 ENDDO
```

No-Operation Format

Syntax



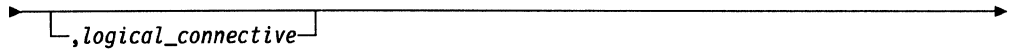
Parameters



Function Specifies the no-operation format.

Format NOP.

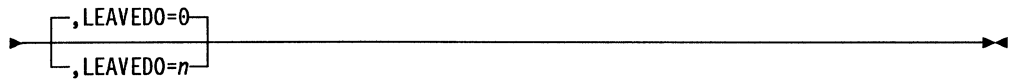
Default None.



Function Specifies the Boolean logic operation used to connect this DOWHILE or DOUNTIL macro with the immediately following DOWHILE or DOUNTIL macros. The function of this keyword corresponds to the AND logical connective for the IF comparison format. See the description under "Logical Connective Evaluation" on page 128.

Format AND, (AND).

Default None.



Function Declares the number of LEAVEDOs that are executed within this particular DO loop.

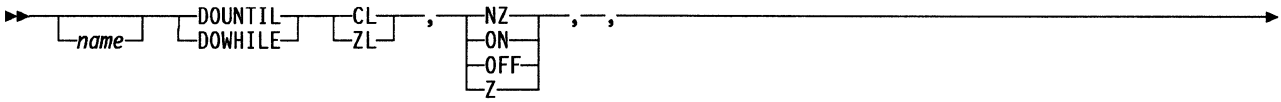
Format Label notation.

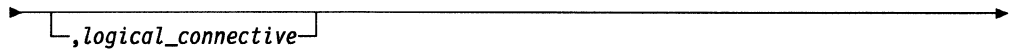
Default 0.

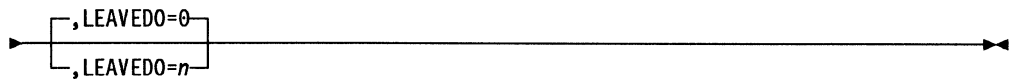
Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

Test CL, ZL Format

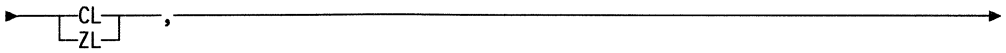
Syntax







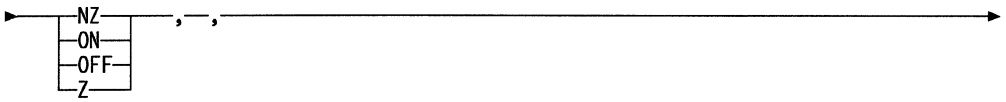
Parameters



Function Specifies that the C condition latch or the Z condition latch is to be tested.

Format CL or ZL.

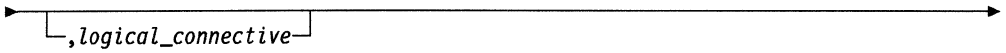
Default None.



Function Specifies the true condition (on, off, zero, or nonzero) for the specified latch.

Format ON, OFF, Z, or NZ.

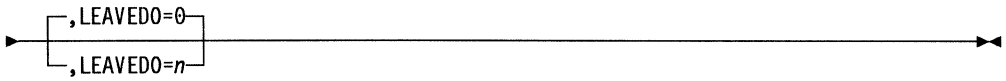
Default None.



Function Specifies the Boolean logic operation used to connect this DOWHILE or DOUNTIL macro with a following DOWHILE or DOUNTIL macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 128.

Format AND, OR, (AND), or (OR).

Default No logical connection is made between this DOWHILE or DOUNTIL macro and the following DOWHILE or DOUNTIL macro.



Function Declares the number of LEAVEDOs that are executed within this particular DO loop.

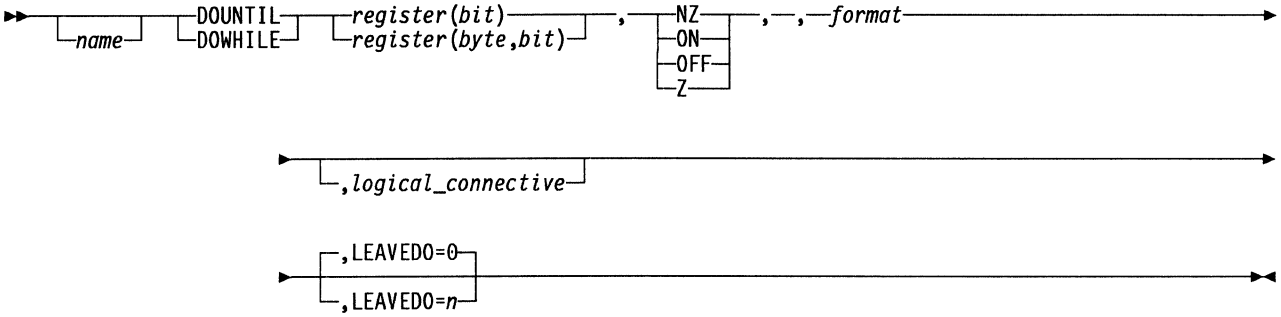
Format Label notation.

Default 0.

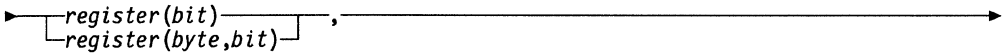
Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

Branch-on-Bit Format

Syntax



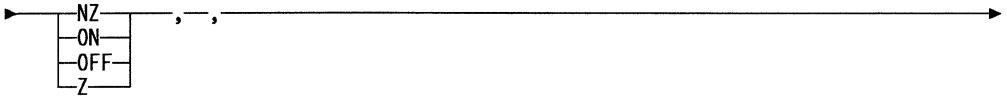
Parameters



Function Specifies the odd-numbered byte or halfword register containing the bit to be tested. If you specify a byte register, you can specify a bit from 0 to 7, and you must specify the format keyword as B. If you specify a halfword register, you can specify a bit from 0 to 15, and you must specify the format keyword as H.

Format Byte or halfword register notation.

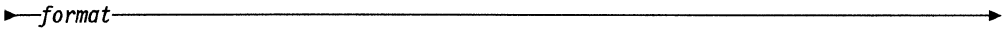
Default None.



Function Specifies the true condition of the specified bit: on, off, zero, or nonzero.

Format ON, OFF, Z, or NZ.

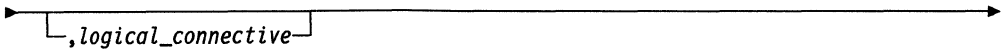
Default None.



Function Specifies whether the register is a byte (B) or halfword (H) register.

Format B or H.

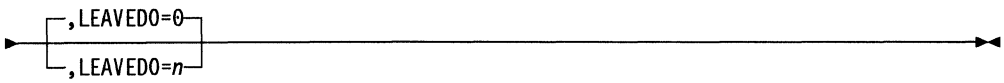
Default None.



Function Specifies the Boolean logic operation used to connect this DOWHILE or DOUNTIL macro with a following DOWHILE or DOUNTIL macro and the order in which the keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 128.

Format AND, OR, (AND), or (OR).

Default No logical connection is made between this DOWHILE or DOUNTIL macro and the following DOWHILE or DOUNTIL macro.



Function Declares the number of LEAVEDOs that are executed within this particular DO loop.

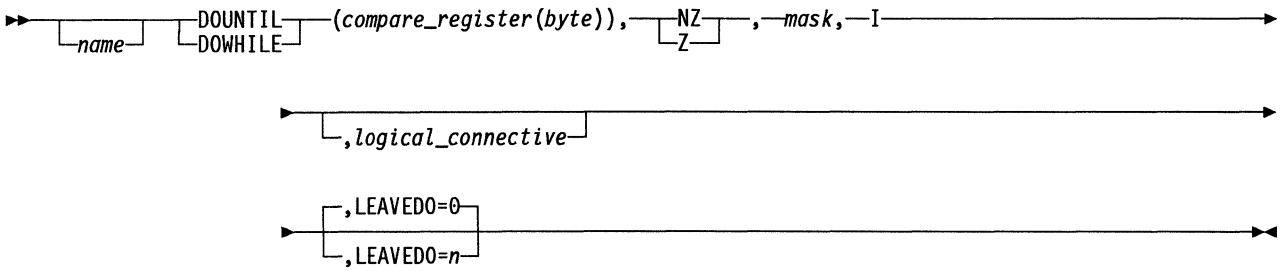
Format Label notation.

Default 0.

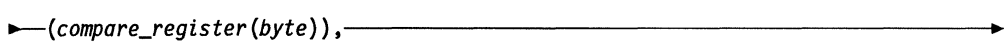
Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

Test-under-Mask Format

Syntax



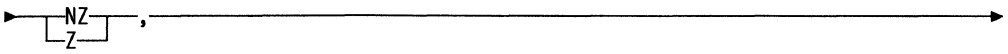
Parameters



Function Specifies the odd-numbered byte register, the contents of which are used in a logical AND comparison with the mask keyword.

Format Byte register notation.

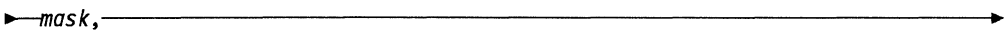
Default None.



Function Specifies the result of a logical AND comparison between the byte register contents and the mask keyword. Z is true if all the byte register bits corresponding to the mask bits are 0. NZ is true if one or more of the byte register bits corresponding to the mask bits are nonzero.

Format Z or NZ.

Default None.



Function Specifies an immediate byte value to be used in a logical AND comparison with the contents of the specified byte register.

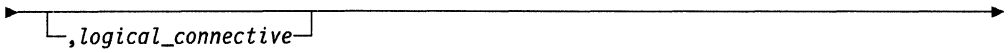
Format Label notation

Default None.

Function Specifies, in conjunction with Z or NZ, the test-under-mask format.

Format I.

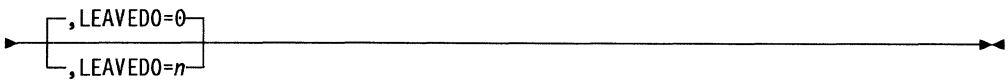
Default None.



Function Specifies the Boolean logic operation used to connect this DOWHILE or DOUNTIL macro with a following DOWHILE or DOUNTIL macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 128.

Format AND, OR, (AND), or (OR)

Default No logical connection is made between this DOWHILE or DOUNTIL macro and the following DOWHILE or DOUNTIL macro.



Function Declares the number of LEAVEDOs that are executed within this particular DO loop.

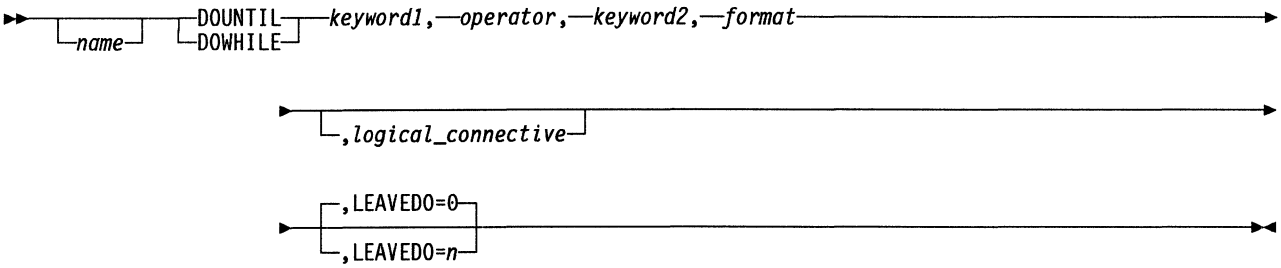
Format Label notation.

Default 0.

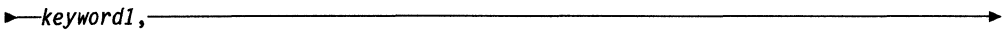
Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

Comparison Format

Syntax



Parameters



Function Specifies the first value for the comparison test.

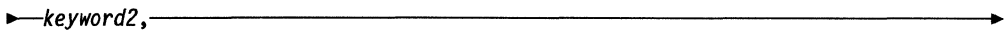
Format Byte, halfword, or fullword (20-bit) register notation.

Default None.

Function Specifies the type of comparison to be made between *keyword1* and *keyword2*: greater than (GT), less than (LT), equal to (EQ), not equal to (NE), not greater than (NG), not less than (NL), greater than or equal to (GE), or less than or equal to (LE).

Format GT, LT, EQ, NE, NG, NL, GE, or LE.

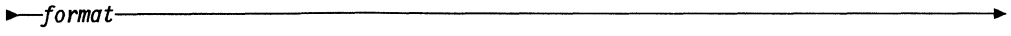
Default None.



Function Specifies the second value for the comparison test.

Format Byte, halfword, or fullword (20-bit) register notation, or if the format keyword is specified as I, label notation.

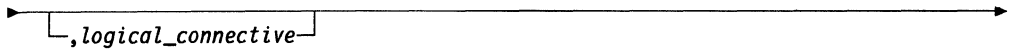
Default None.



Function Specifies whether *keyword1* and *keyword2* are byte registers (B), halfword registers (H), or fullword (20-bit) registers (F). Code I to specify that *keyword1* is a byte register and *keyword2* is a byte-immediate value.

Format B, H, F, or I.

Default None.



Function Specifies the Boolean logic operation used to connect this DOWHILE or DOUNTIL macro with a following DOWHILE or DOUNTIL macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 128.

Format AND, OR, (AND), or (OR)

Default No logical connection is made between this DOWHILE or DOUNTIL macro and the following DOWHILE or DOUNTIL macro.

Function Declares the number of LEAVEDOs that are executed within this particular DO loop.

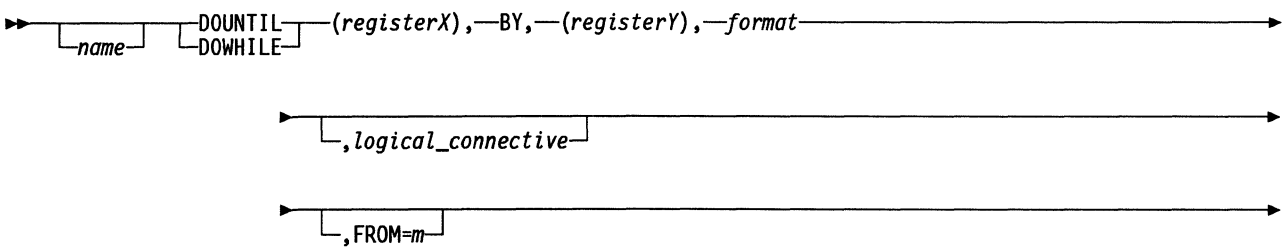
Format Label notation.

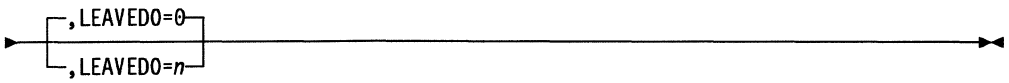
Default 0.

Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

DO-X-by-Y Format

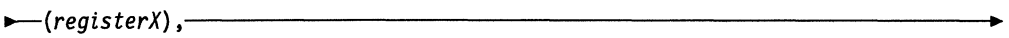
Syntax





Note: This format can occur only once per DO loop, and it is restricted to the first DOWHILE or DOUNTIL in a test.

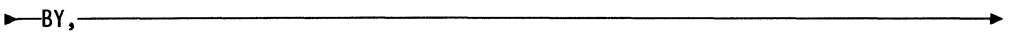
Parameters



Function Specifies the register containing the variable *X*. Each time the test is executed, the contents of this register are decremented by *Y*.

Format Register notation.

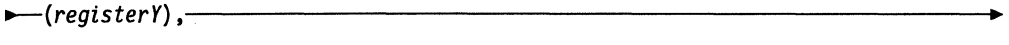
Default None.



Function Specifies the DO-X-by-Y format.

Format BY.

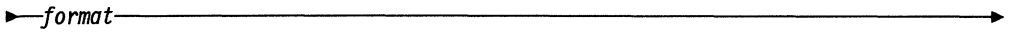
Default None.



Function Specifies *Y*, the value by which *registerX* is to be decremented each time the test is performed.

Format Register or label notation. In label notation, the range is from 0 to 255 for byte format and from 0 to 65 535 for halfword format.

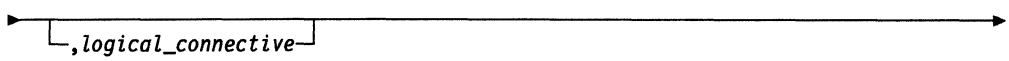
Default None.



Function Specifies whether *registerX* and *registerY* are in byte (B), halfword (H), or fullword (F) format. It may also specify *registerY* as a byte-immediate value (I).

Format B, H, F, or I.

Default None.



Function Specifies the Boolean logic operation used to connect this DOWHILE or DOUNTIL macro with a following DOWHILE or DOUNTIL macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with

parentheses and those without. See the description under "Logical Connective Evaluation" on page 128.

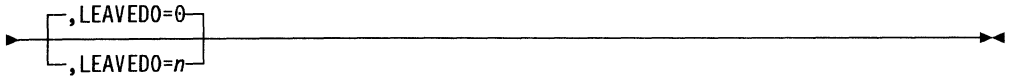
Format AND, OR, (AND), or (OR).

Default None.

Function Specifies the first value of *X*. This value is set into register 1 on entry to the DO loop. If this keyword is blank, register 1 is assumed to be preset to its first value and so is not initialized when the DO loop is entered.

Format Blank or label notation. The range is from 0 to 255 for byte format, and from 0 to 65 535 for halfword format.

Default None.



Function Declares the number of LEAVEDOs that are executed within this particular DO loop.

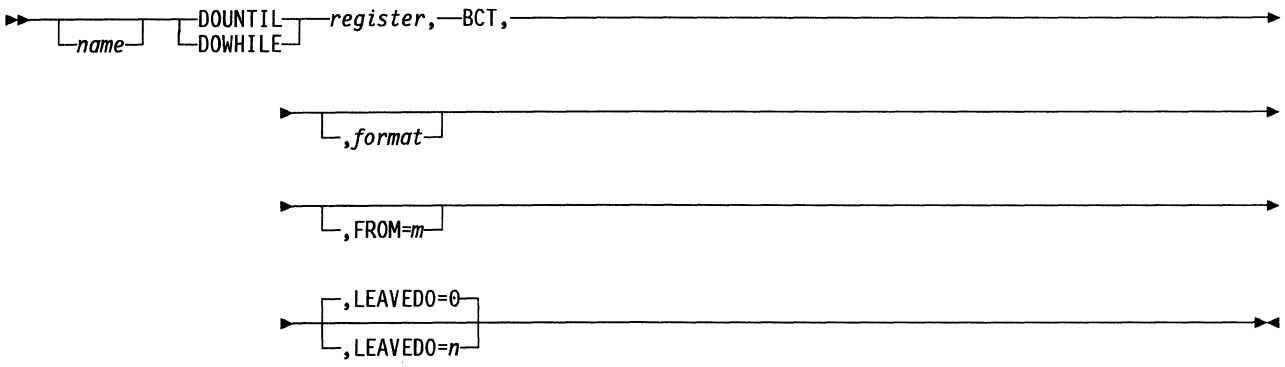
Format Label notation.

Default 0.

Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

Branch-on-Count Format

Syntax



Note: This format cannot be connected logically with another DOWHILE or DOUNTIL in a test; it must be the only DOWHILE or DOUNTIL in a DO loop.

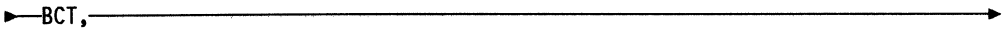
Parameters

Function Specifies the register to be decremented each time the test is performed.

Format Byte or halfword register notation.

Default None.

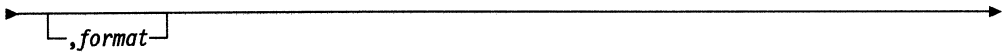
Remarks Specify an odd-numbered register only. When you specify the low-order byte of a register using byte notation, the complete halfword register is decremented.



Function Specifies the branch-on-count format.

Format BCT.

Default None.

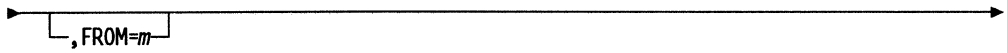


Function Specifies whether the register is in byte (B) or halfword (H) register notation.

Format B or H.

Default None.

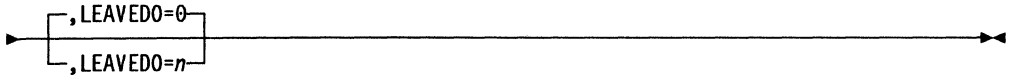
Remarks This keyword in this particular format must be used in the third, rather than the fourth keyword position; the latter is this keyword's position in other formats.



Function Specifies the first count. This value is set into the register on entry to the DO loop. If this keyword is blank, the register is assumed to be preset to its first value and is not initialized when the DO loop is entered.

Format Blank or label notation. The range is from 0 to 255 for byte format, and from 0 to 65 535 for halfword format.

Default None.



Function Declares the number of LEAVEDOs executed within this DO loop.

Format Label notation.

Default 0.

Remarks The use of LEAVEDO within a loop must be declared on the first DOWHILE or DOUNTIL macro.

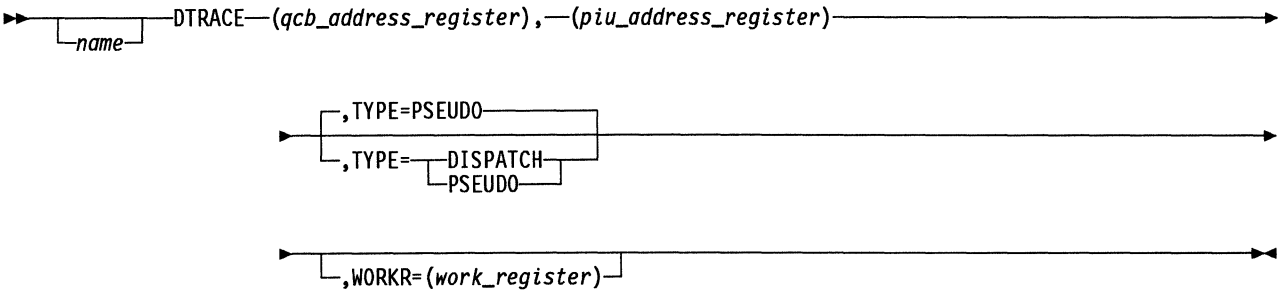
DTRACE—Add an Entry to the Dispatcher Trace Table

The DTRACE macro produces linkage code to branch to the dispatcher trace routine and produces an entry in the dispatcher trace table containing the queue control block (QCB), the path information unit (PIU), and task entry point addresses. A time stamp and indicator (D for normal dispatch and B for pseudodispatch) are also put in the trace entry.

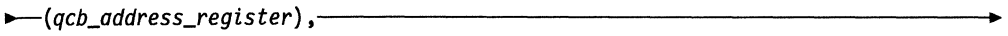
This macro can be used only in level 5 because it does not disable the lower levels. TYPE=DISPATCH is not recommended except for the task dispatcher.

Register 0 is not allowed for register parameters.

Syntax



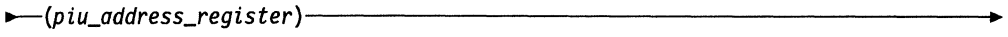
Parameters



Function Specifies the register containing the pointer to the QCB to be traced.

Format Register notation.

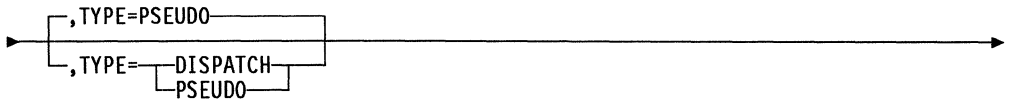
Default None.



Function Specifies the register containing the pointer to the PIU to be traced.

Format Register notation.

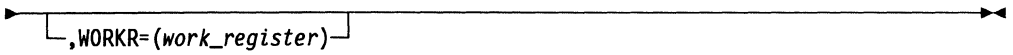
Default None.



Function Specifies the indicator byte of the trace entry. The indicator byte is D when TYPE=DISPATCH (normal dispatch) and B when TYPE=PSEUDO (pseudodispatch).

Format DISPATCH or PSEUDO.

Default PSEUDO.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

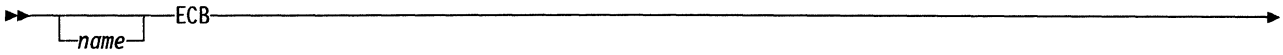
Default No work registers.

ECB—Build an Event Control Block

The ECB macro builds an event control block.

Register 0 is not allowed for register parameters.

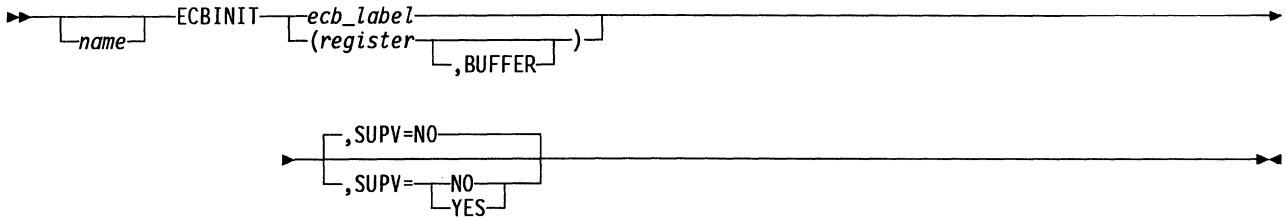
Syntax



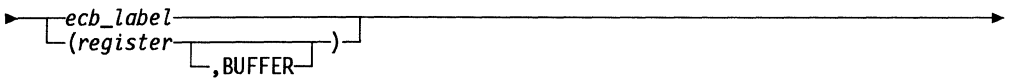
ECBINIT—Initialize an Event Control Block

The ECBINIT macro initializes the status bytes of a specified event control block (ECB).

Syntax



Parameters



Function Specifies the address of the ECB to be initialized.

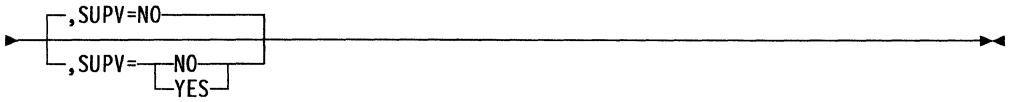
Format Register or label notation.

Remarks If the ECB is not defined in storage but is part of a block control unit (BCU) or a path information unit (PIU), you must use register notation and specify BUFFER. The register must point to the beginning of the first buffer of the BCU or PIU.

If you do not code BUFFER, ECBINIT sets 2 bytes to 0 beginning at offset 0 and 2 bytes to 0 beginning at offset 4 into the ECB specified by the ECB address.

If you code BUFFER, ECBINIT sets 4 bytes to 0 beginning at offset 8 and 2 bytes to 0 beginning at offset 12 into the BCU or PIU specified by the register. The ECB is located beginning at offset 8 into the BCU or PIU.

Note: If you use label notation, register 6 must point to a save area that can be overwritten.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

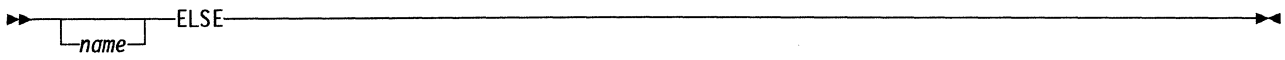
Format YES or NO.

Default NO.

ELSE—Begin an Else Condition Program Structure

The ELSE macro is used with the IF, THEN, and ENDIF macros to form an IF-THEN-ELSE program structure. It precedes the instructions that are executed if the results of tests made by the IF macro are false. You do not need to code an ELSE macro following an IF if the IF macro will be immediately followed by an ENDIF macro.

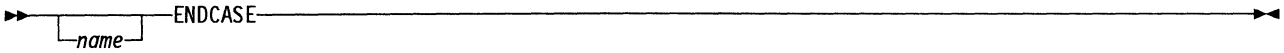
Syntax



ENDCASE—End a Case Program Structure

The ENDCASE macro is used with the CASEIF, CASENTRY, CASEXIT, and CASE macros to form a case program structure. The ENDCASE macro establishes the end of the case routine started by a CASE macro. The return function specified in the CASENTRY macro (for example, a branch back to the corresponding CASEXIT) is carried out at this point.

Syntax



ENDDO—End a DOWHILE or DOUNTIL Program Structure

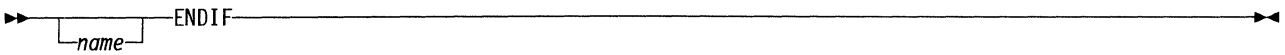
The ENDDO macro is used with the DOWHILE, DOUNTIL, and LEAVEDO macros to form DO loop program structures. The ENDDO macro designates the end of a DO loop function, and one is required for each DO loop.

Syntax

ENDIF—End an IF-THEN-ELSE Program Structure

The ENDIF macro is used with the IF, THEN, and ELSE macros to form an IF-THEN-ELSE program structure. The ENDIF macro follows the instruction sequence that is associated with the ELSE macro; it is always required to end the IF-THEN-ELSE program structure.

Syntax



ENQUE—Attach an Element to a Queue

The ENQUE macro attaches an element, which can be a block control unit (BCU), a path information unit (PIU), a queue control block (QCB), or an event control block (ECB), to the end of a specified system queue and, optionally, activates the queue so that the associated task is scheduled for execution.

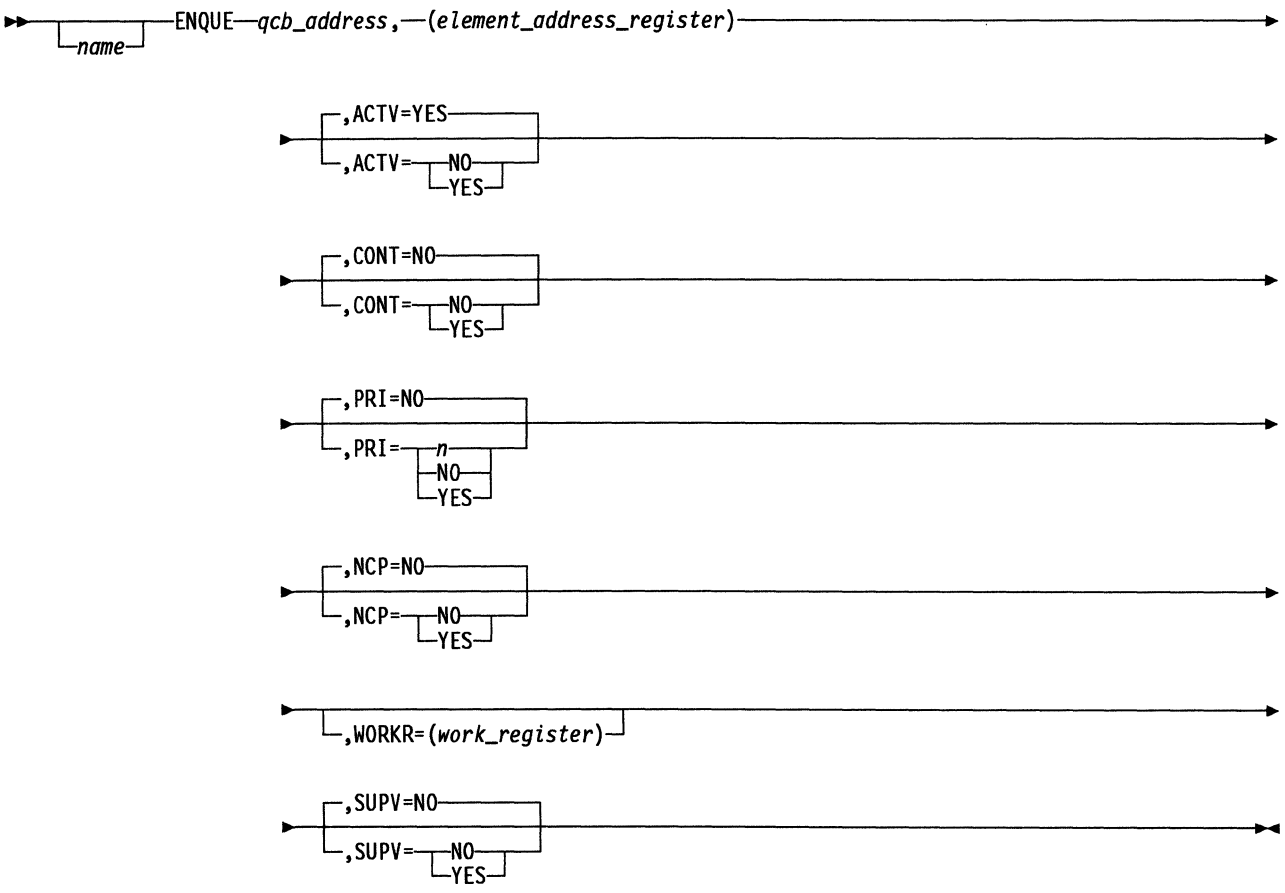
For additional information on how the ENQUE macro affects task states, see "Task Management" in *NCP and EP Reference.*

Use the INSERT macro to add an element anywhere in a queue except the last position.

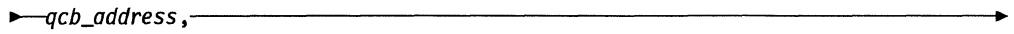
Register 0 is not allowed for register parameters.

Note: This macro writes to the current save area and assumes that this save area can be overwritten.

Syntax



Parameters



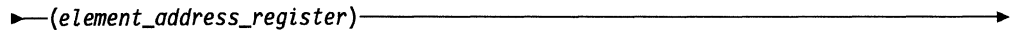
Function Specifies the address of the QCB governing the queue to which the element is to be enqueued.

Format Register or label notation.

Default If SUPV=YES, there is no default. If SUPV=NO, the address of the QCB that activated the task is the default.

Remarks Register 1 is not allowed.

If SUPV=YES, register 2 is standard and register 6 is not allowed.



Function Specifies the register containing the address of the element to be enqueued.

Format Register notation.

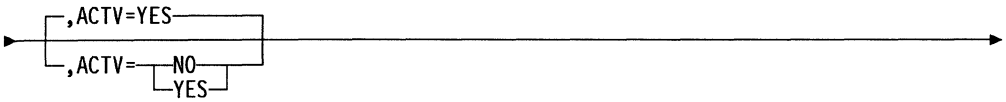
Default None.

Remarks Register 1 is not allowed.

If SUPV=YES, register 3 is standard and register 6 is not allowed.

If SUPV=YES, the element can be either a QCB or an ECB and does not have to be in a buffer.

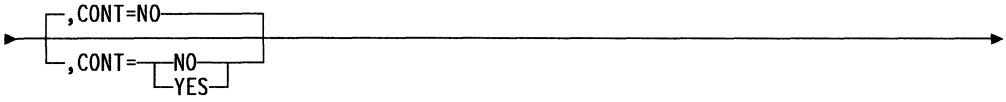
If SUPV=NO and the element is not a QCB, the element must be in a buffer; otherwise, NCP will abend.



Function Specifies whether a task is to be scheduled for execution. ACTV=YES specifies that the task is to be scheduled for execution, that is, put in the pending state if another task is already active or in the active state if another task is not active. ACTV=NO specifies that the task is not to be activated as a result of enqueuing the element, or that the element is to be enqueued to a system work queue.

Format YES or NO.

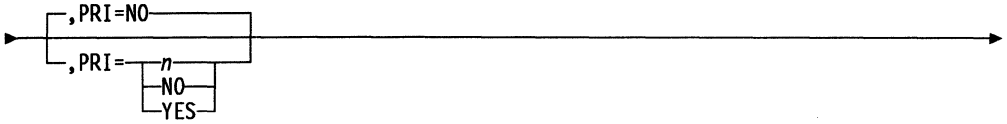
Default YES.



Function Specifies whether the queue is a contention queue (one that can be simultaneously manipulated by two or more interrupt levels).

Format YES or NO.

Default NO.



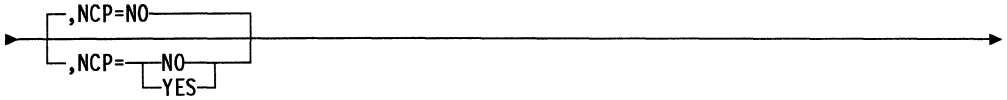
Function Specifies the priority of the element to be enqueued. PRI=NO specifies that the element is to be enqueued at the end of the system queue with priority 0. PRI=*n* may be any value from 0-15 and specifies a priority value. This means that the element is enqueued ahead of all elements on the queue with lower priority, but behind those elements with equal or higher priority. PRI=YES specifies that the enqueuing position is determined as in the *n* option, but the priority value is found in the element.

Format NO, or *n*, or YES.

Default NO.

Remarks The same function is carried out for 0 and NO.

A QCB is always enqueued at the end of the queue (effective priority 0). Do not use priority enqueuing with a queue on which QCBs are enqueued.



Function Specifies whether the routine is to be generated inline.

Format YES or NO.

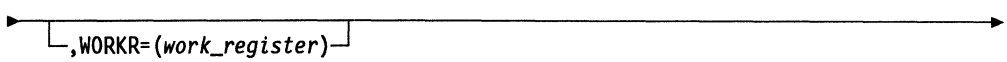
Default NO.

Remarks If NCP=YES, you must specify WORKR.

If NCP=YES, code CONT=NO or allow it to default to NO.

If NCP=YES, code PRI=NO or allow it to default to NO.

If NCP=YES, the module must copy the DSECTs for QCB, ECB, HWE, and XDA.

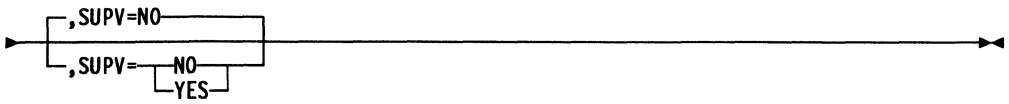


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks You must specify an odd-numbered register.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

EXCR—Build an Exception Response PIU

The EXCR macro builds an exception (negative) response path information unit (PIU) from a given request PIU specified in register 3. Four bytes of sense data, supplied as parameters, are inserted into the request/response unit (RU) followed by the RU prefix, if present. The remainder of the PIU is discarded. If the PIU is a FID0 or FID1, and XPORT=YES is specified, the PIU is sent to the originator of the request, and the content of register 3 is set to 0. If the PIU is a FID2 or if XPORT=YES is not specified, the exception response is returned to the EXCR issuer to be sent to the originator of the request.

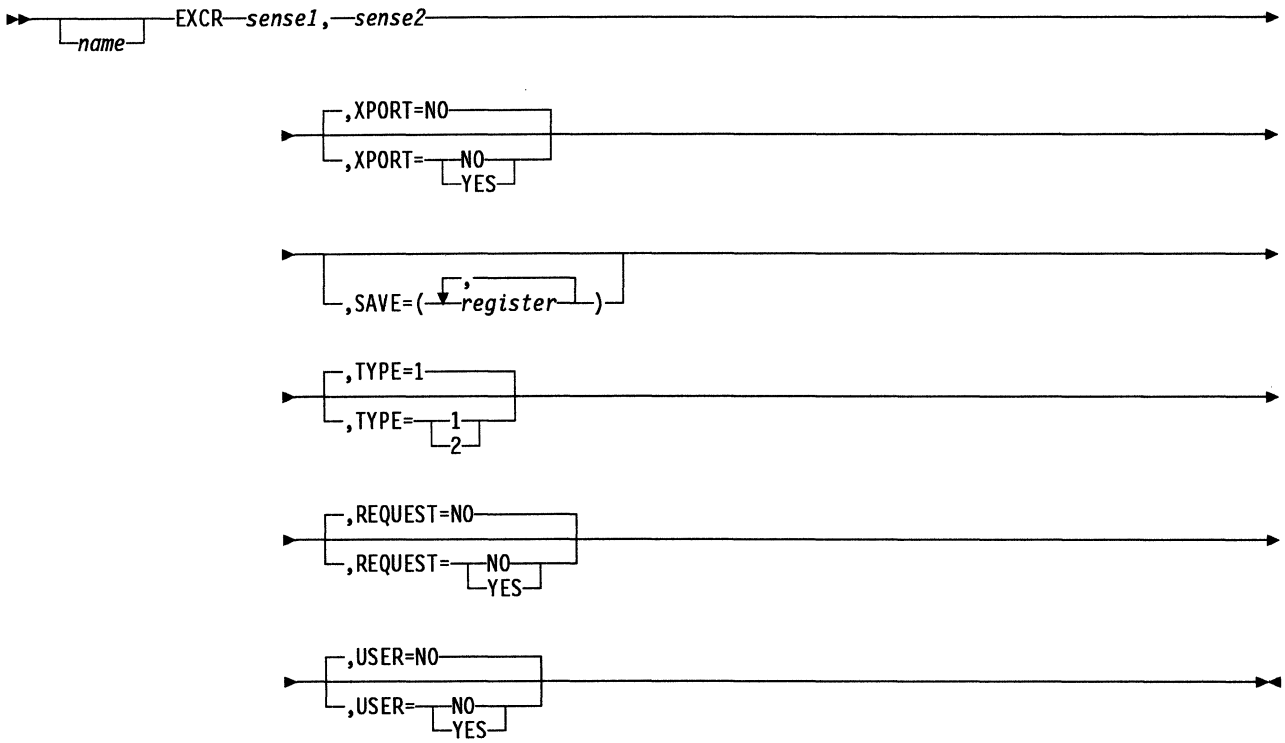
Register 0 is not allowed for register parameters.

The name of the failing module and the displacement of the EXCR macro are placed in the NCP buffer and are included in trace records.

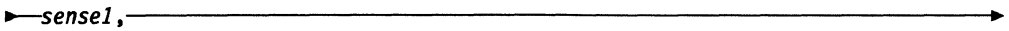
The EXCR macro applies to an unsegmented message or to only the first segment of a segmented message. Succeeding or last message segments are discarded when errors occur. The EXCR macro can be used in program level 5 only.

If the request/response header (RH) of the request PIU indicates "no response required," the PIU will be discarded.

Syntax



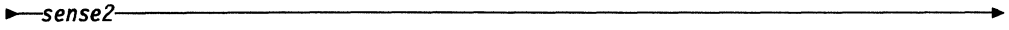
Parameters



Function Specifies 2 bytes of sense data to be included in the exception response.

Format Symbolic representation of the 2 sense bytes as they appear in the XXCXTSNS DSECT, or halfword register notation.

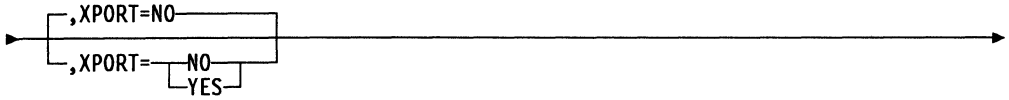
Default None.



Function Specifies 2 bytes of user sense data to be included in the exception response.

Format Symbolic representation of the 2 sense bytes as they appear in the XXCXTSNS DSECT, or halfword register notation.

Default None.



Function Specifies whether the FID0 or FID1 PIU is to be sent to the requestor. XPORT=YES specifies that the FID0 or FID1 PIU is to be sent to the requestor.

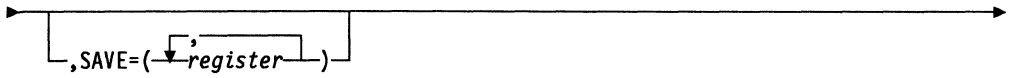
Format YES or NO.

Default NO.

Remarks XPORT=YES is not allowed if TYPE=2.

The PIU must contain a valid virtual route vector table index (VVTI) if XPORT=YES.

Code XPORT=YES only for functions in the peripheral node.



Function Specifies the registers that are to be saved before invoking the subroutine.

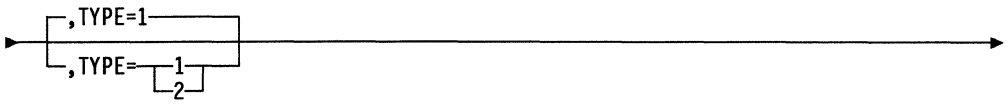
Format Register.

Default No registers are saved.

Remarks If TYPE=1, only registers 4, 5, and 7 may be saved.

If TYPE=2, only registers 5 and 7 may be saved.

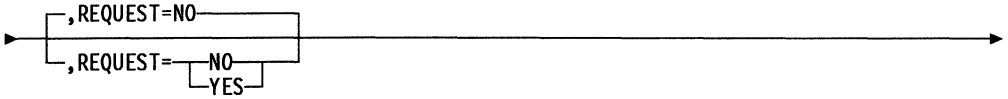
If the registers used above are not saved, their contents will be lost.



Function Identifies the PIU type.

Format 1 for FID0 or FID1 PIU; 2 for FID2 PIU.

Default 1.

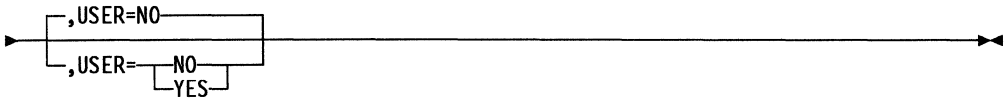


Function Specifies whether an exception request is to be built.

Format YES or NO.

Default NO.

Remarks REQUEST=YES is not allowed if TYPE=2.



Function Specifies the storage in which the macro is being used. USER=NO specifies that the macro is being used in NCP's storage (protection key 0). USER=YES specifies the user's storage (protection key 1).

Format YES or NO.

Default NO.

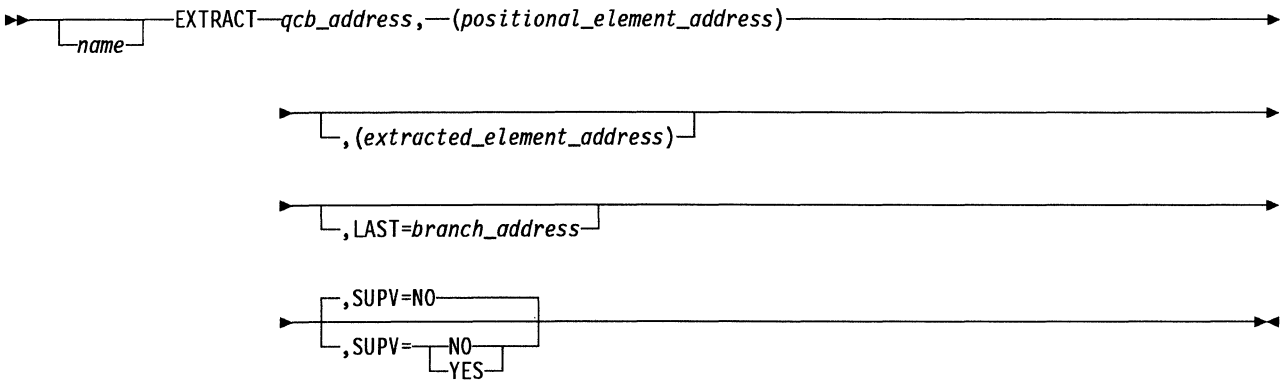
EXTRACT—Detach an Element from a Queue

The EXTRACT macro detaches an element, which can be a block control unit (BCU), a path information unit (PIU), or a queue control block (QCB), from any specified position (except the first) in a given system queue.

Use the DEQUE macro to detach the first element in a system queue.

Register 0 is not allowed for register parameters.

Syntax



Parameters

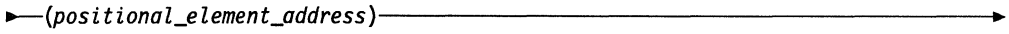
Function Specifies the address of the QCB governing the queue from which the element is to be extracted.

Format Register or label notation.

Default If SUPV=YES, there is no default. If SUPV=NO, the QCB that activated the issuing task is used.

Remarks Register 1 is not allowed.

If SUPV=YES, register 2 is standard and register 6 is not allowed.



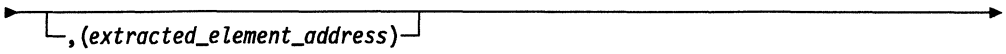
Function Specifies the register containing the address of the element that is immediately before the element to be extracted.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

If SUPV=YES, register 3 is standard and register 6 is not allowed.



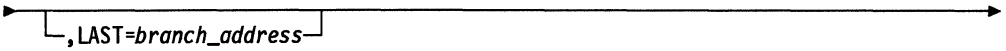
Function Specifies a register to receive the address of the extracted element.

Format Register notation.

Default The address is returned in the register containing the positional element address.

Remarks Register 1 is not allowed. Register 4 is standard.

If SUPV=YES, register 6 is not allowed.



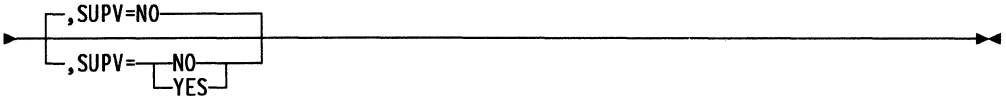
Function Specifies the address to be given control if the positional element is the last element on the queue.

Format Register or label notation.

Default The instruction following the macro gets control.

Remarks Register 1 is not allowed.

If SUPV=YES, register 6 is not allowed.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

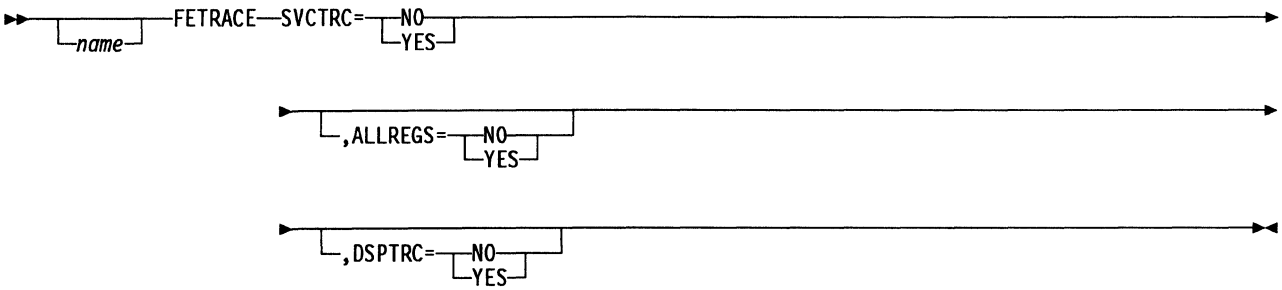
Format YES or NO.

Default NO.

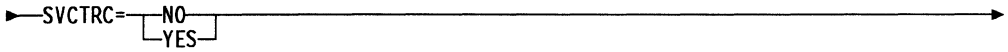
FETRACE—Activate the Supervisor, Register, and Dispatcher Traces

The FETRACE macro activates the supervisor, register, and dispatcher traces.

Syntax



Parameters

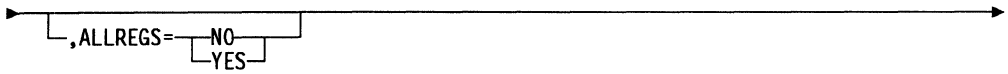


Function Specifies whether the supervisor trace is to be turned on.

Format YES or NO.

Default None.

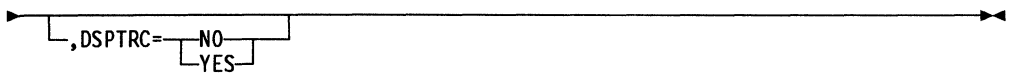
Remarks If you code SVCTRC=YES, you must code DSPTRC=YES.



Function Specifies whether all level 5 registers are to be traced in the supervisor trace entries.

Format YES or NO.

Default None.



Function Specifies whether the dispatcher trace is to be turned on.

Format YES or NO.

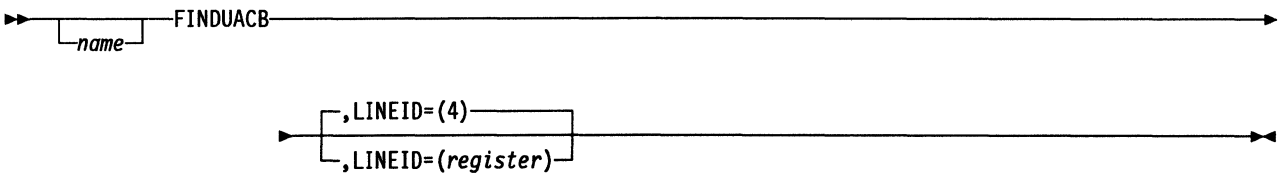
Default None.

Remarks If you code SVCTRC=YES, you must code DSPTRC=YES. If you code DSPTRC=NO, you must code SVCTRC=NO.

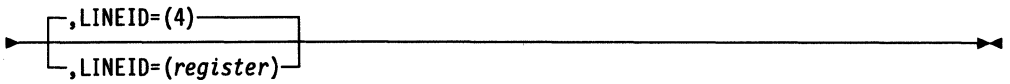
FINDUACB—Find a User Adapter Control Block (UACB)

The FINDUACB macro in your level 2 interrupt handler implements user line control. You can also use it in level 5. It returns a pointer to the user adapter control block (UACB) associated with the interrupting line or the pointer to any other specified UACB. It uses register 1 as a work register.

Syntax



Parameters



Function Specifies the register containing the address of the line vector table (LNVT) entry for the line (line ID).

Format Register notation.

Default If you omit this keyword, the macro assumes that register 4 contains the pointer to the LNVT entry for the line ID, and the macro will return the UACB pointer in register 4.

Remarks Register 1(0) is standard.

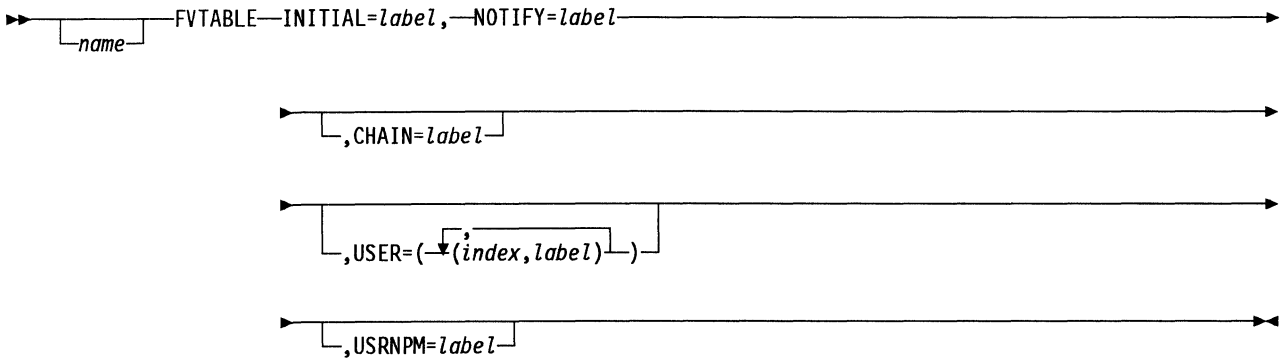
Register 1 is modified by this macro.

Register 0 is not allowed.

FVTABLE—Build a Function Vector Table

The FVTABLE macro builds a list of tasks that can be associated with one or more programmed resources. This list is called a function vector table (FVT). Its name is specified with the resource at program generation time.

Syntax

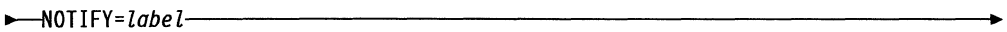


Parameters

Function Specifies the name of the task that will be inserted into the queue control block (QCB) to initialize the resource at NCP initialization. This task will always have an index value of 1 in the FVT.

Format Label notation.

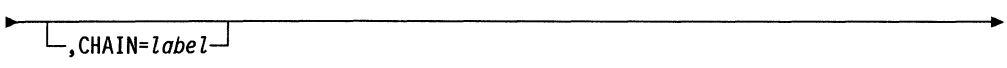
Default None.



Function Specifies the name of the task that will be inserted into the QCB by NCP whenever a condition exists that directly affects the resource. This task will always have an index value of 2 in the FVT. The task should take whatever action is required by SNA and any other action you desire within the constraints of SNA.

Format Label notation.

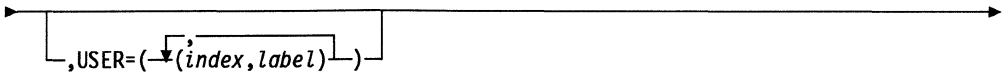
Default None.



Function Chains the FVT to another FVT. *Label* is the name of the next FVT in the chain.

Format Label notation.

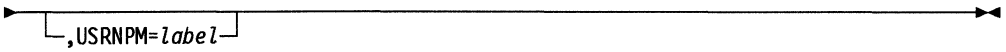
Default None.



Function Specifies user tasks. Each parameter value pair specifies an index and a user task label. Indexes must be in ascending, but not necessarily consecutive, order from 10 to 255. All tasks to be associated with a resource QCB must be included in the FVT specified with the resource or in a chained FVT.

Format Absolute notation and label notation.

Default None.



Function Specifies the user's NetView* Performance Monitor (NPM) task. This task will always have an index value of 4 in the FVT.

Format Label notation.

Default None.

GALERT—Build a Generic Alert NMVT PIU

The GALERT macro is used by IBM special products or user-written code to build the common fields of a generic alert network management vector transport (NMVT) path information unit (PIU). It allows IBM special products or user-written code to interface with NCP code to build the common generic alert NMVT fields without violating storage protection keys. The information that GALERT builds and stores in buffers provided by IBM special products or user-written code includes the following:

- Transmission header (TH)
- Response/request header (RH)
- NMVT request/response unit (RU) header
- Generic alert length bytes and major vector code
- SNA address list subvector X'04'
- Product set ID subvector X'10'
- Product set ID subvector X'11', which is defined by user-written code
- Relative time subvector X'42'.

Note: The product set ID subvector X'11' that has been defined by user-written code will appear in the common generic alert only if the PRODID keyword is coded on the GROUP definition statement. The length of the IBM special products or user-written code subvector X'11' should not exceed 80 bytes.

You can use register 7 as an input register, but the contents may not be preserved. Register 6 must point to a valid save area. Register 0 is not allowed.

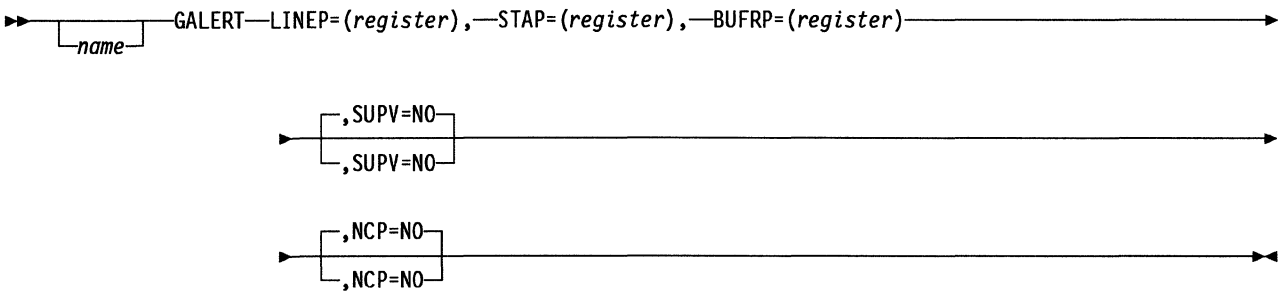
On exit, register 1 contains one of the following return codes:

X'0000' Macro executed successfully, and a common generic alert NMVT is built.

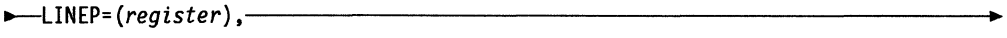
X'0001' No SSCP owners found, and a common generic alert NMVT is not built.

X'0002' Insufficient buffers, and a common generic alert NMVT is truncated.

Syntax



Parameters



Function Specifies a register that points to the line control block (LKB).

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword. Neither register 1 nor register 6 is allowed.

Function Specifies the register that points to the station control block (SCB).

Format Register notation.

Default None.

Remarks The register you specify must not be the same as the register specified for any other keyword. Neither register 1 nor register 6 is allowed.

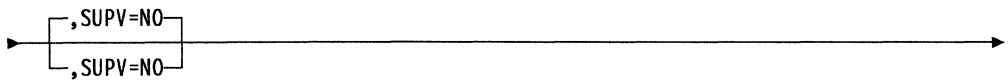
Function Specifies the register that points to a chain of buffers that will be the target for the generic alert items.

Format Register notation.

Default None.

Remarks You must pass a sufficient number of buffers, based on NCP buffer size. The common generic alert fields total a length of 150 bytes excluding the length of the IBM special products or user-written code product identification subvector. When the product identification subvector is included in the generic alert, the total length of the common generic alert fields is 150 bytes plus the length of the IBM special products or user-written code subvector X'11'.

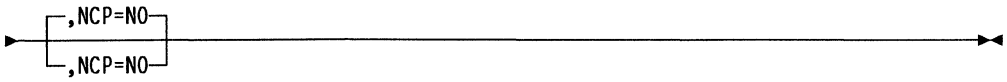
The register you specify must not be the same as the register specified for any other keyword. Neither register 1 nor register 6 is allowed.



Function Specifies that the issuer is running in level 5.

Format NO.

Default NO.



Function Specifies the type of code generated based on the issuer's storage protection key. NCP=NO specifies that the issuer is running in NCP storage protection key 1. SVC code is generated.

Format NO.

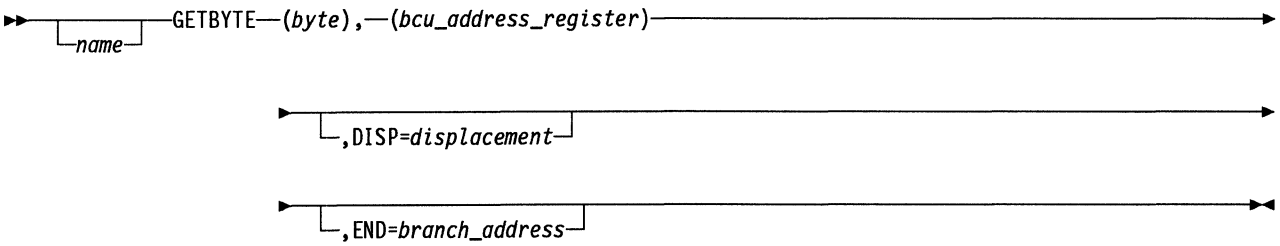
Default NO.

GETBYTE—Load a Register Byte from a Block Control Unit

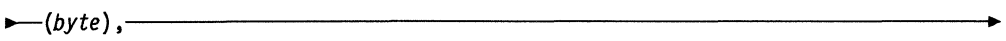
The GETBYTE macro loads a register byte with a data byte from a specified position in a block control unit (BCU). The position from which the byte is fetched is specified in one of two ways. The first way is to specify a displacement value in the DISP keyword. Using this displacement value, the supervisor calculates the appropriate byte in the appropriate buffer of the BCU and fetches the byte. The supervisor then increments the displacement value by 1 and saves it. The second way is to omit the DISP keyword. The supervisor uses the displacement saved from the previous GETBYTE macro to calculate the position of the data byte. The supervisor then increments the displacement value by 1 and saves it.

Register 0 is not allowed for register parameters.

Syntax



Parameters

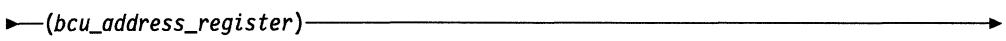


Function Specifies a register byte into which the data byte is to be loaded.

Format Byte register notation.

Default None.

Remarks Register 1(1) is not allowed. Register 1(0) is standard.

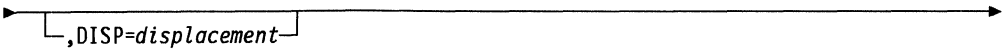


Function Specifies the register containing the address of the first byte of the first buffer of the BCU that contains the data byte to be fetched.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

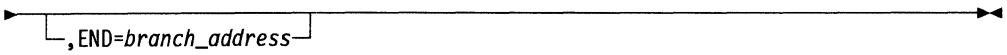


Function Specifies the displacement from the first data byte of the BCU to the position from which a data byte is to be fetched.

Format Absolute or register notation.

Default Computed displacement is used.

Remarks Register 1 is not allowed.



Function Specifies the address to be given control if either the specified or the computed displacement exceeds the total size of the BCU.

Format Register or label notation.

Default The issuing program tests register 1(1), bit 0. If the tested bit is 0, the byte was not fetched successfully.

Remarks Register 1 is not allowed.

GETCB—Get Associated Control Block

For NCP V7R1 or V7R1F and later versions, the GETCB macro returns a pointer to the requested control block when you provide a pointer to the associated control block.

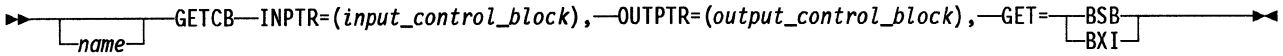
Do not specify the same register for more than one keyword.

Register 0 is not allowed for register parameters.

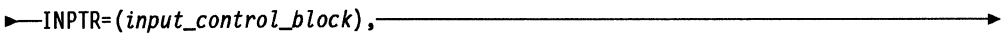
Note: You must invoke the following DSECT macros in the routine that issues the GETCB macro.

When:	You must invoke:
GET=BSB	XCXTBSB and XCXTBXI
GET=BXI	XCXTBSB

Syntax



Parameters



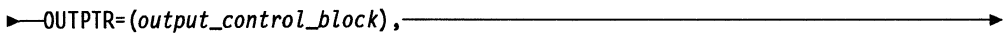
Function Specifies the register containing the address of the input control block.

Format Register notation.

Default None.

Remarks Register 6 is not allowed.

If GET=BSB is specified, INPTR must point to a boundary session block extension (BXI). If GET=BXI is specified, INPTR must point to an LU-LU boundary session block (BSB).



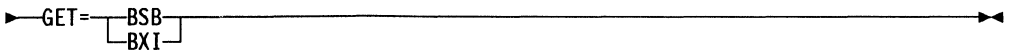
Function Specifies the register in which the address of the requested control block will be returned.

Format Register notation.

Default None.

Remarks OUTPTR must be an odd-numbered register.

If GET=BSB is specified, the address of the requested LU-LU BSB will be returned in OUTPTR. If GET=BXI is specified, the address of the requested BXI will be returned in OUTPTR.



Function Specifies the type of control block that is to be returned. GET=BSB specifies that the input BXI pointer will be used to get the pointer to the associated LU-LU BSB. GET=BXI specifies that the input LU-LU BSB will be used to get the pointer to the associated BXI.

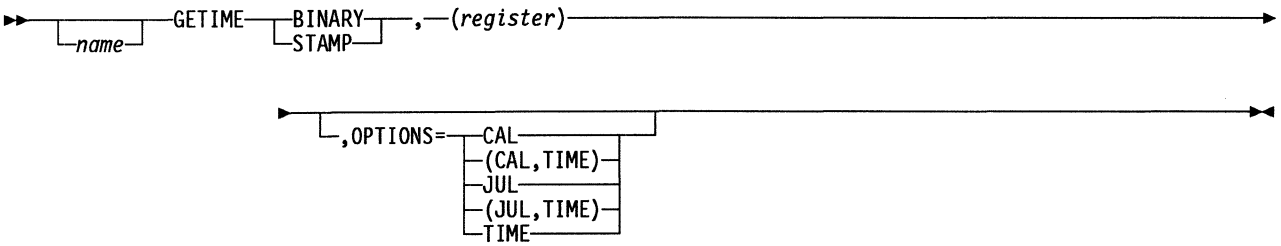
Format BSB or BXI.

Default None.

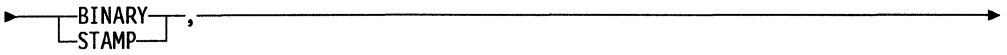
GETIME—Retrieve System Time

The GETIME macro either loads a specified register with the number of seconds since midnight (in binary) or moves all or some segments of the 20-byte system time and date stamp to a specified location.

Syntax



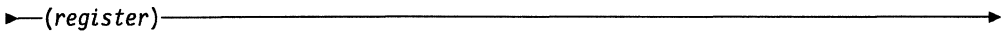
Parameters



Function BINARY indicates that the interval timer is to be loaded into a specified register. STAMP indicates that the system time and date are to be moved to a specified location.

Format BINARY or STAMP.

Default None.



Function Specifies a register into which the interval timer is to be loaded or points to a location to contain the time and date stamp.

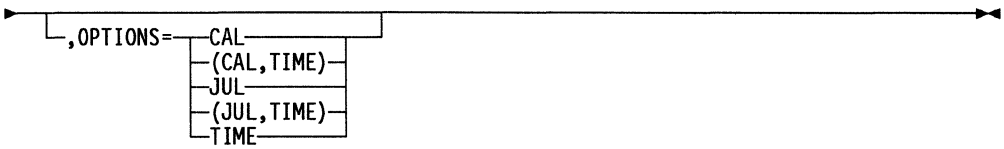
Format Register notation.

Default None.

Remarks If STAMP is specified, register 1 is not allowed.

Register 1 is modified by the macro expansion.

Register 0 is not allowed.



Function Specifies which segments of the 20-byte system time and data stamp are required. JUL means yy.ddd, where yy is the last 2 digits of the year and ddd is the day of the year.

CAL means mm/dd/yy, where mm is the month of the year, dd is the day of the month, and yy is the last 2 digits of the year. The format may also be dd/mm/yy or yy/mm/dd depending on what was specified at NCP generation.

TIME means hh.mm.ss, where hh is the hour, mm is the minute, and ss is the second.

JUL,TIME, means yy.dd.hh.mm.ss.

CAL,TIME means mm/dd/yy (dd/mm/yy,yy/mm/dd), hh.mm.ss.

Format JUL; CAL; TIME; JUL,TIME; or CAL,TIME.

Default The complete 20-byte time and date stamp is moved.

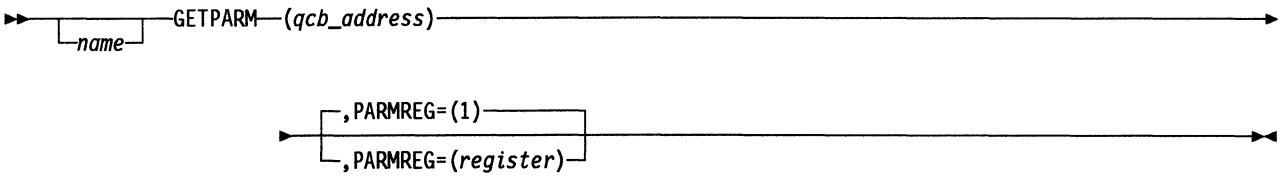
Remarks If you specify BINARY, this keyword is ignored.

GETPARM—Get a Pointer to a Parameter List or Byte Parameter

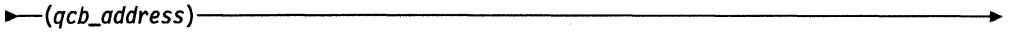
The GETPARM macro is used by block handling routines (BHRs) to get a pointer to a parameter list or a pointer to a byte parameter.

Register 0 is not allowed for register parameters.

Syntax



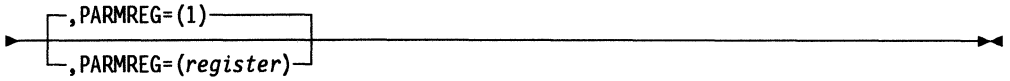
Parameters



Function Specifies the address of the queue control block (QCB) from which the current BHR task is being dispatched.

Format Register notation.

Default None.



Function Specifies the register into which either the pointer to the parameter list or the byte parameter is to be placed.

Format Register notation.

Default Register 1.

Remarks If a BHR has a byte parameter associated with it instead of a pointer to a parameter list, the byte parameter goes into the low-order byte of the register.

GETPT—Get a Block Handler Execution Point

The GETPT macro is used by block handling routines (BHRs) to get a bit configuration that represents the point of execution (1, 2, or 3). The points of execution are as follows:

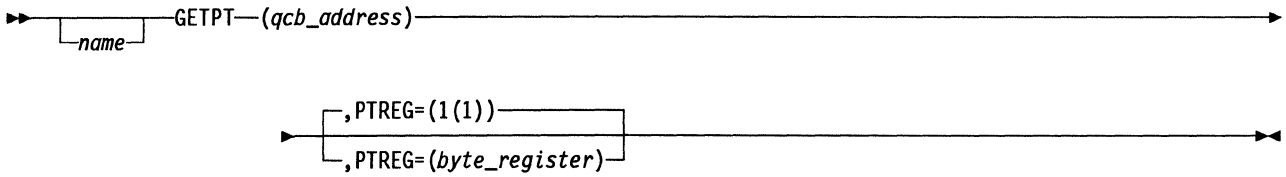
Point 1 After a request for the device is received but before the line is available

Point 2 After a request for the device is received and after the line is available

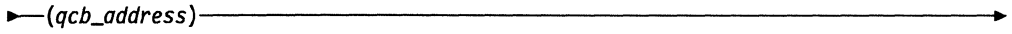
Point 3 After an input operation ends and after the line is released for use with another station.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register that contains the address of the queue control block (QCB) from which the current BHR task is being dispatched.

Format Register notation.

Default None.

Function Specifies the register byte into which the bit configuration representing the point of execution is to be placed.

Format Byte register notation.

Default 1(1).

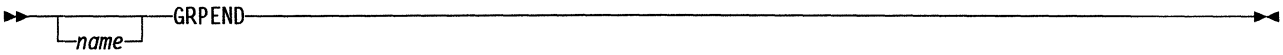
Remarks The bit configuration returned is as follows:

Point 1	01.. ....
Point 2	10.. ....
Point 3	11.. ....

GRPEND—Indicate the End of a Line Group in the Block Dump Table

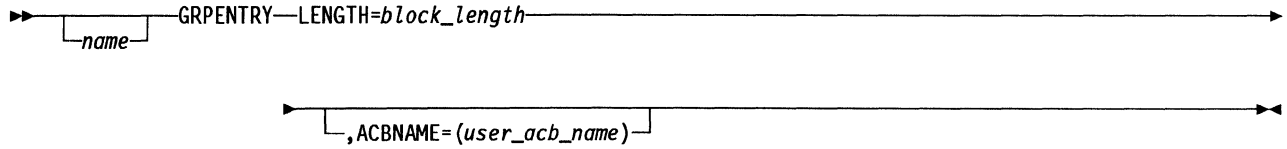
The GRPEND macro indicates the end of the description of one line group in the block dump table (BDT). It sets the last 8-byte BDT entry to 0.

Syntax

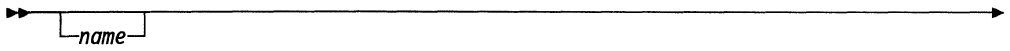


GRPENTRY—Define the First Entry in the Block Dump Table

The GRPENTRY macro defines the first entry in the block dump table (BDT).



Parameters

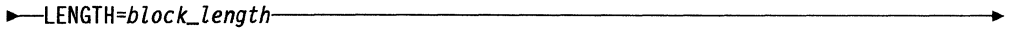


Function Referred to by the ORIGIN keyword of a following (and dependent) BLKENTRY macro.

Format Label notation.

Default None.

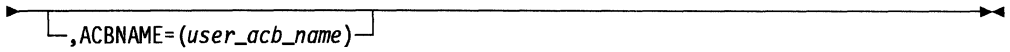
Remarks The label (name) is the name given to a macro that describes a parent block that points to a dependent block. In the macro that describes the dependent block, the ORIGIN keyword must contain the label of the parent block macro to complete the chained relationship.



Function Specifies the user adapter control block (UACB) length in bytes.

Format Decimal notation.

Default None.



Function Specifies the UACB name (the same name that was specified in the UACB keyword of a LINE definition statement) to be printed in the formatted dump. The ACB name can be up to 5 characters long.

Format Label notation; character string from 1 to 5 bytes long.

Default UACB.

Remarks The name is truncated if it is longer than 5 characters.

IF—Test for a Condition

The IF macro is used with the THEN, ELSE, and ENDIF macros to form an IF-THEN-ELSE program structure. The IF macro has four types of format: test CL, ZL; branch-on-bit; test-under-mask; and comparison. The IF macro tests for a specified condition and, if the test is true, passes control to the routine defined by the THEN macro or, if the test is false, passes control to the routine defined by the ELSE macro. The following sections describe how to code logical connectives for multiple-step tests and the four formats of the IF macro.

Logical Connective Evaluation

There are four logical connectives (or logical operators): AND, OR, (AND), and (OR). The different connectives indicate how the Boolean expression is parsed. The Boolean expression is fully parenthesized before execution.

The following examples show how to place parentheses around the logical connectives (AND or OR) in multiple IF macros to perform tests containing two, three, or four parts in various logical groupings.

In these examples, *x*, *y*, and *z* represent either logical connective (AND or OR); this convention lets you match the connectives in the coding samples to the connectives in the tests they perform. A, B, C, and D represent individual tests, such as:

Coding	*Test*
`R4(7),ON`	Bit 7 of register 4 is on.
`R3(0),Z,0F`	Byte 0 of register 3 ANDed with mask X'0F' is zero.
`R5,LT,R7`	The value in register 5 is less than the value in register 7.

Note: Register 0 is not allowed for register parameters.

Code a two-part test as follows:

Code This...	**For This Test**
`IF A,`*x* `IF B`	A *x* B

Code three-part tests as follows:

Code This...	**For This Test**

`IF A,(`*x*`)`
`IF B,`*y*
`IF C`
	(A *x* B) *y* C

`IF A,`*x*
`IF B,(`*y*`)`
`IF C`
	A *x* (B *y* C)

`IF A,(`*x*`)`
`IF B,(`*y*`)`
`IF C`
	A *x* (B *y* C)

Code four-part tests as follows:

Code This...	For This Test
IF A,*x* IF B,*y* IF C,*z* IF D	((A *x* B) *y* C) *z* D
IF A,(*x*) IF B,*y* IF C,*z* IF D	((A *x* B) *y* C) *z* D
IF A,*x* IF B,(*y*) IF C,*z* IF D	(A *x* (B *y* C)) *z* D
IF A,*x* IF B,*y* IF C,(*z*) IF D	(A *x* B) *y* (C *z* D)
IF A,(*x*) IF B,(*y*) IF C,*z* IF D	(A *x* (B *y* C)) *z* D
IF A,(*x*) IF B,*y* IF C,(*z*) IF D	(A *x* B) *y* (C *z* D)
IF A,*x* IF B,(*y*) IF C,(*z*) IF D	A *x* (B *y* (C *z* D))
IF A,(*x*) IF B,(*y*) IF C,(*z*) IF D	A *x* (B *y* (C *z* D))
IF B,(*x*) IF C,*y* IF D,*z* IF A	A *z* ((B *x* C) *y* D) (Notice that you code the individual tests in a different order for this test.)

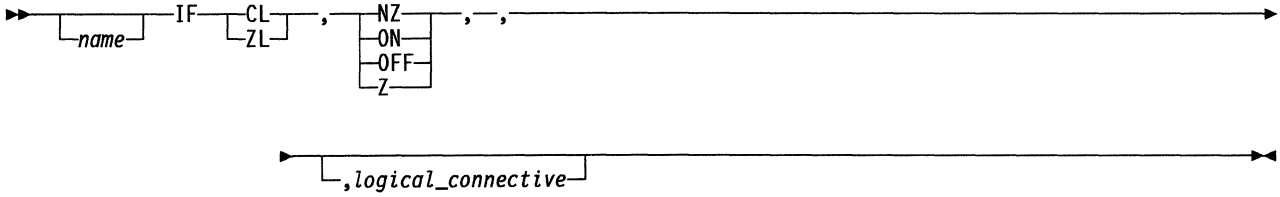
In the following example, the program structure is executed as long as:

- Register 2 is not equal to register 6 *and* register 4 is not equal to register 6 *or*
- The Z condition latch is off.

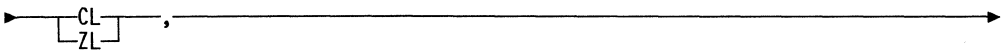
```
IF1      IF R2,NE,R6,H,(AND)
         IF R4,NE,R6,H,OR
         IF ZL,OFF
          .
          .
          .
ENDIF1   ENDIF
```

Test CL, ZL Format

Syntax



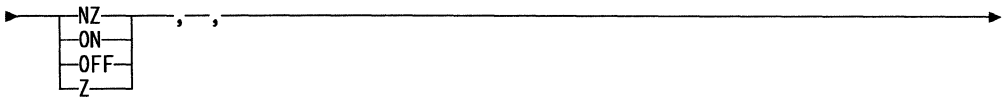
Parameters



Function Specifies that the C condition latch or the Z condition latch is to be tested.

Format CL or ZL.

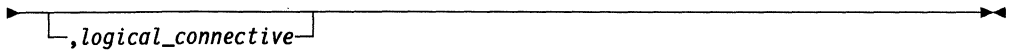
Default None.



Function Specifies the true condition for the specified latch: on, off, zero, or nonzero.

Format ON, OFF, Z, or NZ.

Default None.



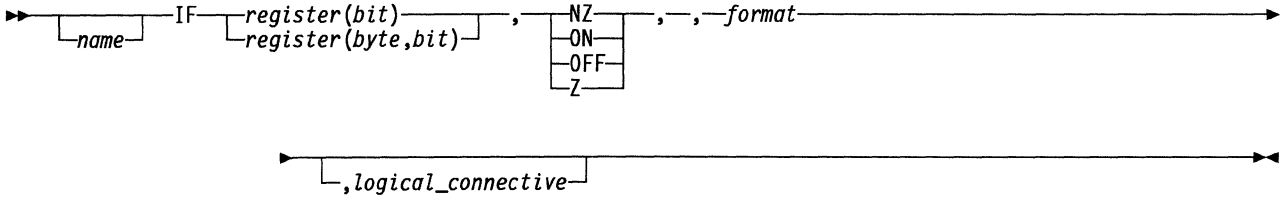
Function Specifies the Boolean logic operation used to connect this IF macro with a following IF macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 176.

Format AND, OR, (AND), or (OR)

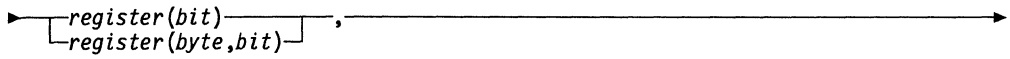
Default No logical connection is made between this IF macro and the following IF macro.

Branch-on-Bit Format

Syntax



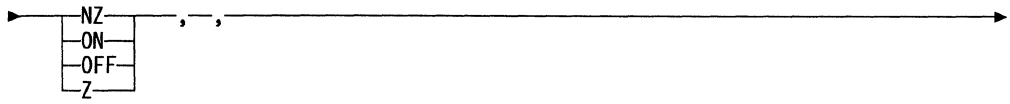
Parameters



Function Specifies the odd-numbered byte or halfword register containing the bit to be tested. If you specify a byte register, you can specify a bit from 0 to 7, and you must code B for the *format* keyword. If you specify a halfword register, you can specify a bit from 0 to 15, and you must code H for the *format* keyword.

Format Byte or halfword register notation.

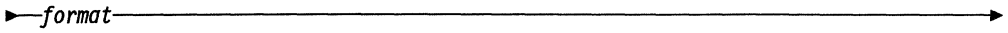
Default None.



Function Specifies the true condition of the specified bit: on, off, zero, or nonzero.

Format ON, OFF, Z or NZ.

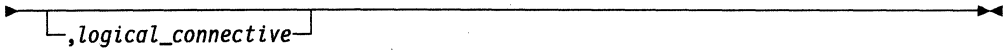
Default None.



Function Specifies whether the register is a byte (B) or halfword (H) register.

Format B or H.

Default None.



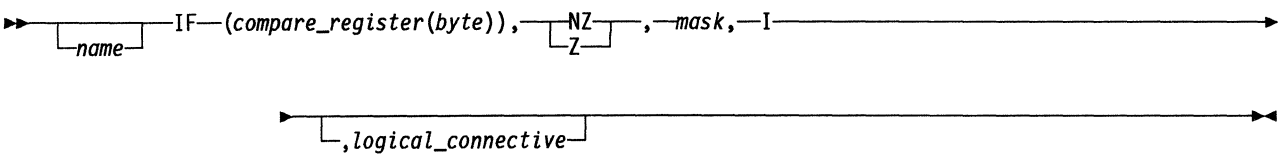
Function Specifies the Boolean logic operation used to connect this IF macro with a following IF macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 176.

Format AND, OR, (AND), or (OR)

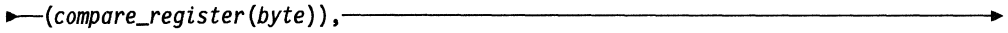
Default No logical connection is made between this IF macro and the following IF macro.

Test-Under-Mask Format

Syntax



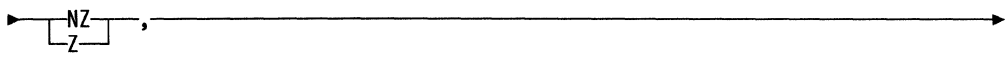
Parameters



Function Specifies the odd-numbered byte register, the contents of which are used in a logical AND comparison with the mask keyword.

Format Byte register notation.

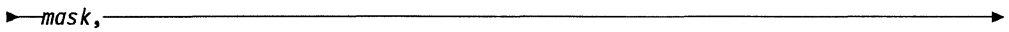
Default None.



Function Specifies the result of a logical AND comparison between the byte register contents and the mask keyword. Z is true if all the byte register bits corresponding to the mask bits are 0. NZ is true if one or more of the byte register bits corresponding to the mask bit are nonzero.

Format Z or NZ.

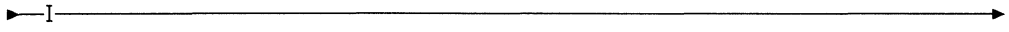
Default None.



Function Specifies an immediate byte value to be used in a logical AND comparison with the contents of the specified byte register.

Format Label notation.

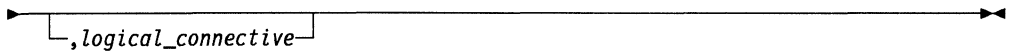
Default None.



Function Specifies, with Z or NZ, the test under mask format.

Format I.

Default None.



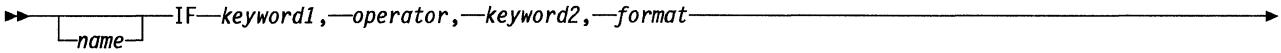
Function Specifies the Boolean logic operation used to connect this IF macro with a following IF macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 176.

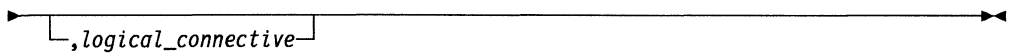
Format AND, OR, (AND), or (OR)

Default No logical connection is made between this IF macro and the following IF macro.

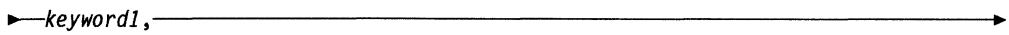
Comparison Format

Syntax





Parameters



Function Specifies the first value for the comparison test.

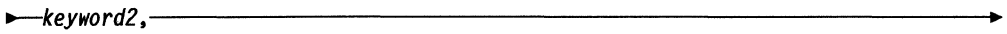
Format Byte, halfword, or fullword (22-bit) register notation.

Default None.

Function Specifies the type of comparison to be made between *keyword1* and *keyword2*: greater than (GT), less than (LT), equal to (EQ), not equal to (NE), not greater than (NG), not less than (NL), greater than or equal to (GE), or less than or equal to (LE).

Format GT, LT, EQ, NE, NG, NL, GE, or LE.

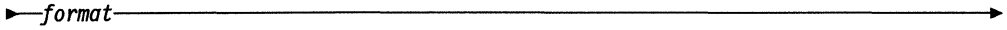
Default None.



Function Specifies the second value for the comparison test.

Format Byte, halfword, or fullword (22-bit) register notation, or if the format keyword is specified as I, label notation.

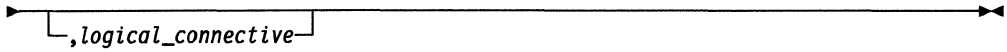
Default None.



Function Specifies whether *keyword1* and *keyword2* are byte registers (B), halfword registers (H), or fullword (22-bit) registers (F). Code I to specify that *keyword1* is a byte register and *keyword2* is a byte immediate value.

Format B, H, F, or I.

Default None.



Function Specifies the Boolean logic operation used to connect this IF macro with a following IF macro and the order in which keywords are paired when the Boolean expression is evaluated. There are two types of logical connectives, those with parentheses and those without. See the description under "Logical Connective Evaluation" on page 176.

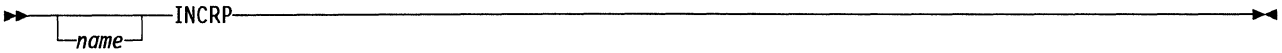
Format AND, OR, (AND), or (OR).

Default No logical connection is made between this IF macro and the following IF macro.

INCRP—Increment the Response Phrase in the BTU System Response Field

The INCRP macro increments the response phase currently in the BTU system response field (BCUSRES). This macro can be used in program level 5 and requires that the BCU DSECT XXCXTBCU be present and addressability be established.

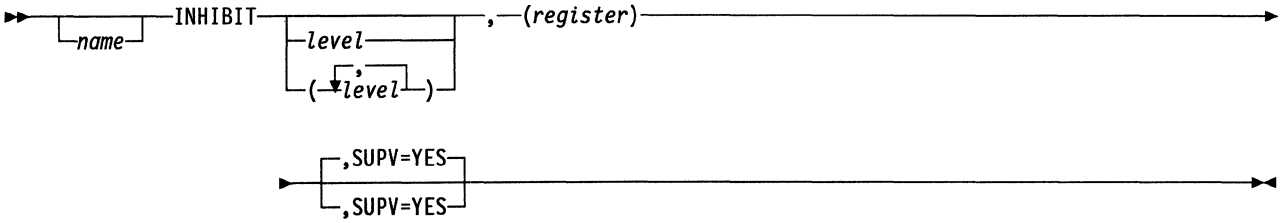
Syntax



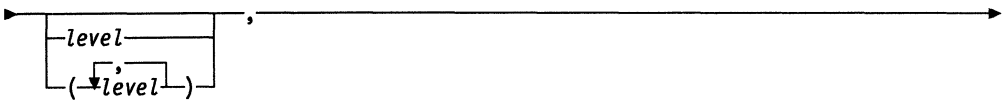
INHIBIT—Inhibit Interrupts

The INHIBIT macro loads a specified register with the appropriate masks and issues an output instruction for that register to inhibit certain interrupts at the specified interrupt levels (1, 2, 3, or 4).

Note: This macro can be executed by interrupt-level code only.



Parameters



Function Specifies the level or levels to be masked.

Format Absolute notation. If more than one level is to be masked, the list of levels is enclosed in parentheses.

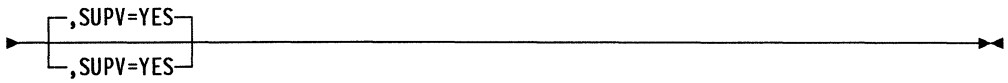
Default The supervisor assumes that the register has been preloaded with the appropriate inhibit mask.

Function Specifies a register that is to contain (or already contains) the mask.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



Function Specifies that the issuer is running in an interrupt level.

Format YES.

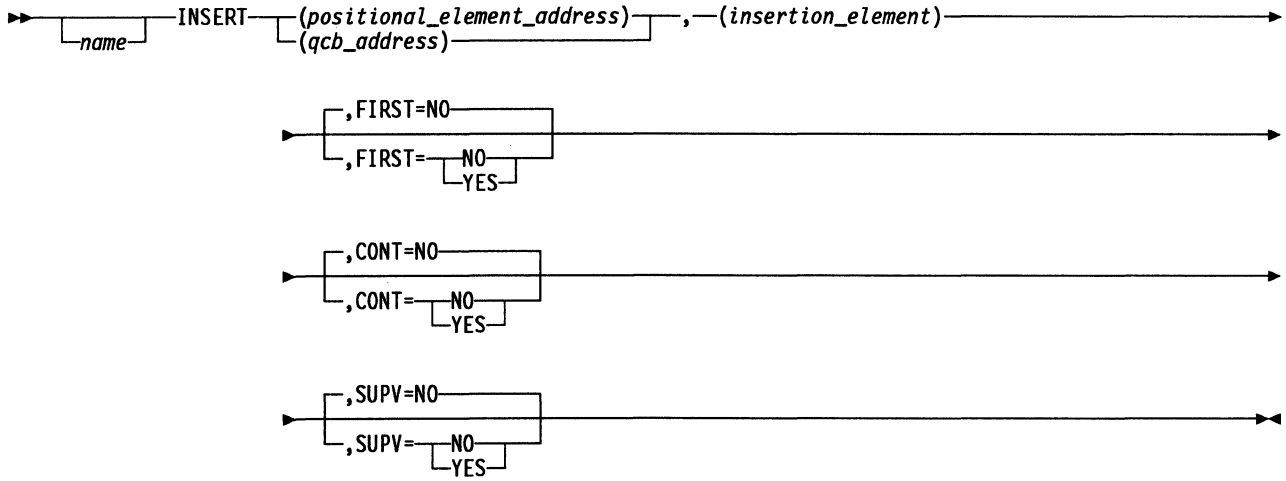
Default YES.

INSERT—Insert an Element into a System Queue

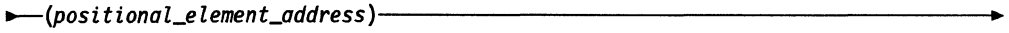
The INSERT macro attaches an element, which can be a block control unit (BCU), a path information unit (PIU), or a queue control block (QCB), to a system queue in any specified position except the last. Use the ENQUE macro instruction to attach an element to the last position in a system queue.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register containing the address of the element that the given element is to be inserted after (FIRST=NO).

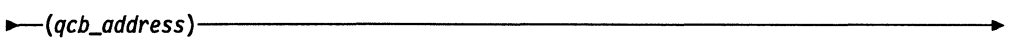
Format Register notation.

Default None.

Remarks If FIRST=NO, this keyword is required.

Register 1 is not allowed.

If SUPV=YES, register 3 is standard and register 6 is not allowed.



Function Specifies the register containing the address of the QCB governing the queue in which a new element is to be inserted in the *first* position (FIRST=YES).

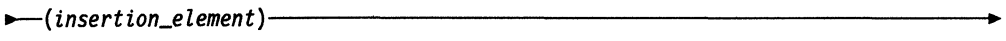
Format Register or label notation.

Default If SUPV=YES and FIRST=YES, there is no default. If SUPV=NO and FIRST=YES, the QCB for the queue that activated the issuing task is used.

Remarks This keyword is required if FIRST=YES.

Register 1 is not allowed.

If SUPV=YES, register 2 is standard and register 6 is not allowed.



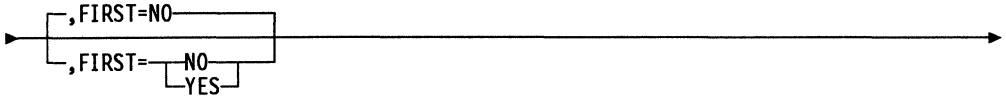
Function Specifies the register containing the address of the element to be inserted into the queue.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

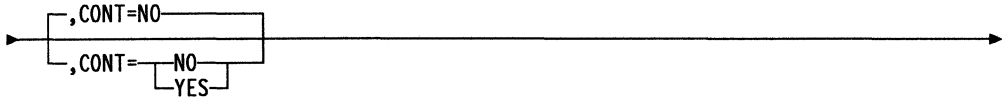
If SUPV=YES, register 4 is standard and register 6 is not allowed.



Function Specifies whether the new element is to be inserted in the first position of the specified queue.

Format YES or NO.

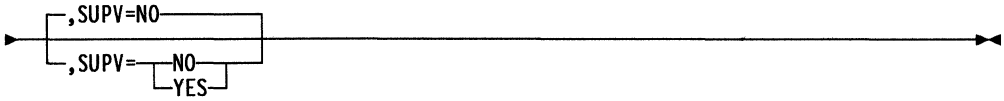
Default NO.



Function Specifies whether the queue is a contention queue (one that can be simultaneously manipulated by two or more interrupt levels).

Format YES or NO.

Default NO.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level. This information is needed so that the proper type of supervisor macro can be issued.

Format YES or NO.

Default NO.

Remarks If SUPV=NO is coded, an EXIT instruction is generated.

IOHM—Issue an Input/Output Halfword Instruction

The IOHM macro issues an adapter input/output halfword (IOH) instruction for the communication scanner processor (CSP) and stores the pending command and parameter status area (PSA) modifiers (if specified) in the PSA.

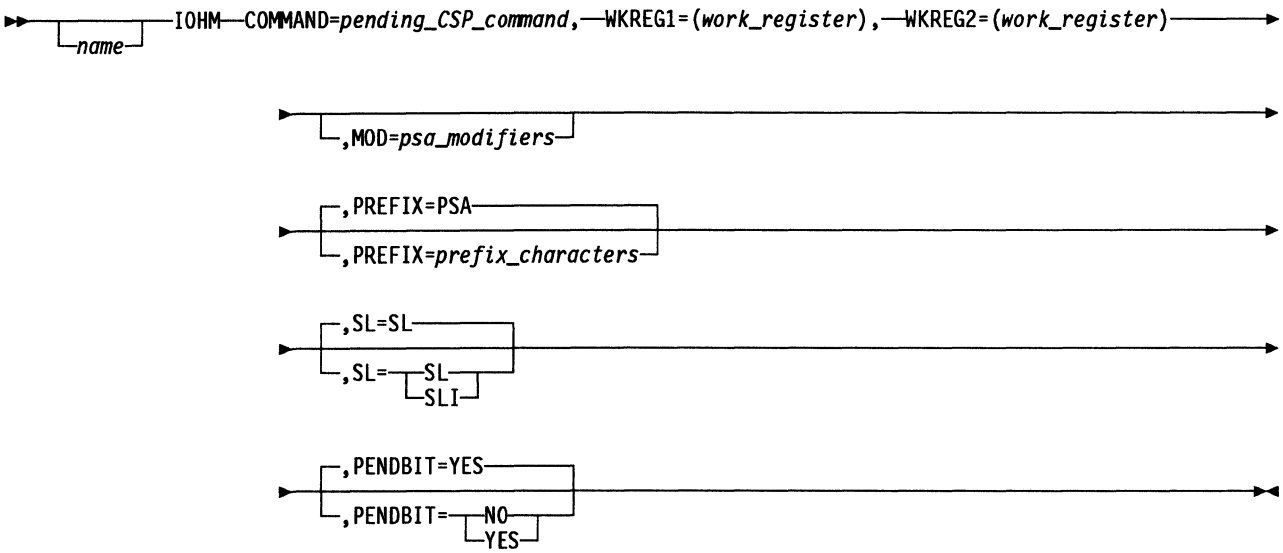
When an IOH command is issued, the high-order bit of the PSA command field is set to 1 to indicate that the command is pending. When a level 2 interrupt occurs, the bit is reset to 0. Commands that set the high-order bit themselves are treated as special cases. These include commands X'F0' to X'F2' and X'F5'.

For these X'Fx' commands, you must use the PENDBIT keyword to ensure that the high-order bit is not reset for these commands. For other commands, the PENDBIT keyword defaults to YES (pending bit can be used).

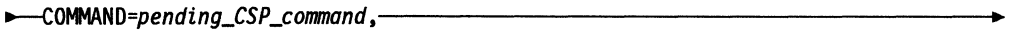
See *Principles of Operation* for your controller for more information about IOH commands.

Register 0 is not allowed for register parameters.

Syntax



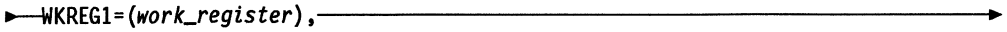
Parameters



Function Specifies the pending command.

Format Register or label notation or hexadecimal value. You cannot specify the HALT and HALT IMMEDIATE commands with register notation.

Default None.

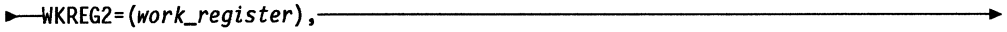


Function Specifies the R1 register for IOH macro call.

Format Register notation.

Default None.

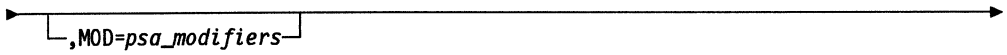
Remarks Must be an odd-numbered general register when SL=SL and any general register when SL=SLI.



Function Specifies the R2 register for IOH macro call. This can be any odd-numbered general register.

Format Register notation.

Default None.



Function Specifies a byte or initial control word (ICW) field to be changed by the CSP. If you code this keyword, the modifiers are stored in the PSA.

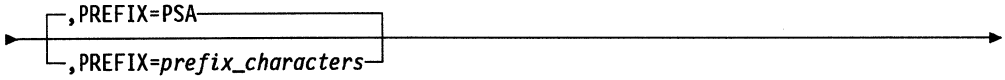
Format Register or label notation.

Default None.

Remarks Must have addressability to the AXB and the PSA.

Mnotes

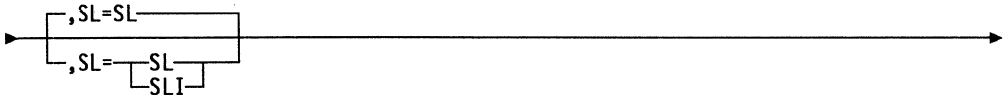
```
MNOTE 12,'IOHM MACRO REQUIRES COMMAND, WKREG1 AND WKREG2.'
MNOTE 12,'IOHM MACRO REQUIRES &SL TO BE SL OR SLI.'
```



Function Specifies the prefix used when XCXTPSA was included.

Format PSA or any 3 character prefix.

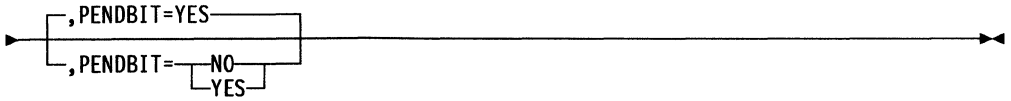
Default PSA.



Function Specifies START LINE (SL) or START LINE INITIAL (SLI).

Format SL or SLI.

Default SL.



Function Specifies whether the command pending bit can be used for the command that is about to be issued.

Format YES or NO.

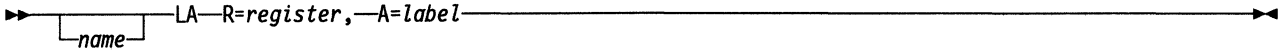
Default YES.

Remarks Commands X'F0' to X'F5' require PENDBIT=NO.

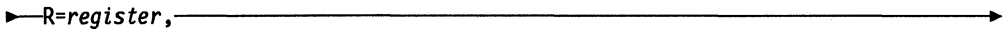
LA—Generate the Load Address Instruction

The LA macro generates the LA instruction as a series of define storage (DS) and define constant (DC) assembler definitions.

Syntax



Parameters

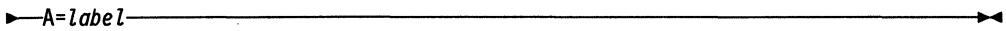


Function Specifies the register that is to contain the specified address.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



Function Specifies the label of the location whose address is to be loaded into the register.

Format Label notation.

Default None.

LASTUACB—No Longer Functional

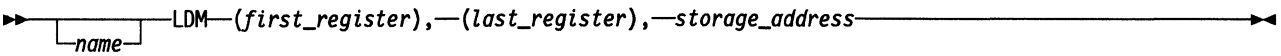
The LASTUACB macro is no longer functional. It has been replaced by function internal to the NCP/EP definition facility (NDF). If you have user-written code that uses the LASTUACB macro, it will not cause any errors.

LDM—Load a Series of Registers

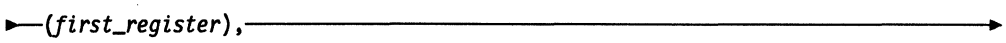
The LDM macro sequentially loads a specified series of registers, starting from a specified area of storage. All 22 bits of each affected register are loaded directly without any shifting, and storage is accessed in 4-byte increments.

Register 0 is not allowed for register parameters.

Syntax



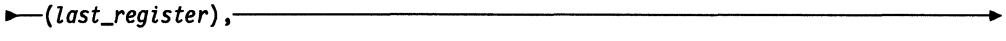
Parameters



Function Specifies the first register in a series to be loaded.

Format Register notation.

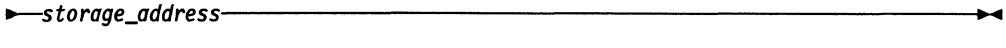
Default None.



Function Specifies the last register in a series to be loaded.

Format Register notation.

Default None.



Function Specifies the storage location from which the first register is loaded.

Format Label or pseudobase displacement; that is, in the format (displacement, base) notation.

Default None.

LEASE—Allocate a Buffer, a Buffer Chain, or a Unit of Storage

The LEASE macro has three forms:

- Normal use, SUPV=YES
- Normal use, SUPV=NO
- Character service use.

The normal-use forms of the LEASE macro allocate a buffer or buffer chain of storage to the requesting program and return the address of the buffer or buffer chain. Both the buffer counts and the buffer offsets are set to 0 in all buffers.

The character service form of the LEASE macro does the following:

- Allocates a unit of storage
- Detects and services buffer threshold allocation crossovers
- Builds a buffer chain.

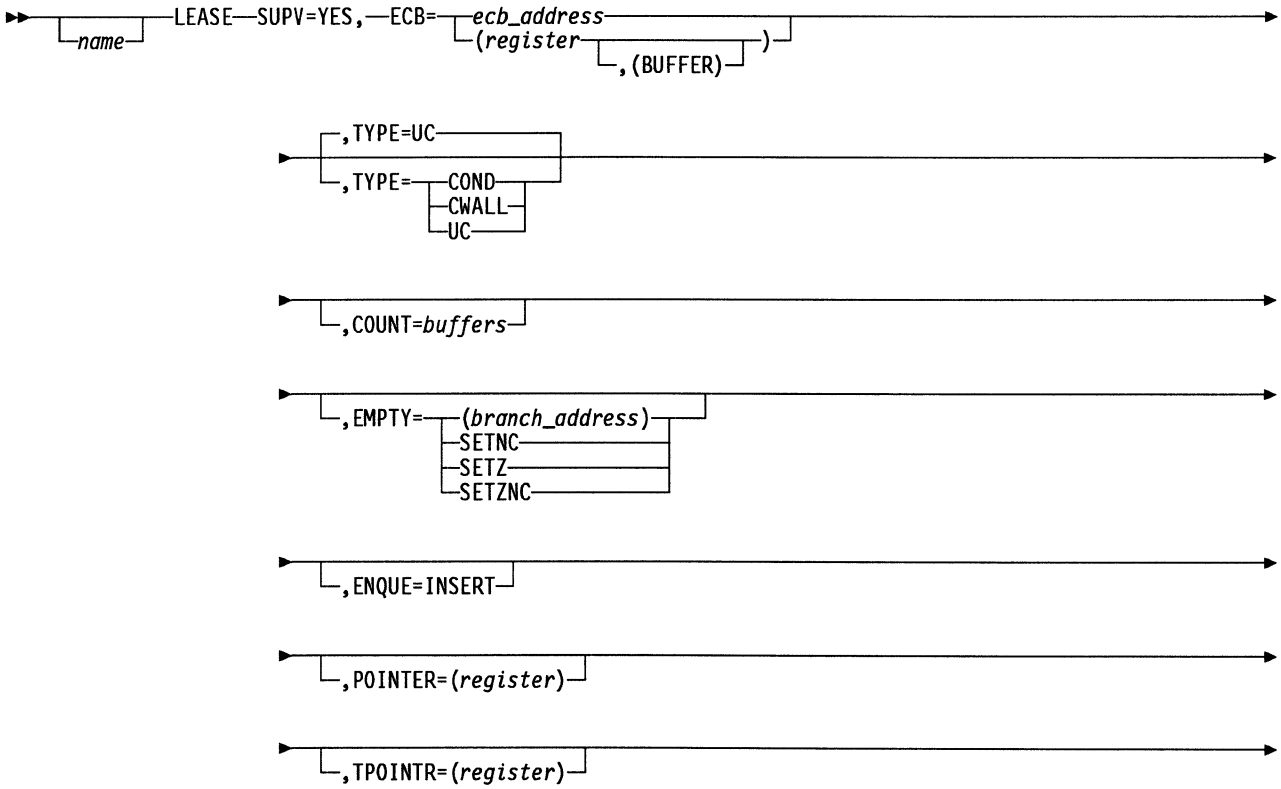
The character service form of the macro allocates only one buffer at a time.

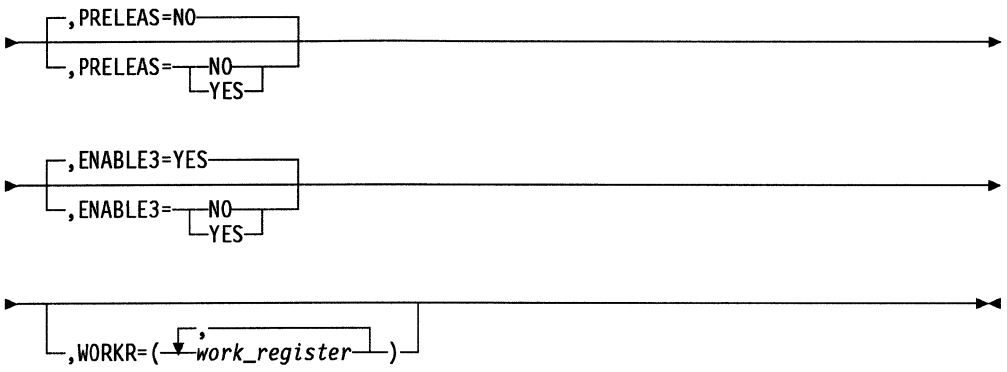
Register 0 is not allowed for register parameters.

Note: This macro uses the current save area and assumes that this save area can be overwritten.

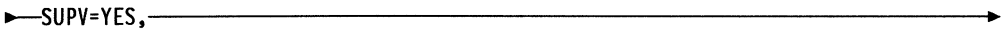
Normal Use, SUPV=YES

Syntax





Parameters

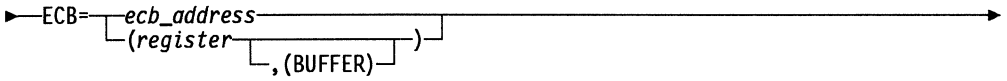


Function SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES.

Default None.

Remark: You must use SUPV=YES when coding in program levels 3 or 4.



Function Specifies the address of the event control block (ECB) governing buffer allocation.

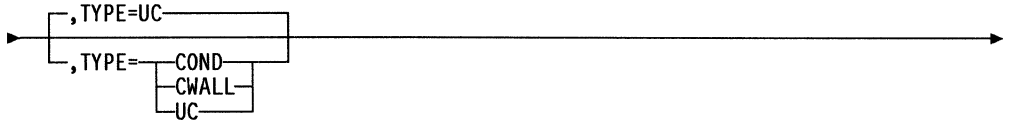
Format Absolute register or label notation.

Default None.

Remarks Do not specify register 1. LEASE processing alters its contents.

When SUPV=YES, register 3 is standard and register 6 is not allowed.

If *register* points to a previously allocated buffer that contains the ECB, you must specify BUFFER.



Function Specifies whether the lease request is unconditional, conditional, or limited by the communication wall.

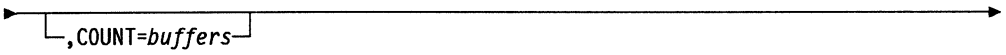
Format UC, COND, or CWALL.

Default UC.

Remarks Unconditional requests (TYPE=UC) are satisfied if buffers are available in the buffer pool. No partial allocation is made for unconditional leases. If the number of buffers requested is not available, the lease is not satisfied.

Conditional requests (TYPE=COND) are satisfied if the system is not in slowdown. If the system is in slowdown, the lease is unsuccessful. If the system is not in slowdown, and there are not enough buffers to completely satisfy the request without entering slowdown (the free buffer count is less than the slowdown entry threshold), a partial allocation of buffers is made. A partial allocation puts NCP into slowdown because one more buffer is used to send an enter slowdown message to the host. This puts the free buffer count to one below the slowdown entry threshold (into slowdown). A partial allocation of buffers is considered successful.

A communication wall request (TYPE=CWALL) is not satisfied if the system is in the CWALL state. If the request will cause the system to pass the CWALL threshold, only the number of buffers required to pass the threshold is returned; otherwise, the request is satisfied. A partial allocation of buffers is considered successful.

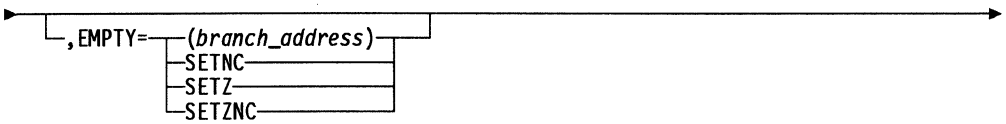


Function Specifies the total number of buffers required.

Format Byte register, label, or absolute notation.

Default The byte register 1(0) is used and is set to one.

Remarks If the request is partially satisfied, the residual count will be in the byte register specified.



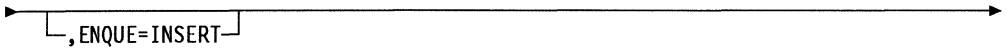
Function Either specifies the address to be given control if the request is not satisfied, or specifies that the C latch or Z latch is to indicate whether the request is satisfied.

Format Label notation, register notation, SETZNC, SETZ, or SETNC.

Default The issuing program tests register 1(1), bit 0 or the register specified by the POINTER keyword. If nonzero, the request is satisfied.

Remarks Do not specify register 1.

For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the request is not satisfied or to 0 if the request is satisfied. For SETZNC or SETNC, the C latch is set to 0 if the request is not satisfied or to 1 if the request is satisfied.

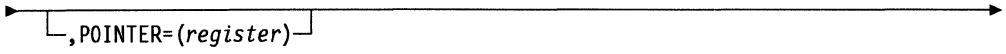


Function Specifies whether an unconditional request for buffers is to disable lower priority interrupt levels if the request cannot be satisfied at present.

Format INSERT.

Default None.

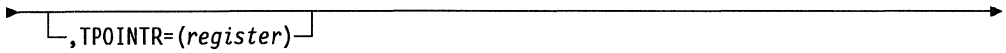
Remarks Use ENQUE=INSERT only when coding unconditional requests (TYPE=UC) in program level 4.



Function Specifies a register to contain the address of the allocated buffer or buffer chain.

Format Register notation.

Default The issuing program loads the allocated buffer address from the ECB, offset by 8.

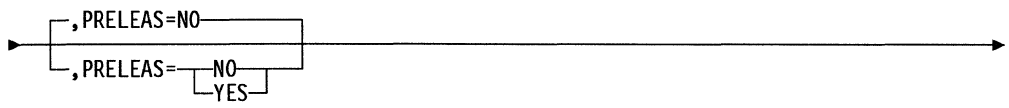


Function Specifies a register to contain the address of the last buffer in the chain allocated. If only one buffer is allocated, the address is the same as the address returned in the register specified for POINTER.

Format Register notation.

Default The issuing program loads the allocated buffer address from the ECB, offset by 0.

Remarks Use this parameter only when you code the POINTER parameter. TPOINTR is not valid when ECB=(*register*,BUFFER) is coded.

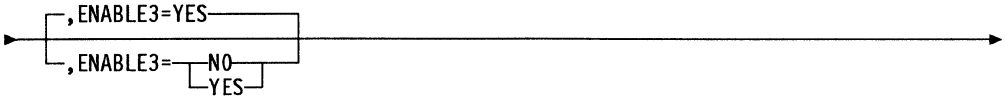


Function Specifies whether the lease routine is to use preleased buffers, if available.

Format YES or NO.

Default NO.

Remarks Use this parameter only when a level 5 routine has preleased buffers specifically for the use of the supervisory routine.



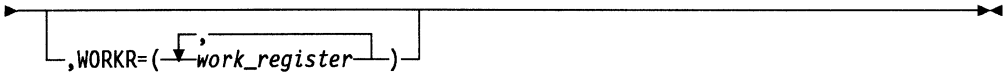
Function Specifies whether level 3 interrupts are to be enabled at the completion of the lease.

ENABLE3=YES allows both level 2 and level 3 interrupts to be enabled at the completion of the lease. ENABLE3=NO allows a level 4 routine to inhibit level 3 interrupts and to issue a lease. Only level 2 interrupts are enabled at the completion of the LEASE when ENABLE3=NO is coded.

Format YES or NO.

Default YES.

Remarks Use this parameter only when SUPV=YES.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

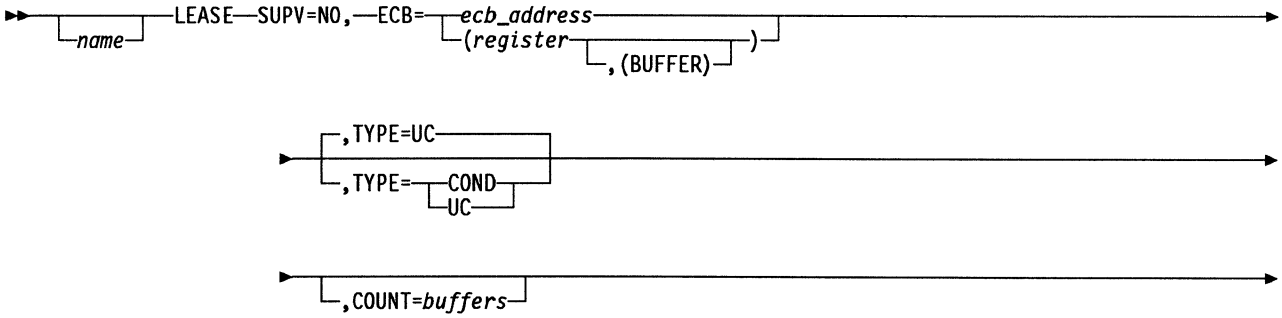
Format Register notation.

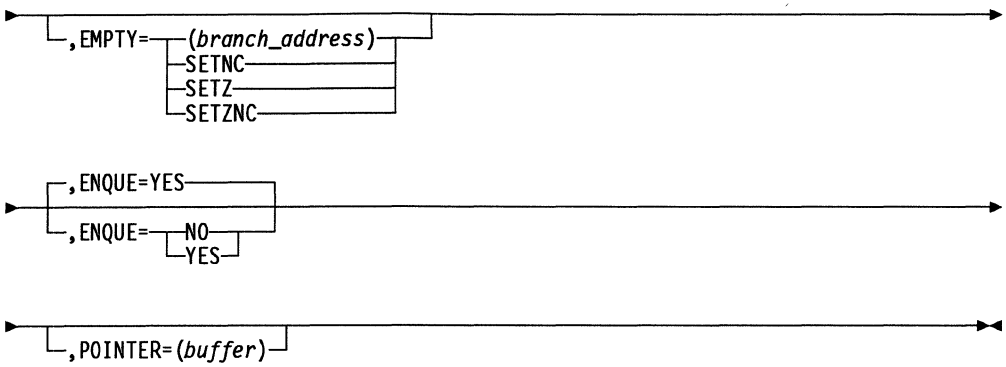
Default No work registers.

Remarks Do not use register 6.

Normal Use, SUPV=NO

Syntax





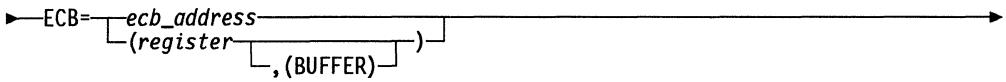
Parameters

Function SUPV=NO specifies that the issuer is running in level 5.

Format NO.

Default None.

Remarks You must use SUPV=NO when coding in program level 5.



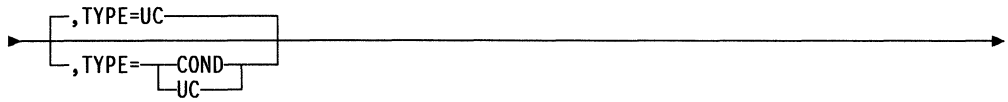
Function Specifies the address of the event control block (ECB) responsible for buffer allocation.

Format Absolute register or label notation.

Default None.

Remarks Do not specify register 1. Its contents are altered by LEASE processing.

If a register points to a previously allocated buffer that contains the ECB, specify BUFFER.



Function Specifies whether the lease request is unconditional or conditional.

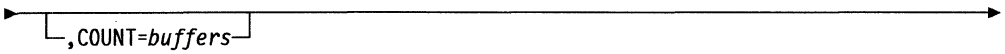
Format UC or COND.

Default UC.

Remarks Unconditional requests (TYPE=UC) are satisfied if buffers are available in the buffer pool. Partial allocation is not made for unconditional leases. If the number of buffers requested is not available, the lease is not satisfied.

Use TYPE=UC for leases of a critical nature, such as those that could cause deadlock of NCP or associated resources; otherwise, use TYPE=COND.

Conditional requests (TYPE=COND) are satisfied if the system is not in slowdown. If the system is in slowdown, the lease is unsuccessful. If the system is not in slowdown, and there are not enough buffers to completely satisfy the request without entering slowdown (free buffer count is less than slowdown entry threshold), a partial allocation of buffers is made. A partial allocation puts NCP into slowdown because one more buffer is used to send an enter slowdown message to the host. This puts the free buffer count to one below the slowdown entry threshold (into slowdown). A partial allocation of buffers is considered successful.

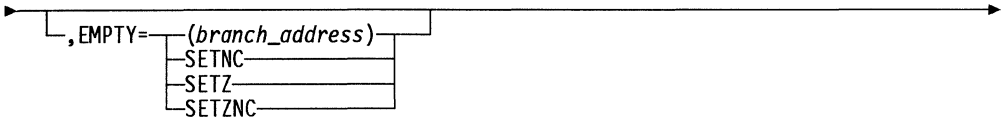


Function Specifies the total number of buffers required.

Format Byte register, label, or absolute notation.

Default The byte register 1(0) is used and is set to one.

Remarks If the request is partially satisfied, the residual count is in the byte register specified.



Function Either specifies the address to be given control if the request is not satisfied, or specifies that the C latch or Z latch is to indicate whether the request is satisfied.

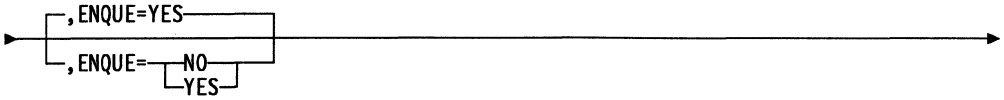
Format Label notation, register notation, SETZNC, SETZ, or SETNC.

Default The issuing program tests register 1(1), bit 0 or the register specified by the POINTER keyword. If nonzero, the request is satisfied.

Remarks Do not specify register 1.

For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the request is not satisfied or to 0 if the request is satisfied. For SETZNC or SETNC, the C latch is set to 0 if the request is not satisfied or to 1 if the request is satisfied.

This parameter is ignored when SUPV=NO, if TYPE=UC and ENQUE=YES.



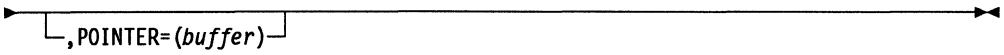
Function Specifies whether an unconditional request for buffers is to disable levels 4 and 5 if the request cannot be satisfied.

Format YES or NO.

Default YES.

Remarks ENQUE=YES is not recommended. The preferable method of enqueuing a task until buffers are available is through the PRELEASE macro with POST=YES. This allows other tasks that do not require buffers to execute. See PRELEASE macro on page 270 for more information.

This parameter is ignored when SUPV=NO and TYPE=COND.



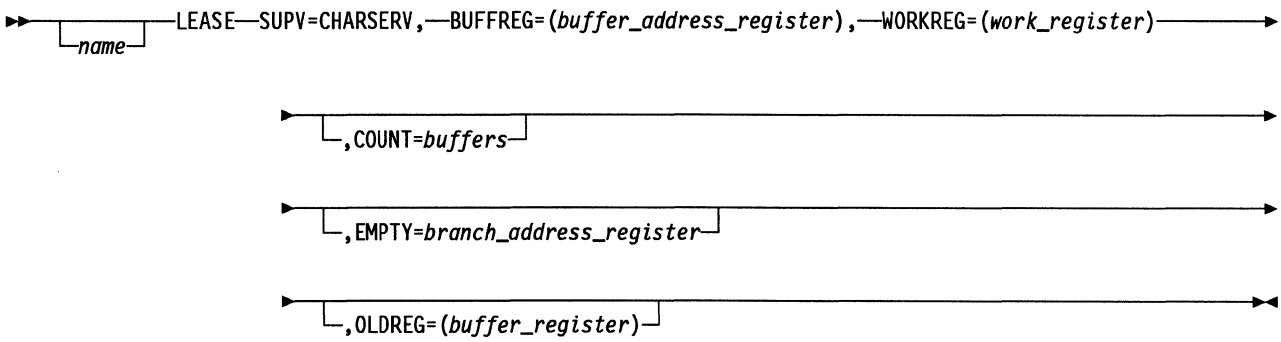
Function Specifies a register to contain the address of the allocated buffer or buffer chain.

Format Register notation.

Default The issuing program loads the allocated buffer address from the ECB, offset by 8.

Character Service Use

Syntax



Parameters

```
►—SUPV=CHARSERV,————————————————————————————————————————————►
```

Function Specifies the character service form of the macro. This form allocates a unit of storage, detects and services buffer threshold allocation crossovers, and builds a buffer chain. The character service form allocates only one buffer at a time, and is only for leases issued in level 2.

Format CHARSERV.

Default None.

Remarks This form of the LEASE macro generates inline code.

Use SUPV=CHARSERV only for leases issued in level 2. Other use could cause programming errors.

The lease is successful if the system is not in CWALL state.

If SUPV=CHARSERV is coded, dsect XCXTABN must be included in the assembly.

```
►—BUFFREG=(buffer_address_register),————————————————————————►
```

Function Specifies a register in which the address of the allocated buffer is returned.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

BUFFREG must be an odd-numbered register.

The address returned is the address of the first byte of the leased buffer.

```
►—WORKREG=(work_register)————————————————————————————————————►
```

Function Specifies a work register, the contents of which may be altered during execution of the macro. This work register is used by the supervisor.

Format Register notation.

Default None.

Remarks The contents of the specified register are not preserved by the macro.

WORKREG must be an odd-numbered register.

```
►————————————————————————————————————————————————————————————►
  └,COUNT=buffers┘
```

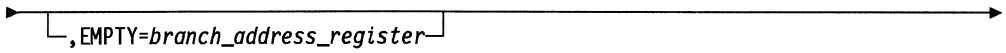
Function Specifies the total number of buffers required.

Format Byte register, label, or absolute notation.

Default The byte register 1(0) is used and is set to one.

Remarks If the request is partially satisfied, the residual count is in the byte register specified.

Do not code both COUNT and OLDREG.



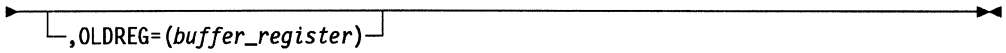
Function Specifies the address to be given control if the buffer allocation is not made.

Format Label or register notation.

Default The next instruction after the macro expansion is given control.

Remarks Register 1 is not allowed.

If you do not code EMPTY, the C and Z latches will indicate whether the request is satisfied. The C latch is set to 1 if the request is satisfied or to 0 if the request is not satisfied. The Z latch is set to 0 if the request is not satisfied.



Function Specifies a register in which the address of the previously allocated buffer is returned.

Format Register notation.

Default The currently allocated buffer is not chained to any buffer chain.

Remarks This keyword causes the LEASE macro to chain the newly allocated buffer to a previously allocated buffer. If the newly allocated buffer is the first in the chain being built for the line, then the OLDREG register is filled with zeros.

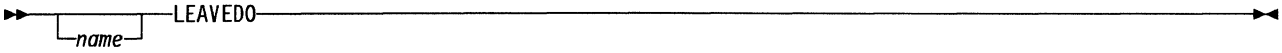
Register 1 is not allowed.

Do not code both COUNT and OLDREG.

LEAVEDO—Exit a DOWHILE or DOUNTIL Program Structure

The LEAVEDO macro is used in conjunction with the DOWHILE, DOUNTIL and ENDDO macros to form DO loop program structures. LEAVEDO generates a branch to the location following the ENDDO macro for the current DO loop. Where DO loops are nested, LEAVEDO branches to the ENDDO for the current next level. The number of LEAVEDO macros that are used within a DO loop must be declared on the first DOWHILE or DOUNTIL macro of the test for the current DO loop.

Syntax

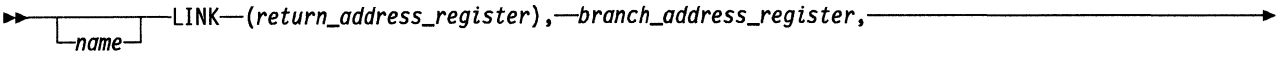


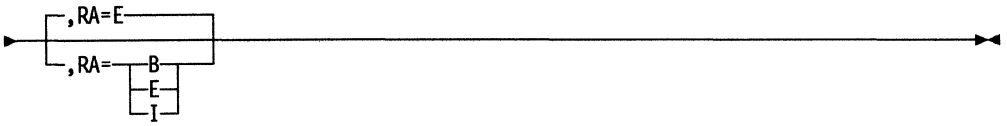
LINK—Store a Return Address and Branch to a Specified Address

The LINK macro stores a return address in the first word of the system save area pointed to by register 6 and transfers control using a branch and link register, branch and link, branch long, or a branch register command to the specified branch address. The routine receiving control must reside in storage that has the same storage protection key as the routine issuing the LINK macro instruction.

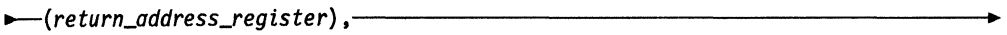
Register 0 is not allowed for register parameters.

Syntax





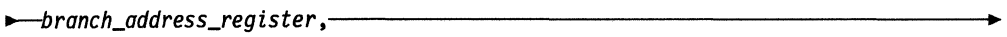
Parameters



Function Specifies the register containing the return address.

Format Register notation.

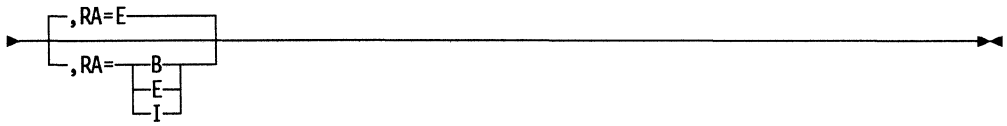
Default None.



Function Specifies the location to which control is to be transferred.

Format Register or label notation.

Default None.



Function Specifies that the return address is inline, saved by the routine to which it is linked, or external.

Format I, B, or E.

Default E.

Remarks If you code I, the LINK macro generates a label following the branch register of a BLG instruction. A load address instruction sets the (*return address register*) keyword as the return address.

If you code B, either a BALR or BAL instruction is generated, and the routine receiving control must save the return address with the SAVE macro.

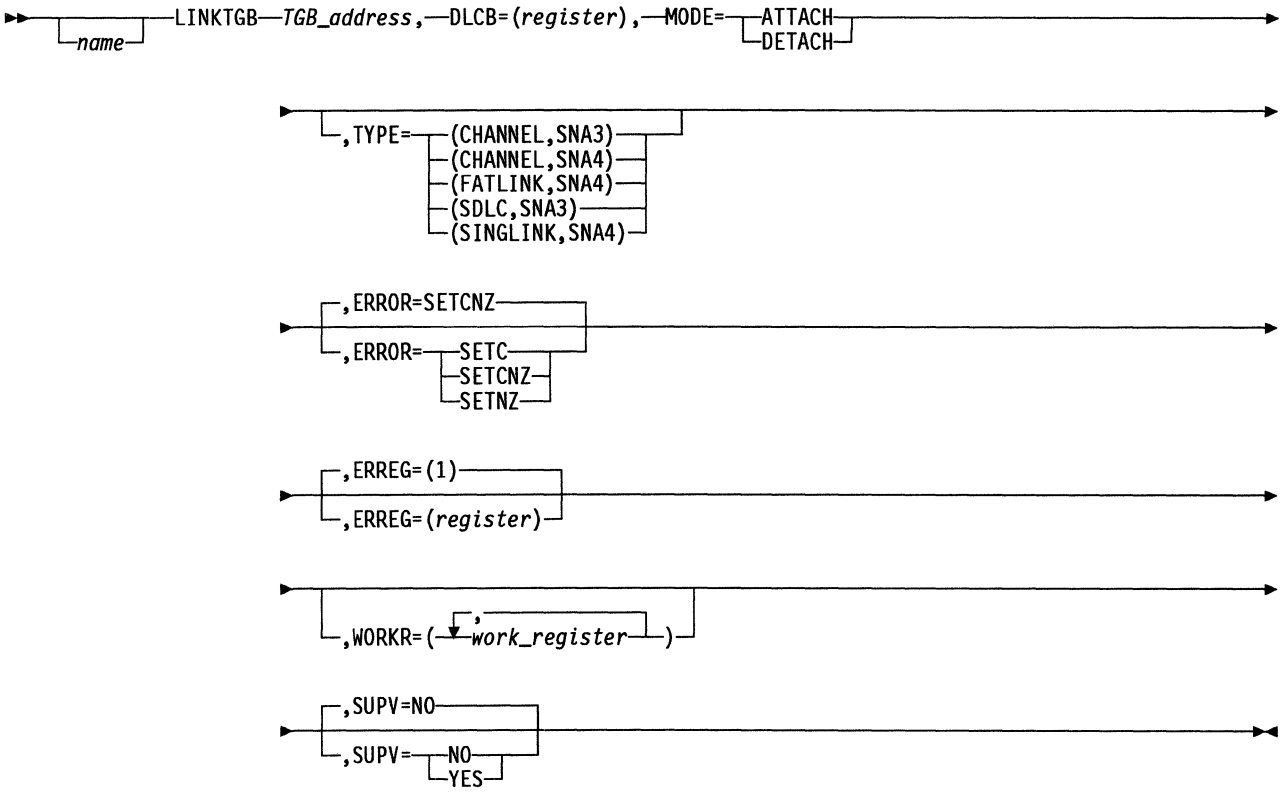
If you code E, the LINK macro generates a BLG instruction. No return address is set up.

LINKTGB—Associate a Data Link Control Block with a Transmission Group Control Block

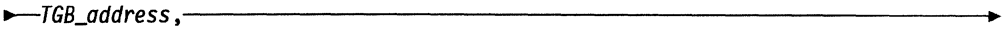
The LINKTGB macro is used to establish and terminate the association of a specific data link control block (SCB, CAB, or FLB) with a specific transmission group control block (TGB).

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Either specifies the register containing the address of the TGB to which the data link control block is to be attached, or specifies the register containing the address of the TGB from which the data link control block is to be detached.

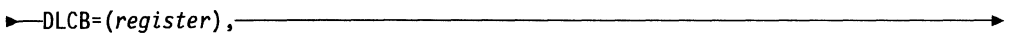
Format Register notation.

Default None.

Remarks If SUPV=NO is coded, register 1 is not allowed.

If SUPV=YES is coded, register 2 is standard and register 6 is not allowed.

The register specified must not be the same as that specified for DLCB, ERREG, or WORKR.



Function Specifies the register containing the address of the station control block (SCB), channel adapter block (CAB), or multilink transmission group control block (FLB).

Format Register notation.

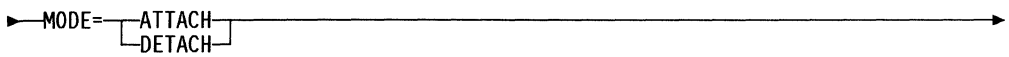
Default None.

Remarks If SUPV=NO is coded, register 1 is not allowed.

If SUPV=YES is coded, register 3 is standard and register 6 is not allowed.

The register specified must not be the same as that specified for the TGB address, ERREG, or WORKR.

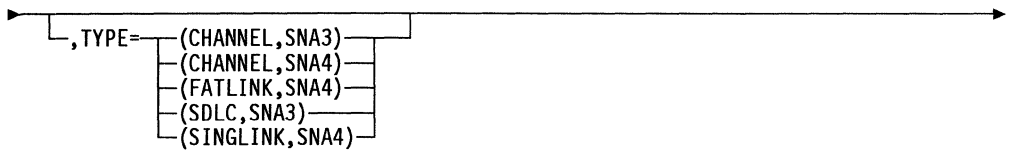
You must specify an SCB, CAB, or FLB; do not specify more than one.



Function Specifies whether to link the control block specified by the DLCB keyword to a TGB or to terminate the current link.

Format ATTACH or DETACH.

Default None.

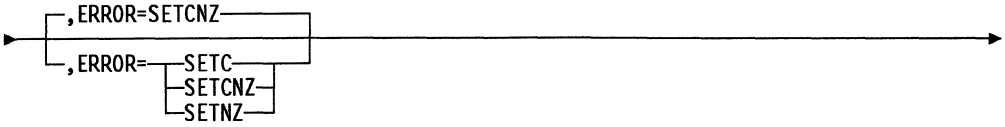


Function Specifies the type of data link control associated with the TGB. It also specifies whether the adjoining subarea node is at SNA 3 level (SNA3) or at SNA 4.2 level (SNA4).

Format (CHANNEL, SNA3), (CHANNEL, SNA4), (FATLINK, SNA4), or (SDLC, SNA3), or (SINGLINK, SNA4).

Default None.

Remarks This keyword is valid only when MODE=ATTACH is coded.



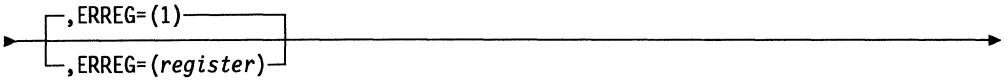
Function Specifies whether the C latch, the Z latch, or both are to indicate whether the requested link can be made. The link will not be made if another data link control block is currently linked to the TGB.

Format SETCNZ, SETC, or SETNZ.

Default SETCNZ.

Remarks This keyword is valid only when MODE=ATTACH is coded.

The C latch is set to 1 if the link cannot be made or to 0 if the link can be made. The Z latch is set to 0 if the link cannot be made or to 1 if the link can be made.



Function Specifies the error-return register that will indicate whether the requested link was made. The link will not be made if another data link control block is currently linked to the TGB.

Format Register notation.

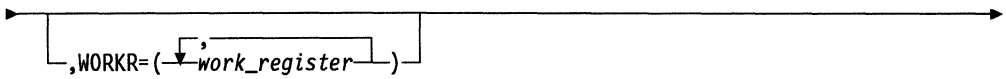
Default 1.

Remarks This keyword is valid only when MODE=ATTACH and SUPV=YES are coded.

Register 1 is standard and register 6 is not allowed.

The register specified must not be the same as that specified for the TGB address or for DLCB.

The register is set to nonzero if the link cannot be made or to 0 if the link can be made.



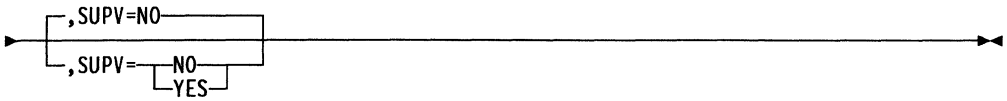
Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Registers 1, 2, 3, 4, 5, and 7 (up to a maximum of 4 registers) can be specified.

Default No work register.

Remarks This keyword is valid only when SUPV=YES is coded.

The registers specified must not be the same as that specified for the TGB address or for DLCB.



Function Specifies the level in which the issuer is running. SUPV=NO indicates that it is being invoked from level 5. SUPV=YES indicates that the macro is to be invoked in an interrupt level.

Format YES or NO.

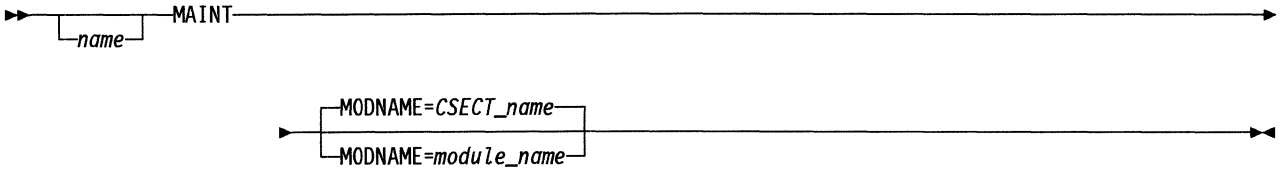
Default NO.

Remarks When SUPV=YES, register 6 must point to a save area that can be overwritten.

MAINT—Supply Maintenance Identification and the Module Name in a Dump

The MAINT macro supplies maintenance identification and the module name in a program dump.

Syntax



Parameters

Function Specifies the name of the module to be used as an identifier in a program dump.

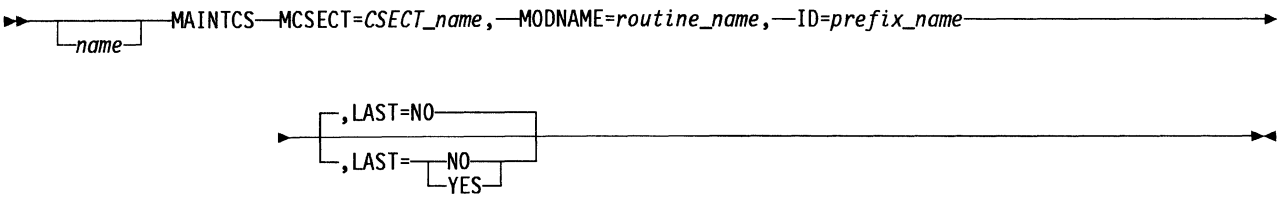
Format Label notation.

Default Control section (CSECT) name.

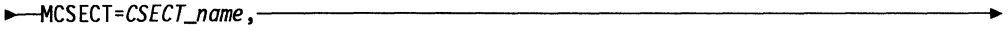
MAINTCS—Generate Additional CSECTs

The MAINTCS macro generates additional control sections (CSECTs) within a large module to allow the use of the branch instruction rather than the branch long instruction to branch to and from patch areas located at the end of CSECTs.

Syntax



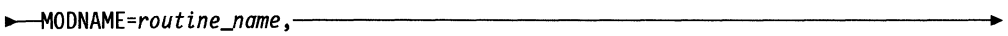
Parameters



Function Specifies the name of the main CSECT module. It will be used as an identifier in a program dump.

Format Label notation.

Default NONE.



Function Specifies the name of the module. It will be used as an identifier in a program dump.

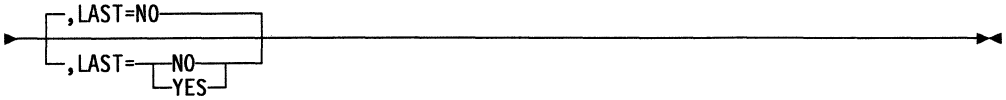
Format Label notation.

Default NONE.

Function Specifies a two-character name, which is used as a prefix to a 1-digit or 2-digit count that MAINTCS generates. MAINTCS uses the prefix and count to generate the new CSECT name. The new CSECT name is CX$*ppcc*, where *pp* is the *prefix_name* and *cc* is the count.

Format Literal two-character string.

Default NONE.



Function LAST=YES specifies that the last CSECT that was generated should be generated again. LAST=NO specifies that a new CSECT should be generated.

Format YES or NO

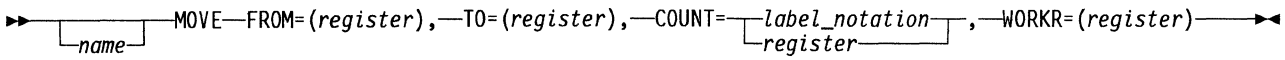
Default NO.

MOVE—Move Bytes from One Storage Location to Another (Inline)

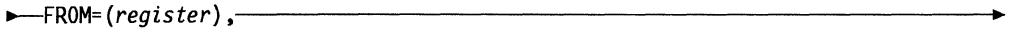
The MOVE macro transfers a specified number of bytes from one storage location to another. The function is executed inline.

Register 0 is not allowed for register parameters.

Syntax



Parameters



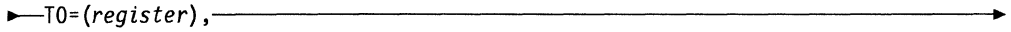
Function Specifies the register containing the starting address of the storage area from which the data is to be transferred.

Format Register notation.

Default None.

Remarks The register must be unique from those used on the other keywords.

At the end of the operation, this register contains the sum of the starting address and the count.



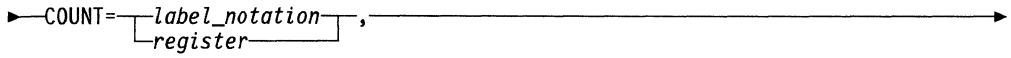
Function Specifies the register containing the starting address of the storage area into which the data is to be transferred.

Format Register notation.

Default None.

Remarks The register must be unique from those used on the other keywords.

At the end of the operation, this register contains the sum of the starting address and the count.



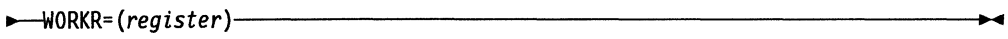
Function Specifies the number of bytes to be transferred.

Format Register or label (immediate) notation.

Default None.

Remarks If in register notation, the register must be unique from those used on the TO and FROM keywords. The register may overlap the WORKR register. The most efficient expansion results when the high order byte of WORKR is used, for example, COUNT=(register(0)).

If in label notation, the immediate count, *n*, must be a value from 0 to 255. The value 0 results in 256 bytes being transferred.



Function Specifies an odd-numbered halfword work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register must be unique from those used on the TO and FROM keywords. This register may overlap the COUNT register.

At the end of this operation, the high-order byte of this register contains the value 0, and the low-order byte contains the last character transferred.

MOVECHAR—Copy Bytes between Storage and Buffer Chains

The MOVECHAR macro enables you to copy bytes from contiguous storage into buffer chains, from buffer chains into contiguous storage, and from buffer chains into buffer chains. Buffers can be leased to build a buffer chain as data is moved into it.

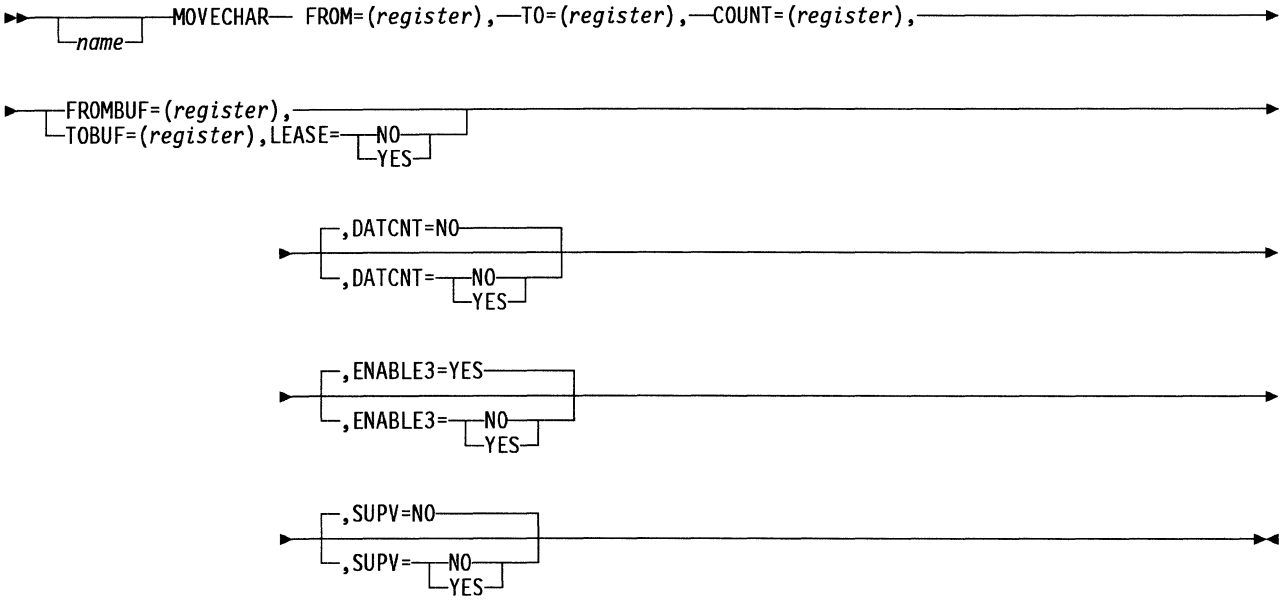
MOVECHAR makes no adjustment of buffer offset fields. It also does no checking for overlapping FROM and TO fields. Generally, if the fields overlap, the buffer counts will be wrong. The buffer data counts must be correct in all buffers before you invoke MOVECHAR. The MOVECHAR macro generates a PERFORM linkage to CXDSMOVE.

If you are using the MOVECHAR macro from level 5, you can use a 2048-byte work area at location CXTMV5. If you using it in a interrupt level, you can use a 64-byte work area at location CXTMV3.

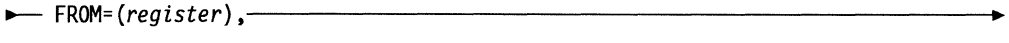
Registers 0 and 6 are not allowed. Registers used to pass keywords will be modified. You must provide a save area that can be overwritten before calling the MOVECHAR macro.

If you do not specify LEASE=YES or if the LEASE fails, only the number of bytes that will fit in the available buffers will be moved.

Syntax



Parameters

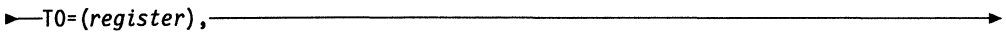


Function Specifies the register that contains the pointer to the start of the string of bytes to be copied.

Format Absolute register notation.

Default None.

Remarks The pointer to the byte following the last byte copied is returned in this register.



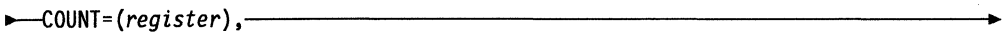
Function Specifies the register containing a pointer to the place to which data is to be copied.

Format Absolute register notation.

Default None.

Remarks The register specified must not be the same as that specified for FROM or FROMBUF.

The pointer to the byte following the last byte moved is returned in this register.



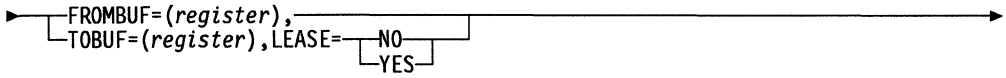
Function Specifies the register that contains a count of the number of bytes to be copied.

Format Absolute register notation.

Default None.

Remarks The register specified must not be the same as that specified for FROM, FROMBUF, TO, or TOBUF.

A value of 0 is returned in this register if all bytes are successfully copied. A count of the remaining bytes is returned if all bytes could not be moved.



Function Specifies the register that points to the buffer containing the start of the string of bytes to be copied, or the buffer into which data is to be moved.

LEASE specifies that buffers are to be leased if the string overruns buffer boundaries.

Format Absolute register notation; YES or NO.

Default None.

Remarks The register specified for FROMBUF must not be the same as that specified for FROM. The register specified for TOBUF must not be the same as that specified for FROMBUF or TO.

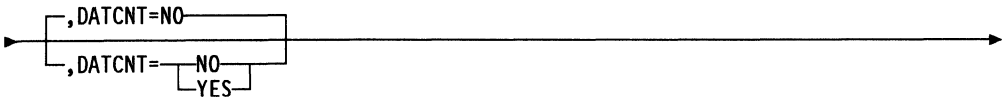
If you do not code the FROMBUF keyword, it indicates that the FROM string is in contiguous storage. If you do not code TOBUF, it indicates that the data string is to be moved to contiguous storage.

You must code either the FROMBUF or the TOBUF keyword.

The pointer to the buffer containing the end of the FROM string is returned in the FROMBUF register. The pointer to the buffer that contains the end of the data moved is returned in the TOBUF register.

LEASE is valid only if you code the TOBUF keyword; it must be coded if you code TOBUF.

If a LEASE fails before all data is copied, a residue count is returned.



Function Specifies whether the buffer data count is incremented. DATCNT=YES specifies that the buffer data count field is incremented for each byte moved into the buffer. DATCNT=NO specifies that the data count field is left unchanged.

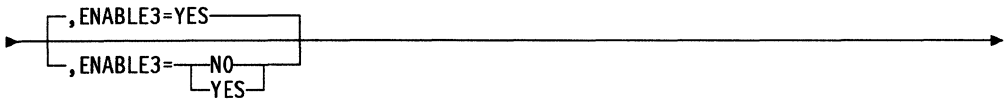
Format YES or NO.

Default NO.

Remarks If you code DATCNT=YES, you must code TOBUF.

New leased buffers have data count set to 0. If DATCNT=NO, no change is made. If DATCNT=YES, the data count is incremented.

If a buffer has a nonzero data count and DATCNT=YES, the number of bytes moved into that buffer is added to the existing count. If DATCNT=NO, the buffer data count is not changed.

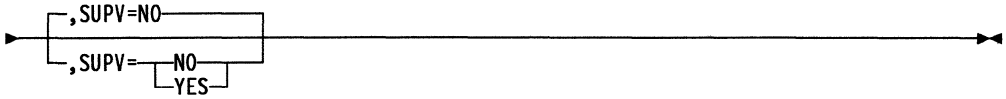


Function Specifies whether level 3 interrupts are enabled at the completion of the LEASE. ENABLE3=YES allows both level 2 and level 3 interrupts to be enabled at completion of the LEASE. ENABLE3=NO allows a level 4 routine to inhibit level 3 interrupts and issue a LEASE macro; only level 2 interrupts are enabled at the completion of the LEASE.

Format YES or NO.

Default YES.

Remarks This keyword is valid only when SUPV=YES.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in interrupt level. This information is needed by CXDSMOVE so that the proper type of supervisor macro can be issued.

Format YES or NO.

Default NO.

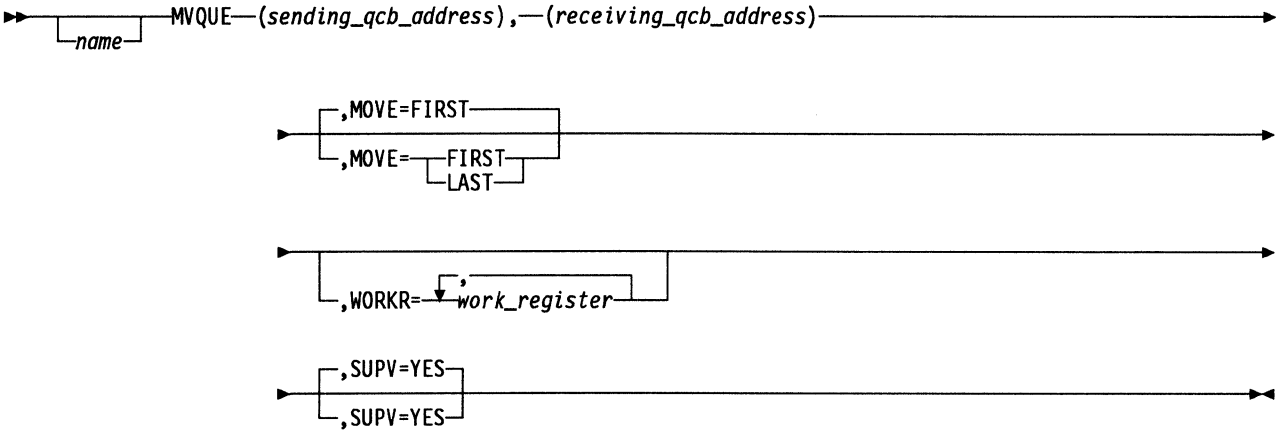
Remarks If you code SUPV=NO, an EXIT instruction is generated.

MVQUE—Move the Contents of One Queue to Another

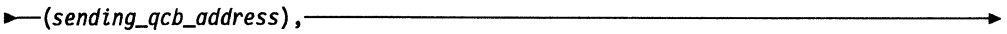
The MVQUE macro moves the entire contents of one queue to another queue. The moved queue contents can be moved ahead of or behind the receiving queue contents. SUPV=YES is the only valid usage of this macro. This function assumes that the first two adjacent fullwords of the sending and receiving queue control blocks (QCBs) are the head and tail pointers, respectively. The elements of the sending QCB must contain an event control block (ECB). Register 1 is destroyed if standard registers are used for the sending and receiving QCB addresses.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register that contains the address of the QCB, the complete queue contents of which are to be moved.

Format Register notation.

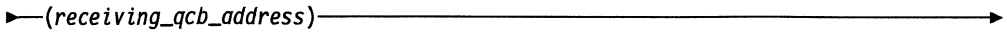
Default None.

Remarks Register 2 is standard.

The sending QCB address register cannot be the same as the receiving QCB address register. Register 1 is not allowed.

If the sending QCB is empty, no action is taken.

The sending QCB's head and tail pointers are each set to 0.



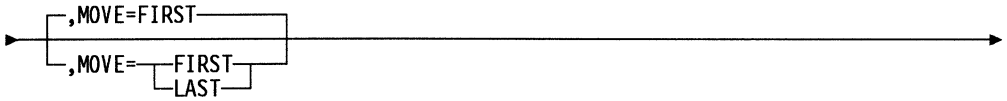
Function Specifies the register that contains the address of the QCB, the queue of which is to receive the sending queue contents.

Format Register notation.

Default None.

Remarks Register 3 is standard.

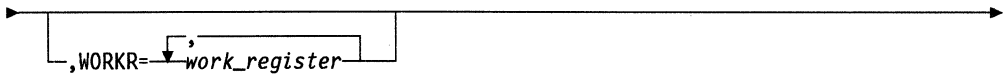
The receiving QCB address register cannot be the same as the sending QCB address register. Register 1 is not allowed.



Function Specifies the order in which queue contents are to be queued. FIRST specifies that the sending queue contents are to be queued ahead of the receiving queue contents; LAST specifies queueing behind the receiving queue contents.

Format FIRST OR LAST.

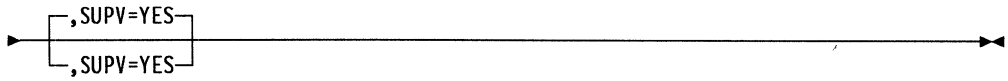
Default FIRST.



Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying such registers makes execution of the macro more efficient.

Format Registers 1, 2, 3, 4, 5, and 7 (up to a maximum of 6 registers) can be specified. Equated values cannot be used.

Default: No work registers.



Function Specifies that the issuer is running in an interrupt level.

Format YES.

Default YES.

NCHNG—Change Fields in a Programmed Resource Control Block

The NCHNG macro changes fields in programmed resource control blocks. It generates a parameter list and an SVC call to the NCHNG service routine. The macro uses register 1, so register 1 cannot be used to pass or receive information. If the information being passed requires less than a full register, the information must be right-justified in the register and the remainder of the register must be set to 0.

If the NCHNG requests are satisfied, register 1 will contain 0. If one or more of the requests cannot be satisfied, the low-order byte of register 1 will contain error bits identifying the errors:

Bit 0 (X'80') Specified index was not found in a function vector table (FVT).

Bit 1 (X'40') Request was not valid for the control block specified.

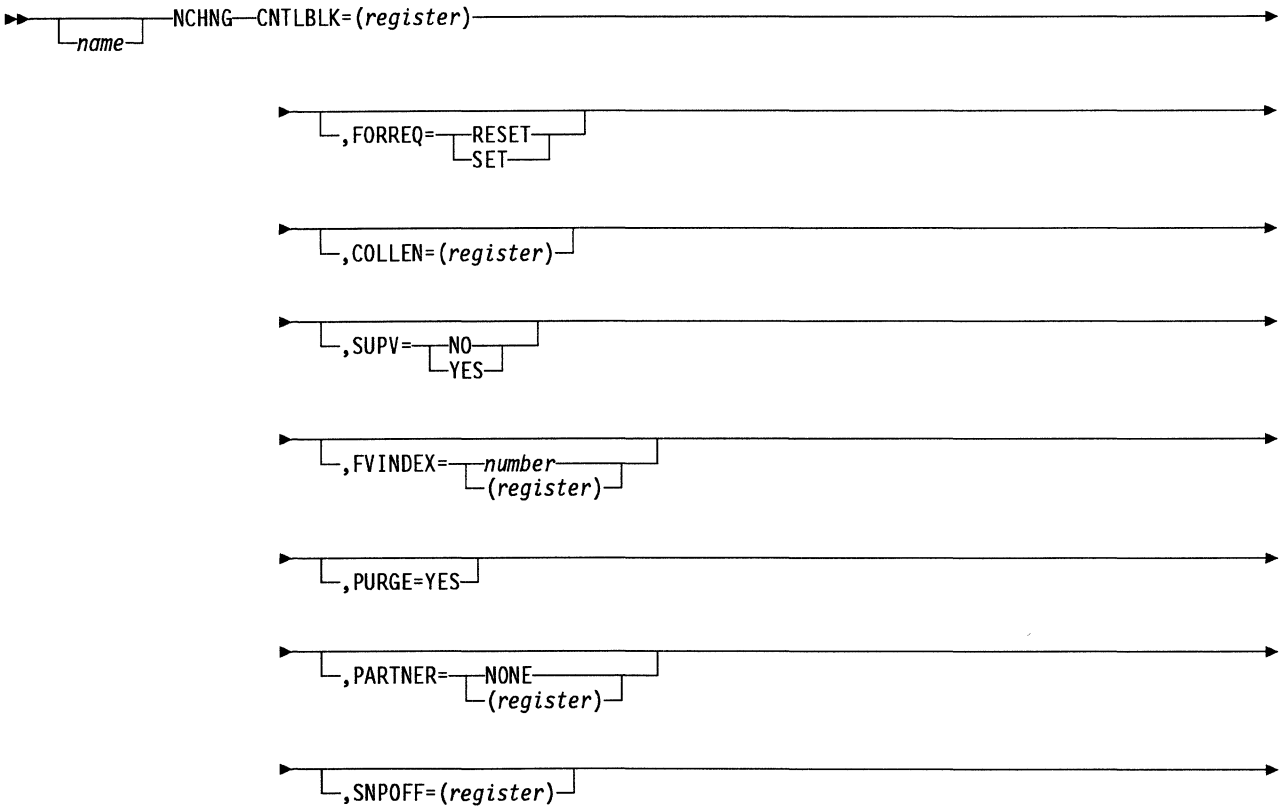
Bit 2 (X'20') New session partner specified was already in session or no virtual route vector table index (VVTI) was specified when using extended network addressing (ENA).

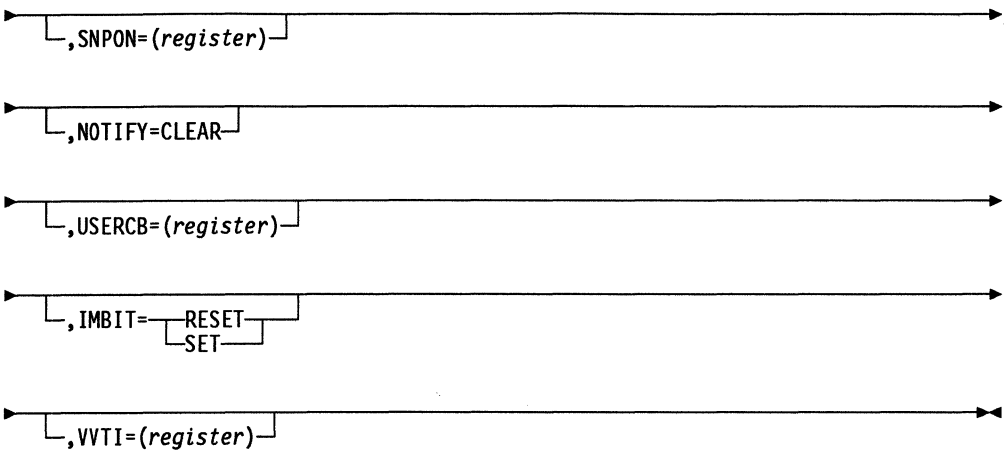
Bit 3 (X'10') Programmed resource logical unit block extension (NLX) specified was unavailable for a new session.

Register 0 is not allowed for register parameters.

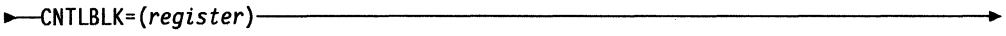
Note: You must code the CNTLBLK keyword and at least one other keyword.

Syntax





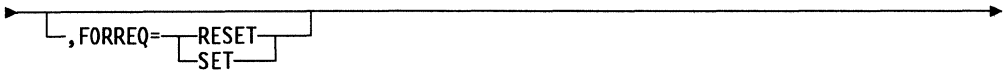
Parameters



Function Specifies the register containing the address of the programmed resource control block defining the resource being serviced.

Format Register notation.

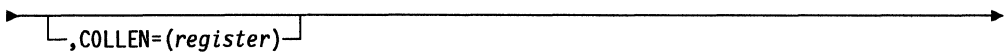
Default None.



Function Sets or resets the request for FORWARD notification during NetView Performance Monitor (NPM) processing. This keyword is valid only when the CTLBLK keyword specifies an NQE.

Format SET or RESET

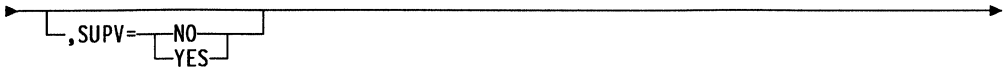
Default None.



Function Specifies the register that contains the length of the resource record that contains NPM performance data.

Format Register notation

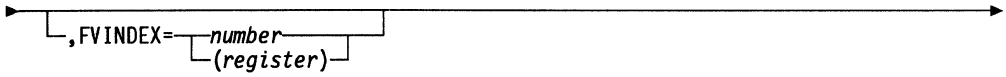
Default None.



Function Specifies the level in which the issuer is running. SUPV=NO indicates that it is invoked from level 5. SUPV=YES indicates that this macro is invoked in an interrupt level.

Format YES or NO

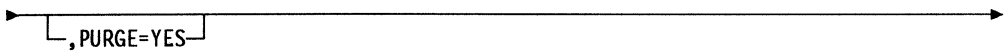
Default None.



Function Specifies an index to a new task pointer on the FVT for the resource. The task pointer is placed in the queue control block (QCB) for the resource, and NCP dispatches the new task the next time the QCB is activated. The FVTABLE values of both the previous and the new tasks are saved in the programmed resource control block.

Format Absolute or register notation.

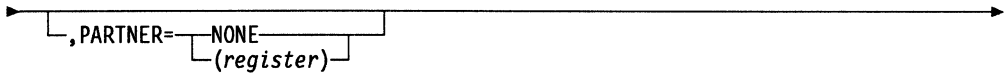
Default None.



Function Purges all elements currently queued to the control block.

Format YES.

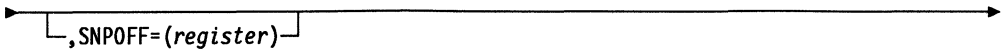
Default None.



Function Replaces the session-partner field in the given session control block with 0 if NONE is specified or with the element address in the specified register. This keyword is valid only when the CNTLBLK keyword specifies an NLX. Use NONE to clear an NLX at the end of a session and make it available for future use. Any path information unit (PIU) whose destination address field (DAF) matches the element address in the programmed resource logical unit block (NLB) and whose origin address field (OAF) matches the session partner's element address field (nonzero) in the NLX will be enqueued to the NLX rather than to the NLB.

Format NONE or register notation.

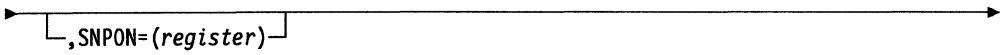
Default None.



Function Specifies the register whose low-order byte contains a session control block (SNP) mask that NCP uses to turn off a corresponding bit in the SNPMASK field of the specified programmed resource control block. This shows that the SSCP identified by the SNP mask is no longer an owner of the resource. This keyword is valid only if the CNTLBLK keyword specifies a programmed resource virtual line block (VLB) or a programmed resource physical unit block (NPB) and if the SNPON keyword is not coded.

Format Register notation.

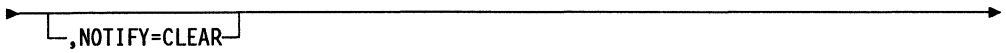
Default None.



Function Specifies the register whose low-order byte contains an SNP mask that NCP uses to turn on a corresponding bit in the SNPMASK field of the specified programmed resource control block. This shows that the SSCP identified by the SNP mask is now an owner of the resource. This keyword is valid only if the CNTLBLK keyword specifies a VLB or an NPB and if the SNPOFF keyword is not coded.

Format Register notation.

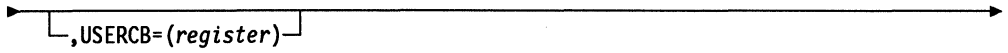
Default None.



Function Indicates that the user has completed handling the condition identified by the NOTIFY task information byte in the programmed resource control block and wants to set the field to 0.

Format CLEAR.

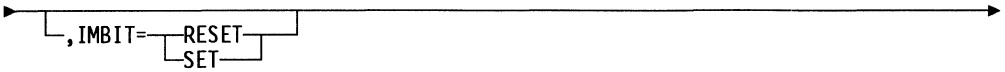
Default None.



Function Specifies the register which contains the address of the user control block that is to be inserted into the specified programmed resource control block.

Format Register notation.

Default None.

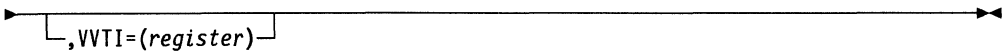


Function Sets or resets the immediate bit in the resource connection block (RCB). The immediate bit determines if the user's notify option, which is NCP system generated, is executed when the virtual route is held (virtual route blocked) or not held (virtual route unblocked). If the immediate bit is on, the user's notify option for this control block is executed. If the immediate bit is reset, the user's notify option is ignored.

Format SET or RESET.

Default None.

Remarks This keyword is valid only if the CNTLBLK keyword specifies an NPB, an NLB, or an NLX.



Function Specifies the register containing the VVTI to be stored in the specified programmed resource control block. This keyword is valid only when the CNTLBLK keyword specifies an NLX.

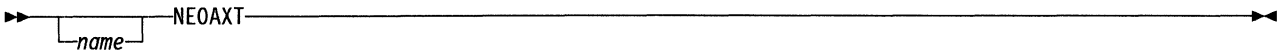
Format Register notation.

Default None.

NEOAXT—Generate an Accounting Routine Exit

The NEOAXT macro generates the proper return linkage to NCP from IBM special products or user-written code accounting routines. When control is returned to NCP, register 6 must contain the same address that was passed to the IBM special products or user-written code accounting routine. Refer to Table 14 on page 444 for attachment details for a user accounting notify routine.

Syntax



NEOENQ—Pass a PIU to NCP Physical Services

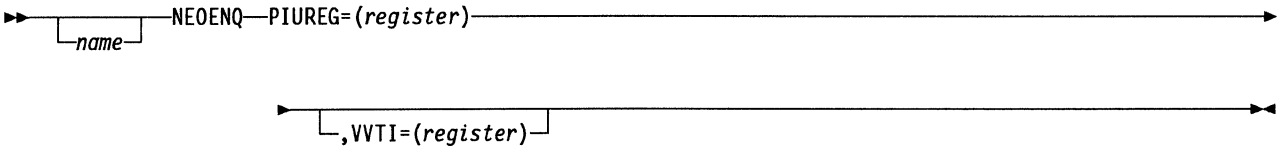
The NEOENQ macro passes to the NCP physical services any path information unit (PIU), such as INOP, that must flow in the NCP physical services-SSCP session. The PIU is then forwarded using the destination address field (DAF) already in the PIU, but with the origin address field (OAF) and sequence control of NCP physical services. The macro can be used only in program level 5.

The PIU DSECT (XCXTPIU) must be included in the module invoking the NEOENQ macro.

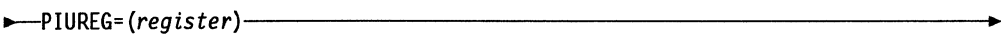
You must supply the virtual route vector table index (VVTI), either by coding the VVTI keyword or by including the VVTI in the PIU to be passed.

Register 0 is not allowed for register parameters.

Syntax



Parameters



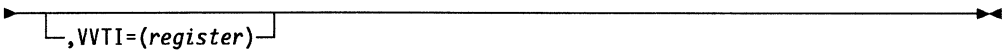
Function Specifies the register containing the address of the PIU to be forwarded.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

The register specified for PIUREG cannot be the same as that specified for VVTI.



Function Specifies the register containing the VVTI to be put in the PIU.

Format Register notation.

Default None.

Remarks The register specified for the VVTI keyword cannot be the same as that specified for PIUREG.

You must code the VVTI keyword if the VVTI is not already included in the PIU being passed.

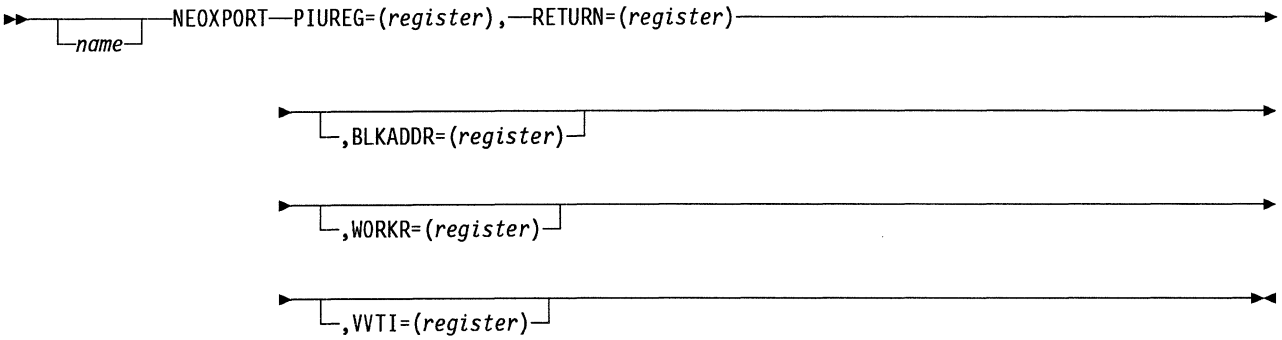
If no register is specified, 0 is put in the PIU VVTI field.

NEOXPORT—Route PIUs to a Network Address

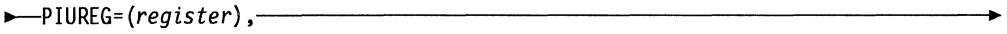
The NEOXPORT macro routes path information units (PIUs) to the network address specified in the destination address field (DAF) of the PIU in user-written code instead of the XPORTVR macro. The NEOXPORT macro can be used only in program level 5.

Register 0 is not allowed for register parameters.

Syntax



Parameters

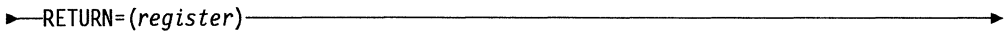


Function Specifies the register containing the PIU to be sent.

Format Register notation.

Default None.

Remarks This register may be the same as that specified for the RETURN register, but must be different from that coded for the BLKADDR or virtual route vector table index (VVTI) register. The Z latch is set to 1 if the PIU was routed. Register 1 is not allowed.

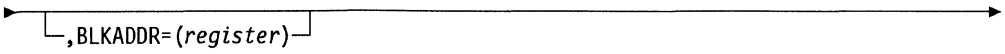


Function Specifies the register to contain the return code for the operation. See the XPORTVR macro description for the definition of these return codes.

Format Register notation.

Default None.

Remarks This register may be the same as that specified for any other keyword.



Function Specifies the address of the control block associated with the virtual route.

Format Register notation.

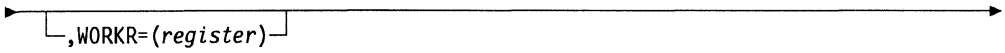
Default None.

Remarks This register cannot be the same as the PIU register.

You must code either this keyword or the VVTI keyword.

When this keyword is coded, the WORKR keyword is required.

If you code neither this keyword nor the VVTI keyword, the VVTI is extracted from the PIU.



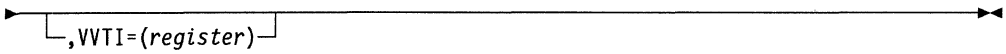
Function Specifies a work register, the contents of which may be altered during execution of the macro. You must code WORKR when you code the BLKADDR keyword.

Format Register notation.

Default None.

Remarks This register cannot be the same as the PIU register.

Only register 3, 5, or 7 can be used.



Function Specifies the register that contains the VVTI. If you code this keyword, it overrides the VVTI contained in a control block.

Format Register notation.

Default None.

Remarks This register cannot be the same as the PIU register.

You must code either this keyword or the BLKADDR keyword.

Register 1 is not allowed.

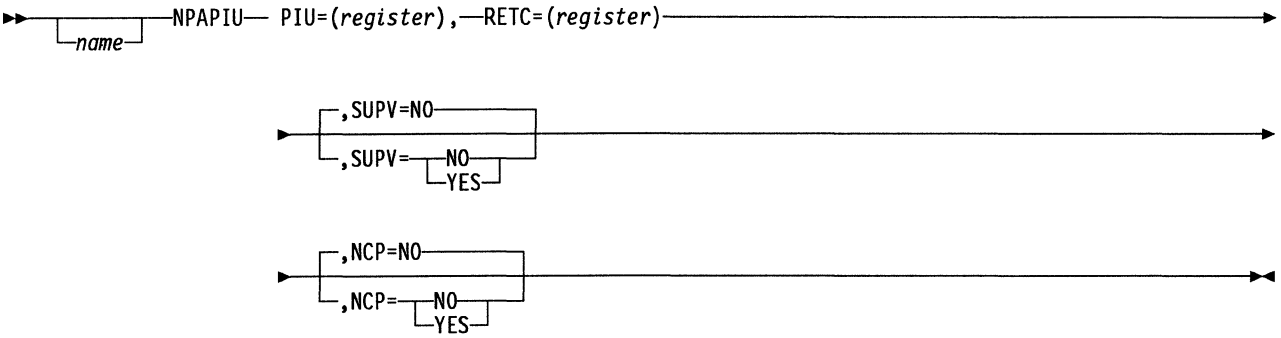
If neither this keyword nor the BLKADDR keyword is coded, the VVTI is extracted from the PIU.

NPAPIU—Send an NPA PIU to NetView Performance Monitor

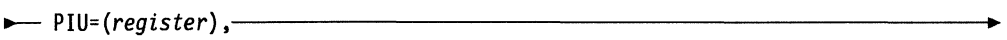
The NPAPIU macro sends network performance analyzer path information units (PIUs), accounting data request/response units (RUs), session start RUs, session end RUs, and session start/end RUs to NetView Performance Monitor (NPM).

Register 6 must be a valid save area pointer. Register 0 is not allowed.

Syntax



Parameters



Function Specifies the register containing the address of the PIU to be sent.

Format Register notation.

Default None.

Remarks This register should not be the same as that specified for the RETC keyword. Register 6 is not valid.

Function Specifies the register to receive the return code. The possible values are:

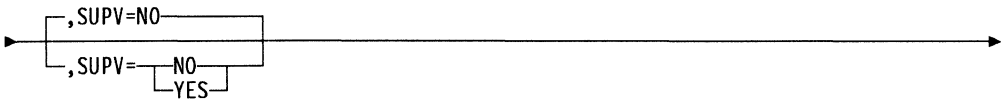
- X'0000' PIU enqueued (PIU register will be 0 on output).
- X'0001' PIU in error. The PIU should be a FID1, FM data PIU with the data counts correct.
- X'0002' PIU not sent (accounting data send mode for IBM special products or user-written code is disabled).
- X'0003' Boundary session accounting function not supported by NCP.
- X'0004' PIU enqueued (however, accounting data send mode for IBM special products or user-written code is disabled because the virtual route is held).

Format Register notation.

Default None.

Remarks This register should not be the same as that used by the PIU keyword. Register 6 is invalid.

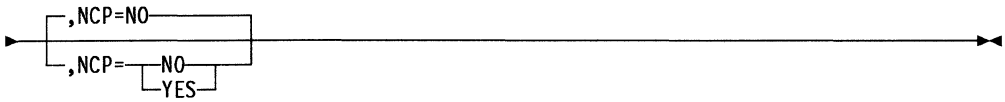
For a RETC of X'0001', X'0002', and X'0003', the PIU is not enqueued; it is returned to the issuer of the macro.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.



Function Specifies the type of code generated on the basis of the issuer's storage protection key. NCP=YES indicates that the macros are to be expanded inline. NCP=NO indicates that an SVC call is to be used.

Format YES or NO.

Default NO.

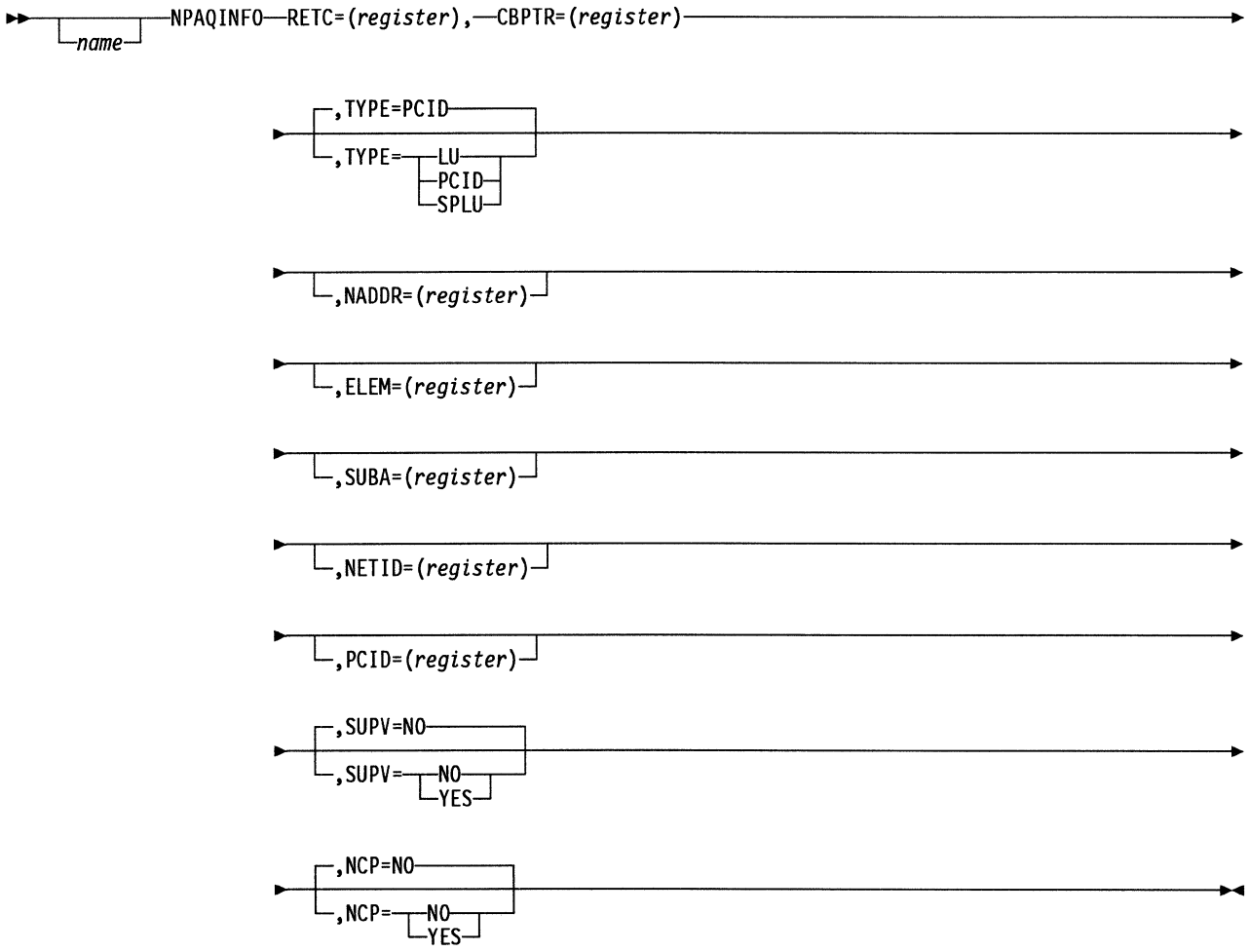
Remarks Specify this keyword only when SUPV=NO.

NPAQINFO—Retrieve SNA Session Information for User Line Control

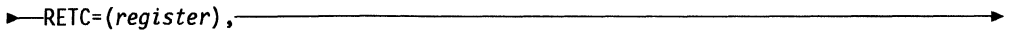
The NPAQINFO macro retrieves Systems Network Architecture (SNA) LU-LU session information for user line-control resources only. This macro can retrieve information about the logical unit, the session partner logical unit, or the procedure-correlation identifier (PCID). If information about more than one of these is to be retrieved, you must code NPAQINFO more than once.

Register 6 must be a valid save area pointer. Register 0 is not allowed.

Syntax



Parameters



Function Specifies the register to get the return code after execution of the NPAQINFO macro. The possible values are:

X'0000' Successful; the requested information was returned.

X'0001' The requested information was returned; the logical unit is not accounted for by the boundary session accounting function.

X'0002' The requested information was not returned; NCP does not include the boundary session accounting function.

X'0003' The requested information was not returned; information is not available (TYPE=PCID only).

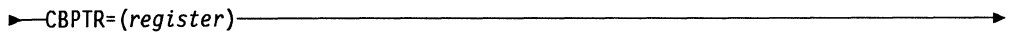
Format Register notation.

Default None.

Remarks Must be an odd-numbered register.

The register you specify must not be the same as that specified for any other keyword.

Register 6 is invalid.



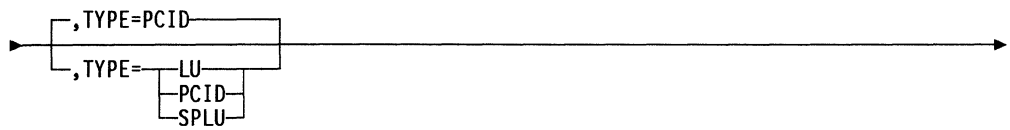
Function Specifies the register containing a pointer to the LU session control block (BSB).

Format Register notation.

Default None.

Remarks The register you specify must not be the same as that specified for any other keyword.

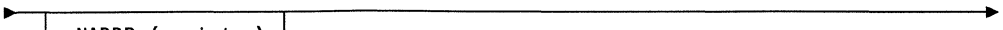
Register 6 is invalid.



Function Specifies what type of information is to be retrieved. This can be either PCID, logical unit (LU), or session partner logical unit (SPLU).

Format PCID, LU, or SPLU.

Default PCID.



Function Specifies the register containing a pointer to the location that is to contain the name portion of the network-qualified network name (TYPE=LU or SPLU) or control point name (TYPE=PCID).

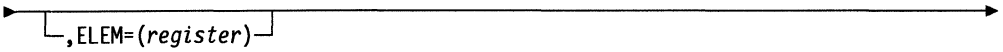
Format Register notation.

Default None.

Remarks The name stored will be 8 bytes long, left-justified, and padded with blank spaces.

The register you specify must not be the same as that specified for any other keyword.

Register 6 is invalid.



Function Specifies the register containing the element address (LU or SPLU).

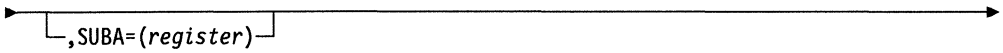
Format Register notation.

Default None.

Remarks This parameter is valid only if TYPE=LU or TYPE=SPLU.

The register you specify must not be the same as that specified for any other keyword.

Register 6 is invalid.



Function Specifies the register containing a pointer to the location that is to contain the subarea address (LU or SPLU).

Format Register notation.

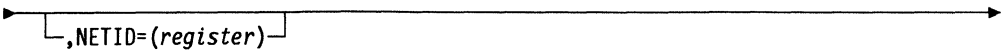
Default None.

Remarks This parameter is valid only if TYPE=LU or TYPE=SPLU.

The subarea address stored will be 4 bytes long.

The register you specify must not be the same as that specified for any other keyword.

Register 6 is invalid.



Function Specifies the register containing a pointer to the location that is to contain the NETID portion of the network-qualified network name (TYPE=LU or TYPE=SPLU) or control point name (TYPE=PCID).

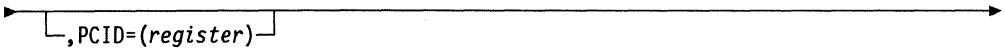
Format Register notation.

Default None.

Remarks The NETID stored will be 8 bytes long, left-justified, and padded with blank spaces.

The register you specify must not be the same as that specified for any other keyword.

Register 6 is invalid.



Function Specifies the register that contains a pointer to the location that is to contain the fully qualified PCID.

Format Register notation.

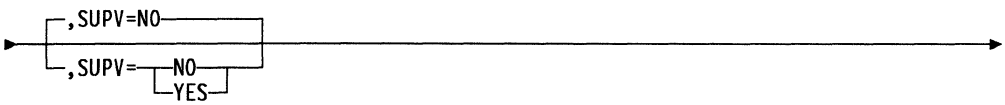
Default None.

Remarks This parameter is valid only if TYPE=PCID.

The PCID stored will be 8 bytes long.

The register you specify must not be the same as that specified for any other keyword.

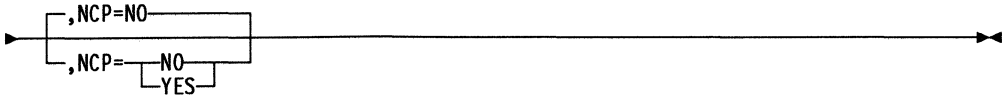
Register 6 is invalid.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.



Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES indicates that the macros are to be expanded inline. NCP=NO indicates that an SVC call is to be used.

Format YES or NO.

Default NO.

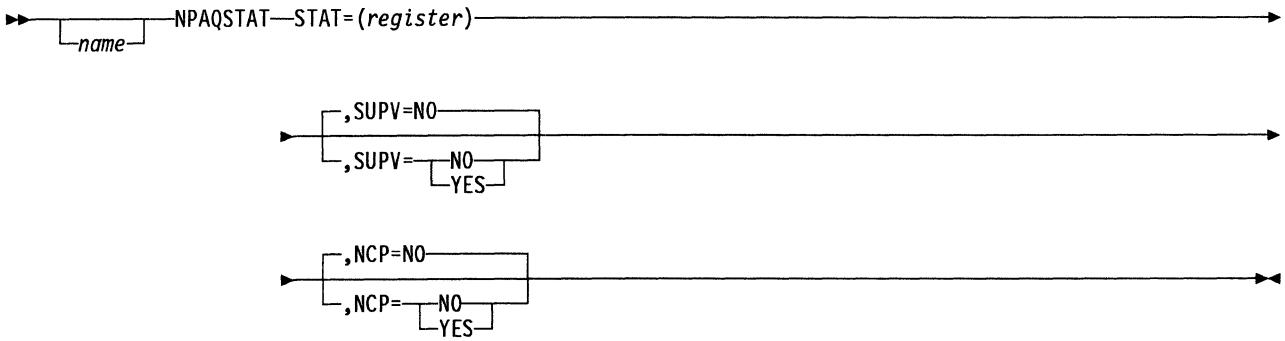
Remarks Specify this parameter only when SUPV=NO.

NPAQSTAT—Retrieve the Boundary Session Accounting Status

The NPAQSTAT macro retrieves the status of the boundary session accounting function.

Register 6 must be a valid save area pointer. Register 0 is not allowed.

Syntax



Parameters

Function Specifies the register to get the status of the boundary session accounting. The possible values are:

X'0000' Accounting collection is included and the accounting data send mode for IBM special products or user-written code is enabled.

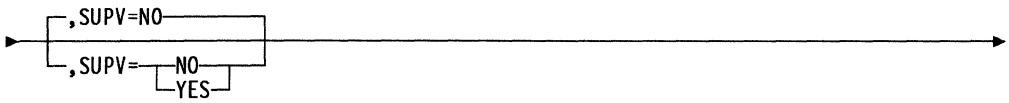
X'0001' Accounting collection is included and the accounting data send mode for IBM special products or user-written code is disabled.

X'0002' Accounting collection is not included.

Format Register notation.

Default None.

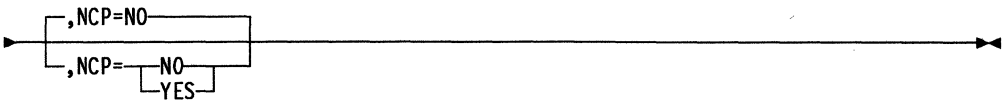
Remarks Register 6 is invalid.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.



Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES indicates that the macros are to be expanded inline. NCP=NO indicates that an SVC call is to be used.

Format YES or NO.

Default NO.

Remarks Specify this parameter only when SUPV=NO.

NPARMS—Get Information from a Programmed Resource Control Block

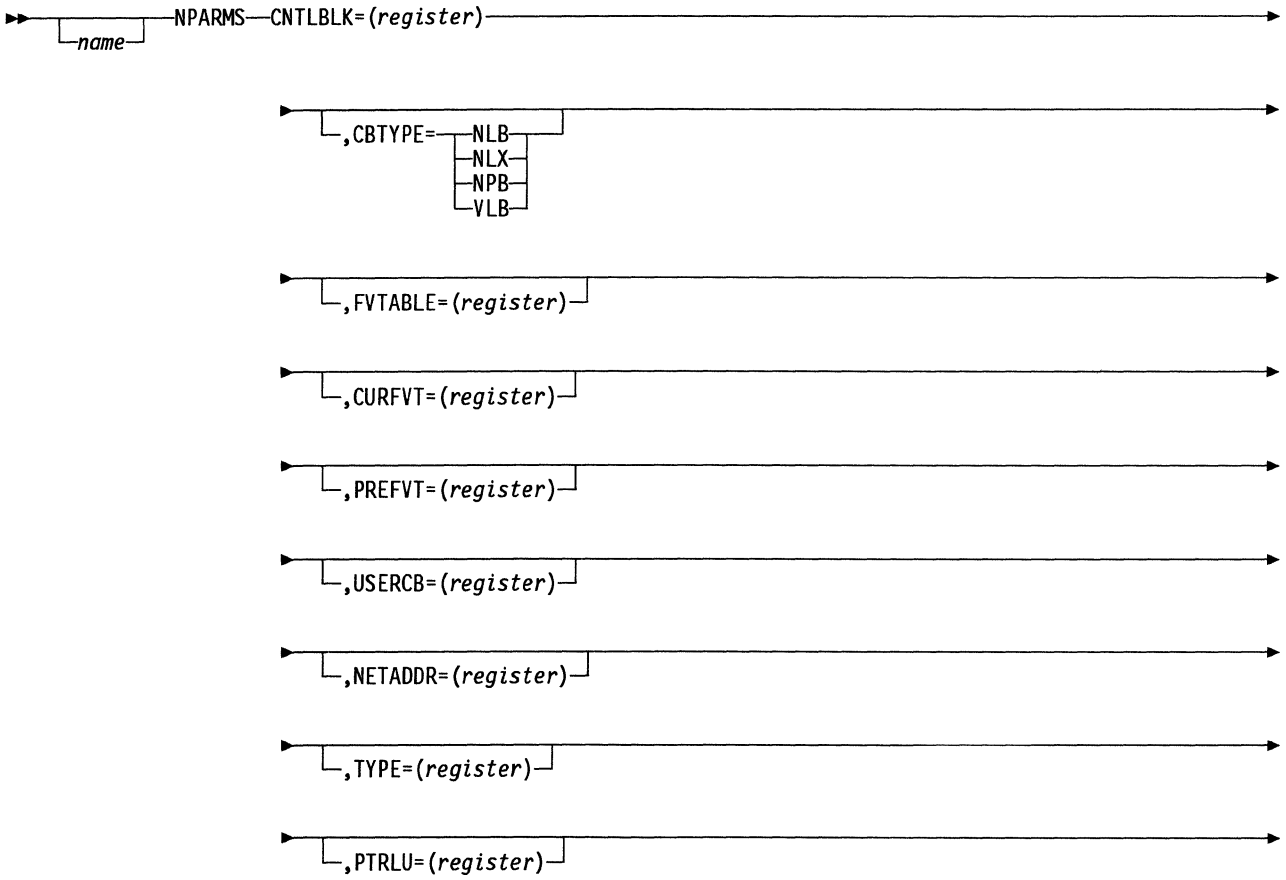
The NPARMS macro tasks or subroutines get information from the programmed resource control blocks. Because this macro might use registers 1 and 6 as work registers, these registers might not be preserved when the macro is called. The descriptions of the keywords indicate which registers they use and if error codes are returned. Register 0 is not allowed for register parameters.

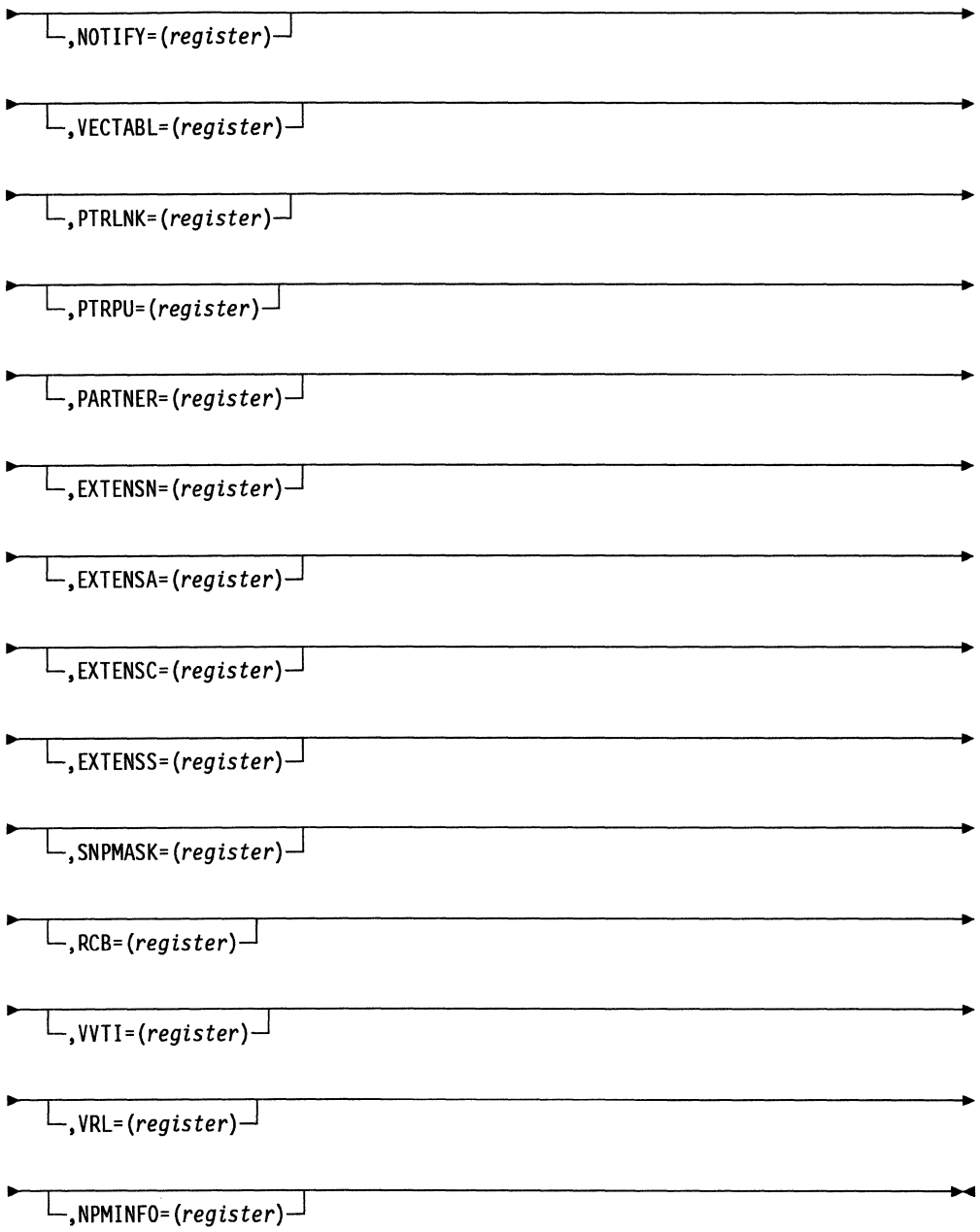
The NPARMS macro without the CBTYPE keyword can be used only in a routine with storage protection key 1. When CBTYPE is specified, the NPARMS macro can be used in a routine executing with any storage protection key or in a routine that executes before storage keys are set.

If the information requested is less than a full register, the information will be returned right-justified in the specified register with the remainder of the register set to 0. If CBTYPE is not specified and the control block whose address is supplied is not a valid programmed resource control block, or if a register that is not allowed is specified, the results are unpredictable and you must assume that none of your requests were satisfied.

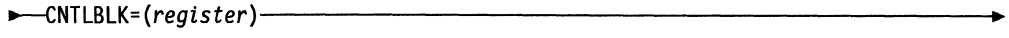
Note: You must code the CNTLBLK keyword and at least one, but not more than five, of the other keywords.

Syntax





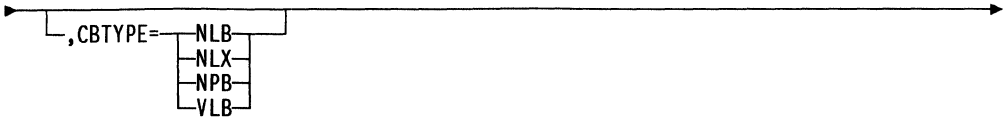
Parameters



Function Specifies the register that contains the address of the control block.

Format Register notation.

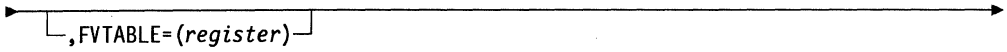
Default None.



Function This keyword has two functions: it specifies the type of programmed resource control block whose address is in the register specified by the CNTLBLK keyword, and it causes the generation of inline code instead of establishing the CALL linkage to the NPARMS service routine (CXDNPRM). All other keywords remain unchanged.

Format NLB, NLX, NPB, or VLB.

Default None.



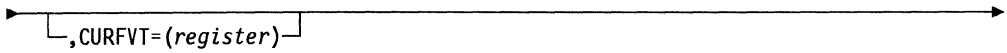
Function Specifies the register to contain the address of the function vector table (FVT) associated with the control block.

Format Register notation.

Default None.

Errors None.

Work Register None.



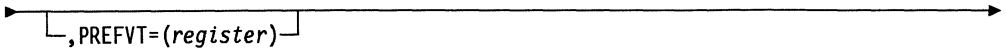
Function Specifies the register to contain the address of the current task function vector index (byte).

Format Register notation.

Default None.

Errors None.

Work Register Register 1.



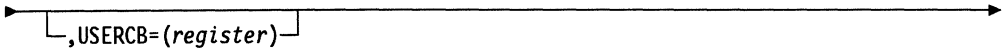
Function Specifies the register to contain the address of the previous task function vector index (byte).

Format Register notation.

Default None.

Errors None.

Work Register Register 1.



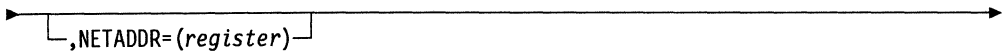
Function Specifies the register to contain the address of the user control block associated with the specified control block.

Format Register notation.

Default None.

Errors None.

Work Register None.



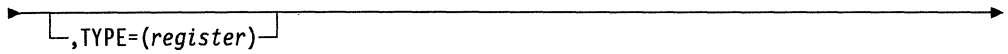
Function Specifies the register to contain the element address (halfword) of the resource associated with the control block.

Format Register notation.

Default None.

Errors None.

Work Register Register 6.



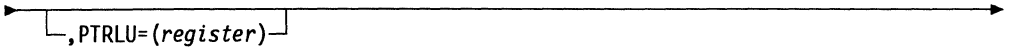
Function Specifies the register to contain the resource type (byte). The type bytes are VLB, X'01'; NPB, X'02'; NLB, X'04'; NLX, X'08'.

Format Register notation.

Default None.

Errors None.

Work Register Register 1.



Function Specifies the register to contain the address of the programmed resource logical unit block (NLB) associated with the control block.

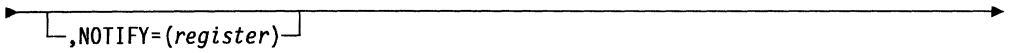
Format Register notation.

Default None.

Errors None.

Work Register None.

Remarks The control block specified by the CNTLBLK keyword must be a programmed resource logical unit block extension (NLX) for this keyword to be valid.



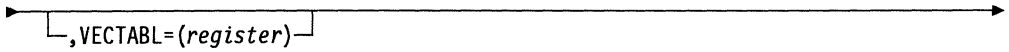
Function Specifies the register to contain the "notify" task information field (byte) of the FVT.

Format Register notation.

Default None.

Errors None.

Work Register Register 1.



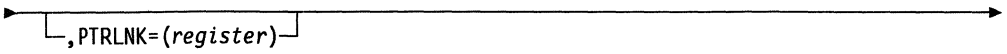
Function Specifies the register to contain the address of the vector table containing the addresses of all the resources owned by this resource. If you code VECTABL, CNTLBLK must specify a virtual link block (VLB) or a programmed resource physical unit block (NPB). For example, if the address supplied via CNTLBLK is associated with a link (VLB), the macro will return the address of the physical unit vector table for that link.

Format Register notation.

Default None.

Errors An MNOTE is issued and the VECTABL register contains 0.

Work Register None.



Function Specifies the register to contain the address of the VLB associated with the control block specified by the CNTLBLK keyword.

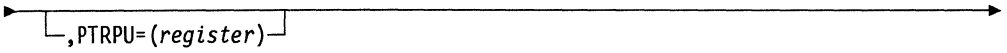
Format Register notation.

Default None.

Errors An MNOTE is issued and the PTRLNK register contains 0.

Work Register None.

Remarks For this keyword to be valid, the control block specified by the CNTLBLK keyword must be an NPB, an NLB, or an NLX. This keyword is not valid if CNTLBLK specifies an NLB that is an NCPNAU, because the NLB would have no associated VLB.



Function Specifies the register to contain the address of the NPB associated with the control block specified by the CNTLBLK keyword.

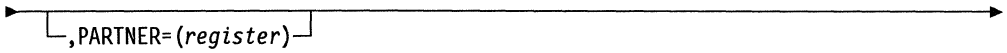
Format Register notation.

Default None.

Errors An MNOTE is issued and the PTRPU register contains 0.

Work Register None.

Remarks The control block specified by the CNTLBLK keyword must be an NLB or an NLX for this keyword to be valid. This keyword is not valid if CNTLBLK specifies an NLB that is an NCPNAU, because the NLB would have no associated NPB.



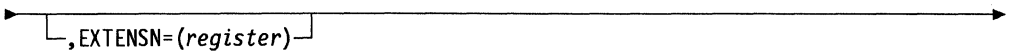
Function Specifies the register to contain the element address (halfword) of the session partner. The keyword is valid only if the control block specified by the CNTLBLK keyword is an NLX.

Format Register notation.

Default None.

Errors An MNOTE is issued and the PARTNER register contains 0.

Work Register None.



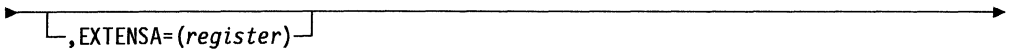
Function Specifies the register to contain the number (1 to 8) of this NLX.

Format Register notation.

Default None.

Errors An MNOTE is issued and the EXTENSN register is set to 0.

Work Register None.



Function Specifies the register to contain the address of the next available NLX for the logical unit.

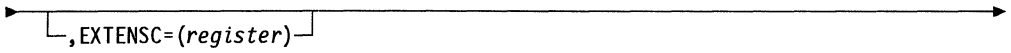
Format Register notation.

Default None.

Errors An MNOTE is issued and the EXTENSA register contains 0.

Work Register Register 1.

Remarks This keyword is valid only if the control block specified by the CNTLBLK keyword is an NLX or an NLB.



Function Specifies the register to contain the address of the NLX whose number is supplied in the same specified register.

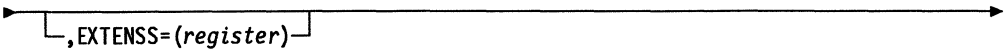
Format Register notation.

Default None.

Errors An MNOTE is issued and the EXTENSC register contains 0.

Work Register Registers 1 and 6.

Remarks This keyword is valid only if the control block specified by the CNTLBLK keyword is an NLB or an NLX.



Function Specifies the register to contain the address of the NLX whose element address is supplied in the same specified register.

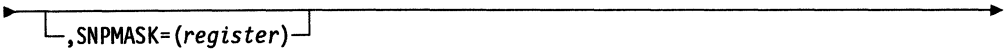
Format Register notation.

Default None.

Errors An MNOTE is issued and the EXTENSS register contains 0.

Work Registers Registers 1 and 6.

Remarks This keyword is valid only if the control block specified by the CNTLBLK keyword is an NLB or an NLX.



Function Specifies the register to contain the SNPMASK field (byte). The SNPMASK field identifies the owning SSCP.

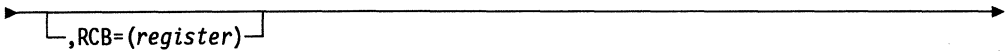
Format Register notation.

Default None.

Errors An MNOTE is issued and the SNPMASK register contains 0.

Work Register Register 1.

Remarks This keyword is not valid if the control block specified by the CNTLBLK keyword is an NLB or an NLX.



Function Specifies the register to contain the address of the resource connection block.

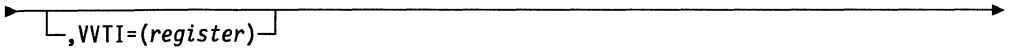
Format Register notation.

Default None.

Errors An MNOTE is issued and the RCB register contains 0.

Work Register Register 1.

Remarks This keyword is valid only if the CNTLBLK keyword specifies an NPB, an NLB, or an NLX.



Function Specifies the register to contain the virtual route vector table index (VVTI).

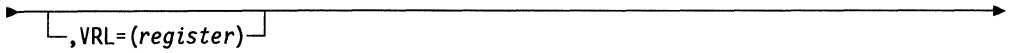
Format Register notation.

Default None.

Errors An MNOTE is issued and the VVTI register contains 0.

Work Registers Registers 1 and 6.

Remarks This keyword is valid only if the CNTLBLK keyword specifies an NPB, an NLB, or an NLX.



Function Specifies the register to contain the address of the virtual route activation work list.

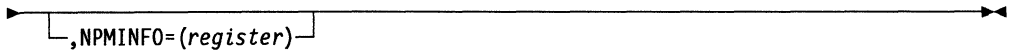
Format Register notation.

Default None.

Errors An MNOTE is issued and the VRL register contains 0.

Work Register Register 1.

Remarks This keyword is valid only if the CNTBLK keyword specifies an NPB, an NLB, or an NLX.



Function Specifies the register to contain the NetView Performance Monitor (NPM) information byte that contains the NPM requests. The NPM requests are:

X'01'	Start collection for resource.
X'02'	Forward NPM data.
X'03'	Stop collection for resource.

Format Register notation.

Default None.

Errors An MNOTE is issued and the NPMINFO register contains 0.

Work Register Register 1.

Remarks This keyword is valid only if the control block specified by the CNTLBLK keyword specifies a VLB or an NPB.

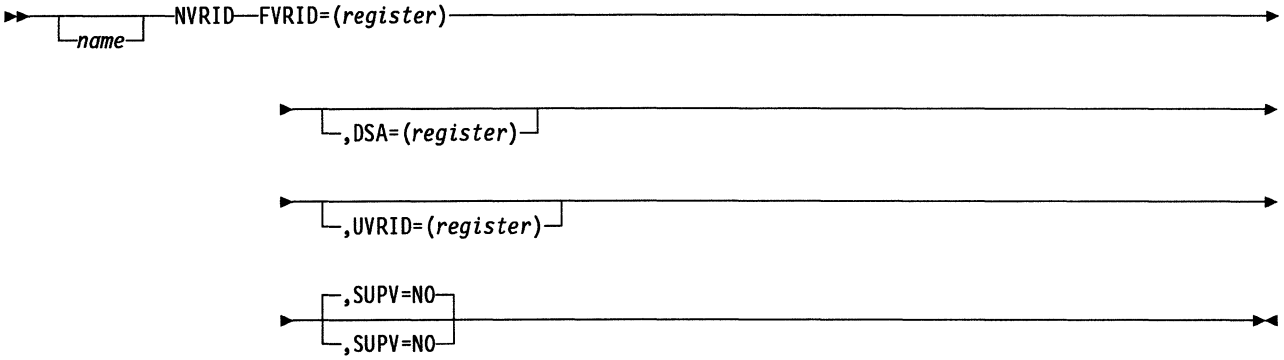
NVRID—Format the Virtual Route Identifier List

The NVRID macro generates an SVC service routine. The function of the macro and its service routine is to format your virtual route identifier (VRID) list. When your VRID list is formatted, the NVRID macro's service routine merges your VRID list into NCP's internal VRID list. When it is known, the destination subarea address (DSA) is stored in the formatted VRID list. See the VRACT and VRACTCK macros in this chapter.

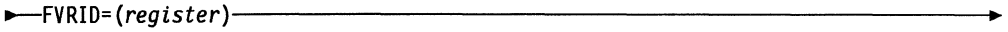
Register 0 is not allowed for register parameters.

Note: Assume that the address passed to the NVRID macro for the formatted VRID list points to enough space for the VRID list.

Syntax



Parameters



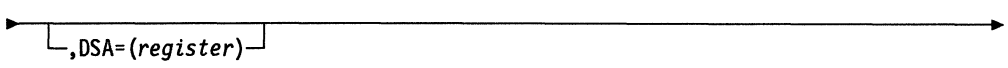
Function Specifies the register that contains the VRID list pointer. If you specify UVRID, this register contains a pointer to the location that is to contain the formatted VRID list. If you specify DSA, this register contains a pointer to a previously formatted VRID list.

Format Absolute register notation.

Default None.

Remarks Neither register 1 nor register 6 is allowed.

The register you specify must not be the same as that specified for any other keyword.



Function Specifies the register that contains the DSA.

Format Register notation.

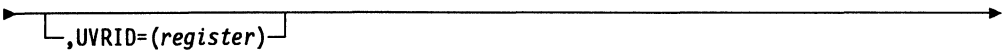
Default None.

Remarks Neither register 1 nor register 6 is allowed.

The register you specify must not be the same as that specified for any other keyword.

If this keyword is specified, assume that the FVRID keyword points to a previously formatted VRID list.

The DSA is the address of a subarea within NCP network.



Function Specifies the register that contains the pointer to the user's VRID list.

Format Absolute register notation.

Default None.

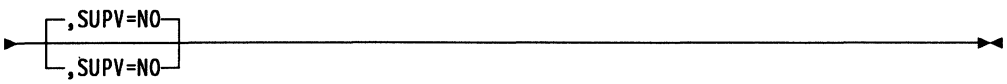
Remarks Neither register 1 nor register 6 is allowed.

The register you specify must not be the same as that specified for any other keyword.

Your VRID list is expected to be in contiguous storage.

Your VRID list is expected to be in the format of the virtual route list control vector (X'1B') or control vector X'0D'.

You can omit this keyword if the FVRID keyword has a pointer to a previously formatted VRID list. If you omit this keyword, you must specify the DSA keyword.



Function Specifies that the issuer is running in level 5.

Format NO.

Default NO.

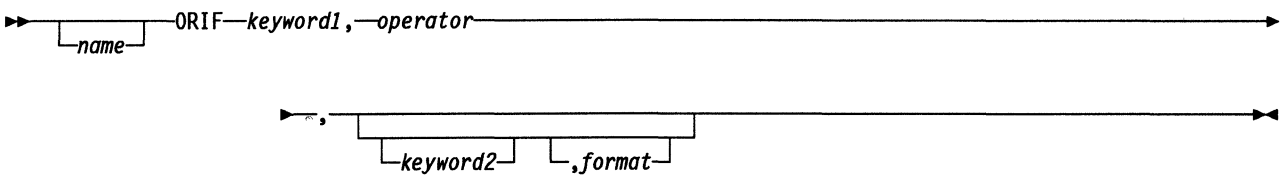
ORIF—Provide a Logical OR for an IF Macro

The ORIF macro is used with the IF macro to provide a logical OR decision capability in structured programming. It may follow an IF macro, an ANDIF macro, or another ORIF macro.

Register 0 is not allowed for register parameters.

Note: For efficiency, clarity, and better structure, you should use the IF macro with the AND or OR keyword, rather than using the ORIF macro.

Syntax



For a description of the keywords, see the IF macro description.

Example

```
IF   RGTEMP,EQ,X'80',I
ORIF RGSTAT,EQ,X'40',I
THEN ----
ELSE ----
```

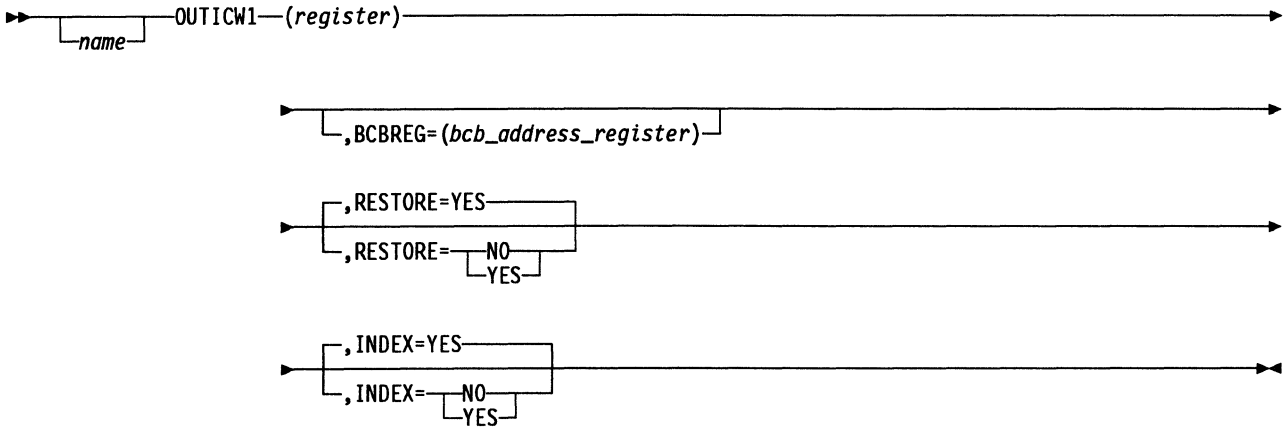
If the value in RGTEMP is X'80' or the value in RGSTAT is X'40', the instructions that are prefixed by the THEN macro are executed. If neither condition is met, the instructions that are prefixed by the ELSE macro are executed.

OUTICW1—Transfer the Contents of a Register to the ICW

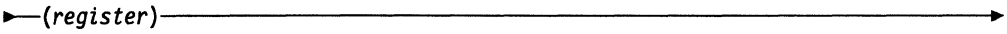
The OUTICW1 macro transfers the contents of a specified register to bits 0 to 15 (SCF/PDF) of the initial control word (ICW). This macro is used in program levels 1, 2, 3, and 4, and requires that the adapter control block (ACB), boundary session block (BCB), or ICW DSECT be present and addressability be established.

Register 0 is not allowed for register parameters.

Syntax



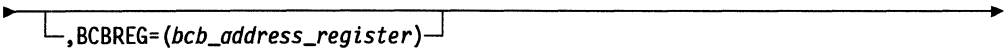
Parameters



Function Specifies the register that contains the new SCF/PDF to be transferred to the ICW.

Format Register notation.

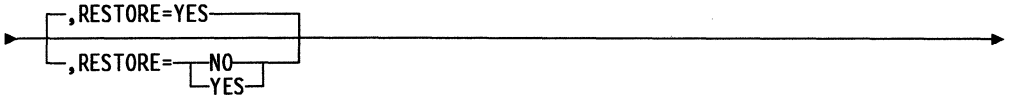
Default None.



Function Specifies the register that contains the address of the BCB in which the contents of the out register are to be stored.

Format Register notation.

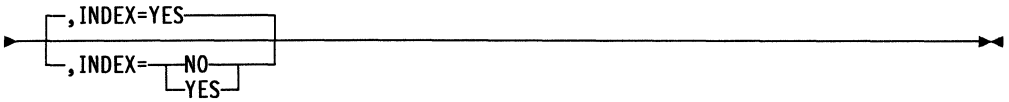
Default None.



Function Specifies whether to reestablish the ACB address register.

Format YES or NO.

Default YES.



Function Specifies whether initialization of the BCB address is required.

Format YES or NO.

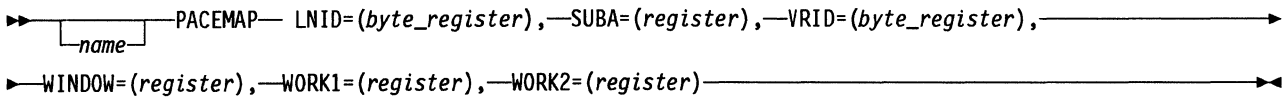
Default YES.

PACEMAP—Determine Virtual Route Pacing Window Sizes

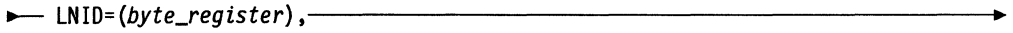
The PACEMAP macro produces inline code that uses local network identification (LNID), a subarea address, and the virtual route identifier (VRID) to determine the virtual route pacing window maximum and minimum sizes for the input virtual route if these sizes have been specified during NCP generation. If no window sizes have been specified, the macro returns a 0. The invoking routine must include the network vector table (NVT) and flow control parameter table (FCT) DSECTs (XCXTNVT and XCXTFCT) and must have an EXTRN for CXTNVT0, because PACEMAP issues the RSLVNET macro to get a pointer to the scanner interface trace (SIT) for the appropriate network. The invoking routine must also include an EXTRN for CXTNVT so that PACEMAP can access the NVT header and extension to get the FCT pointer.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register containing the LNID.

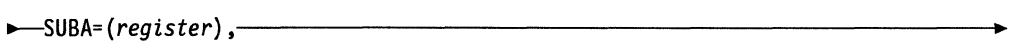
Format Register notation.

Default None.

Remarks The register must be a byte register.

The register you specify must not be the same as that specified for any other keyword.

The LNID is destroyed if this register is specified on another keyword.



Function Specifies the register containing the subarea address of the device at the other end of the virtual route.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as that specified for any other keyword.

```
►──VRID=(byte_register),──────────────────────────────────────────►
```

Function Specifies the register containing the virtual route identifier, which consists of the virtual route number and transmission priority.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as that specified for any other keyword; however, VRID and LNID may specify 2 different bytes of the same register.

```
►──WINDOW=(register),─────────────────────────────────────────────►
```

Function Specifies the register in which the window sizes will be returned.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as that specified for any other keyword.

You must specify an odd-numbered register.

If no window sizes have been specified, the content of this register is set to 0. If they have been specified, the minimum window size is in the high-order byte, and the maximum window size is in the low-order byte of the register.

```
►──WORK1=(register),──────────────────────────────────────────────►
```

Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as that specified for any other keyword.

You must specify an odd-numbered register.

```
►──WORK2=(register)──────────────────────────────────────────────►◄
```

Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register you specify must not be the same as that specified for any other keyword.

PCIL4—Update the Control Byte in the Level 4 Router Control Block

The PCIL4 macro updates the proper control byte in the level 4 router control block (L4B) with all bits set to 1 to indicate a SET function and to 0 to indicate a RESET function. If TYPE=NSPEC, the control block remains unchanged. It then issues an OUT X'7D' to set the PCI latch if MODE=SET (except when TYPE=L45WT). If MODE=RESET, the macro issues an OUT X'77' to reset the program controlled interruption (PCI) latch only if TYPE=NSPEC.

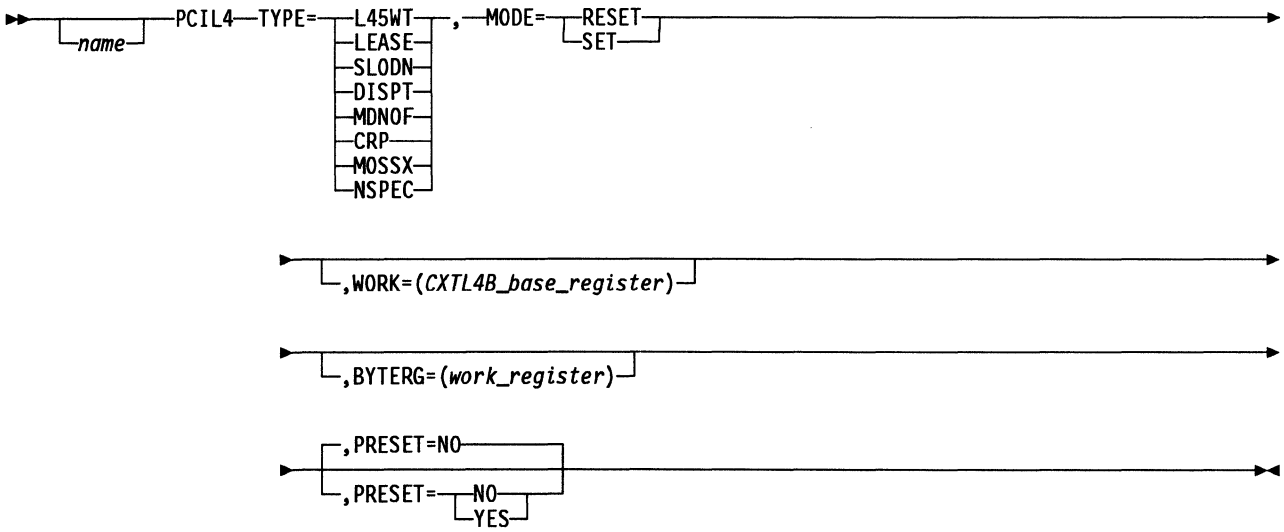
Whenever the level 4 or level 5 wait mask is reset by any routine (for example, the release service routine), that routine must ensure that all maskable-type PCI interrupt requests get processed. That routine, once the level 4 or level 5 wait mask is reset using the PCIL4 macro, should begin a PCI-hardware-interrupt-continued sequence so that the level 4 router can be reentered to process the PCI interrupt requests. Note that during the length of time the CCU is in the wait state, any PCI hardware interrupt for the level 4 or level 5 maskable-type PCI interrupt requests is reset by the router and an exit level 4 is executed.

Whenever you want to issue the CXTL4B DSECT outside the PCIL4 macro, you must either know that an EXTRN will be generated by the macro or code a GBLA &L4B, to be used as follows:

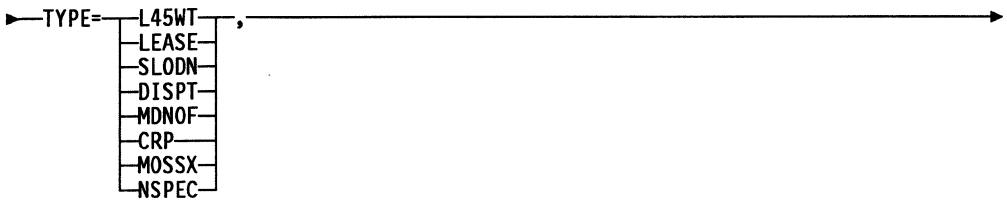
```
         AIF      (&L4B).LABEL
         EXTRN    CXTL4B
&L4B     SETA     1
.LABEL   continue
```

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the type of level 4 PCI being requested or controls the level 4 PCI processing.

TYPE=L45WT causes the level 4 or level 5 wait mask to be set. This request does not indicate a specific type of PCI but instead controls the level 4 router processing. When level 5 is unable to carry out a lease, the level 4 or level 5 wait mask is set so that level 4 PCI lease requests, slowdown requests, dispatcher requests, and SVC requests are masked off.

TYPE=LEASE causes a lease request. This occurs when level 5 has been disabled and parts of level 4 have been masked off because of an unsatisfied level 5 lease. When buffers again become available because they are released, this PCI type is issued so that level 4 gives control to a special entry point of the lease service routine.

TYPE=SLODN causes a slowdown request. This PCI type is issued so that level 4 will give control to the system slowdown manager.

TYPE=DISPT causes a dispatcher request. This PCI type causes the level 5 task dispatcher to be invoked.

TYPE=MNDOF causes a MOSS-down request. This PCI type is issued when level 1, level 3, or the mailbox manager has determined that the maintenance and operator subsystem (MOSS) is down or offline and causes the level 4 MOSSDOWN processing to get control.

TYPE=CRP causes a CRP entry service request. This PCI type occurs once an entry has been made in the check record pool by a level 1, level 2, or level 3 error-processing routine. This level 4 PCI causes the CRP processor in level 4 to get control so that a BER can be built from the CRP entry and sent to MOSS.

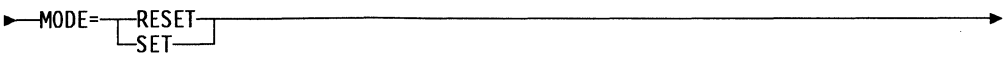
TYPE=MOSSX causes a MOSS transfer request. Whenever any NCP or EP component wishes to send a mailbox-out request to MOSS, a MOSS transmit buffer is placed on the MOSS outbound queue, and this PCI type occurs. This PCI causes the MOSS transfer section of the mailbox manager to get control so that a mailbox out request can be sent to MOSS.

TYPE=NSPEC causes a nonspecific request. This PCI type causes the level 4 PCI bit to be set or reset. No specific level 4 PCI type is set in the level 4 router control block.

Format L45WT, LEASE, SLODN, DISPT, MDNOF, CRP, MOSSX, NSPEC.

Default None.

Remarks You must code this keyword.



Function Specifies whether the PCI is to be set or reset.

MODE=SET causes the proper flag byte in the level 4 router control block to be set to X'FF', unless TYPE=NSPEC. It also causes an OUT X'7D' to set the level 4 PCI hardware latch unless TYPE=L45WT.

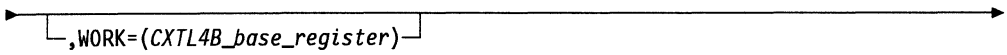
MODE=RESET causes the proper flag byte in the level 4 router control block to be set to X'00', unless TYPE=NSPEC. An OUT X'77' is issued to reset the PCIL4 latch only if TYPE=NSPEC or CRP.

Format SET or RESET.

Default None.

Remarks You must code this keyword.

If MODE=RESET, an OUT X'77' to reset the PCI latch is issued by the level 4 router and is not generated here unless TYPE=NSPEC or TYPE=CRP.

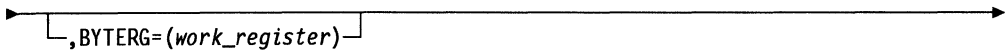


Function Specifies the register to use to establish addressability to the level 4 router control block.

Format Register notation.

Default The macro will save and use register 4.

Remarks This parameter is optional. You can specify register 1, 2, 3, 4, 5 or 7. If you do not specify a register, register 6 must point to an available save area. The generated code is more efficient if you code this parameter and have no need to save the specified register.

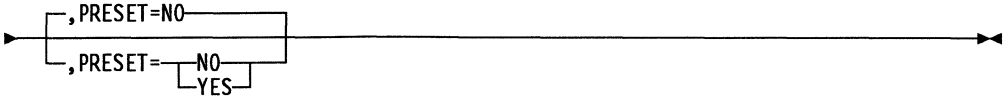


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register (byte) notation.

Default The macro will save register 3 and use 3(1).

Remarks This parameter is optional. You can specify register 1, 3, 5, or 7. If you do not specify a register, register 6 must point to an available save area. The generated code is more efficient if you code this parameter and have no need to save the specified register.



Function Specifies whether the macro needs to establish addressability to the CXTL4B control block.

If PRESET=YES, the macro does not generate the load address (LA) to establish addressability, but assumes addressability already exists in the register specified for the WORK keyword &WORK or its default (R4).

If PRESET=NO, the macro generates the following:

```
LA &WORK,CXTL4B
```

Format YES or NO.

Default NO.

Remarks It is your responsibility to maintain addressability between calls to this macro if you want to code PRESET=YES.

PERFORM—Transfer Control to a Routine and Establish Return Linkage

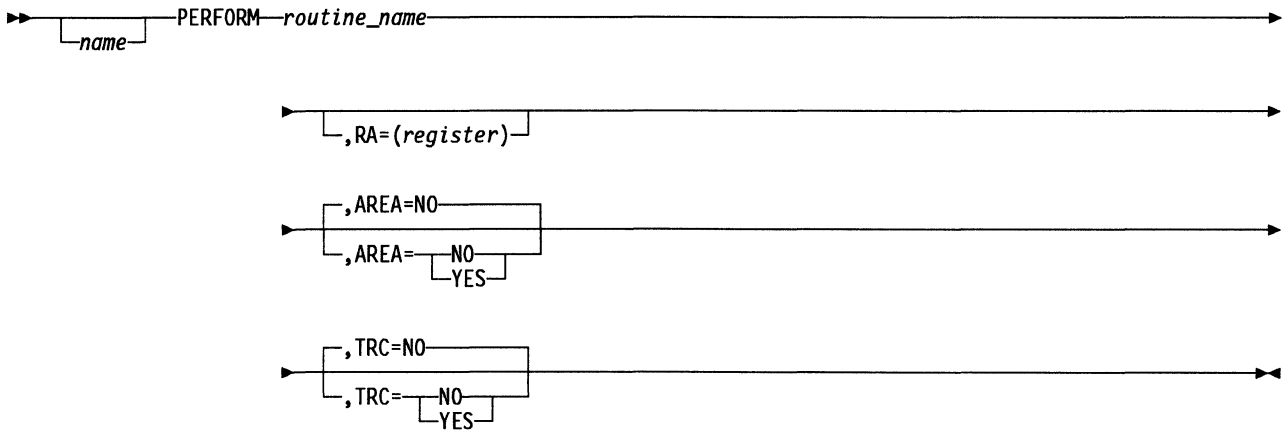
The PERFORM macro transfers control to a specified routine and establishes linkage back to the instruction immediately following the PERFORM macro statement. An optional keyword will update register 6 to point to the next save area in the save area chain. Register 6 should point to the current save area when the PERFORM macro is executed. The entry point to the routine to be performed should be established by the ROUTINE macro. The performed routine is responsible for saving and restoring any registers it alters.

The PERFORM, ROUTINE, and RETURN macros provide a reentrant form of linkage if unique save areas are available. For example, levels 1, 3, 4, and 5 could share suitably coded common routines because each level has a separate save area chain.

Register 0 is not allowed for register parameters.

Note: If TRC=YES, the issuing module must contain an EXTRN for PERFTRC.

Syntax



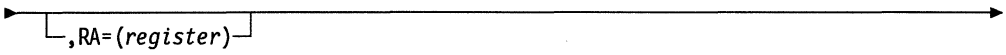
Parameters

Function Specifies the routine to be given control.

Format Register or label notation.

Default None.

Remarks Register 6 is not allowed.



Function Specifies the return address register.

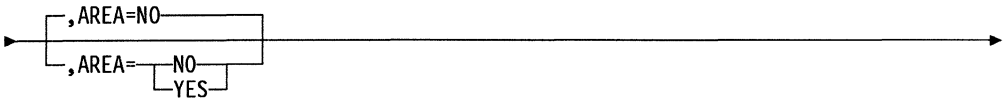
Format Register notation.

Default Register 7.

Remarks Register 6 is not allowed.

The RA=(r) specified on PERFORM must match the RA=(r) specified on the ROUTINE macro referred to by the *routine name* keyword.

The linkage default is register 7 on both PERFORM and ROUTINE and is the preferred register to use.

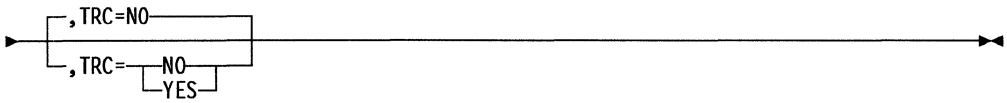


Function Specifies which save area is to be used. AREA=YES indicates that register 6 is to be updated to point to the next save area in the chain before performing the routine, then restored on the return. AREA=NO indicates that register 6 is to remain pointing to the current save area.

Format YES or NO.

Default NO.

Remarks Code YES only if the current save area is in use. The current save area is in use if a SAVE macro with AREA=NO was issued prior to this PERFORM macro. The performed routine must use the next save area in that instance. Refer to the SAVE macro and the RESTORE macro for more information.



Function Specifies whether an entry for the address of the PERFORM macro expansion will be added to the dispatch trace table during program execution.

Format YES or NO.

Default NO.

Remarks Coding TRC=YES saves registers 1, 3, 4, and 5 in the current save area.

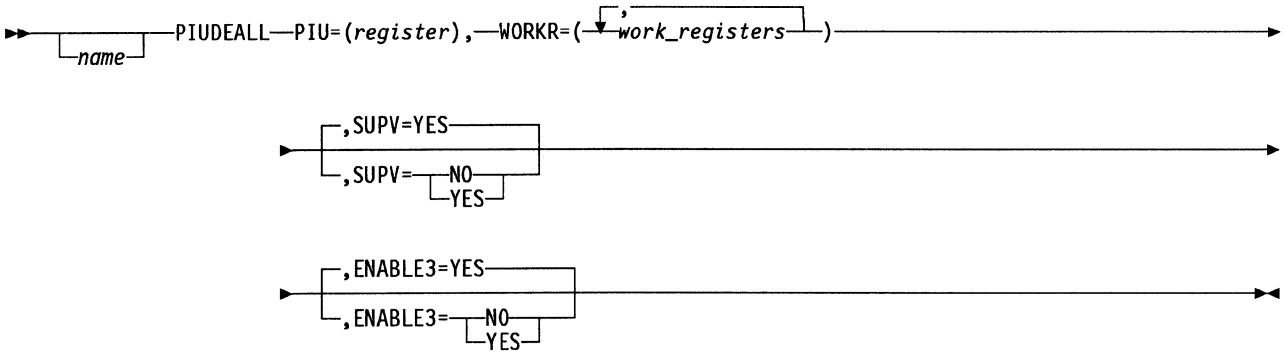
Use of TRC=YES can degrade performance.

Note: If TRC=YES, the issuing module must contain an EXTRN for PERFTRC.

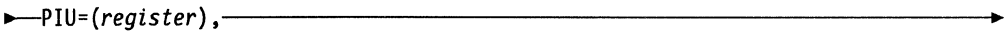
PIUDEALL—Deallocate a PIU from an Inbound Boundary Pool

The PIUDEALL macro deallocates a path information unit (PIU) from the inbound boundary pool to which it has been allocated. If the PIU is marked as allocated, the virtual route vector table index (VVTI) is used to obtain the virtual route control block (VRB) where the count and threshold are kept. The count is decremented and the PIU is marked as deallocated. If the count is less than the threshold, the wake-up task (CXDCXSD) is triggered.

Syntax



Parameters



Function Specifies the register containing the address of the PIU that is to be deallocated.

Format Absolute register notation.

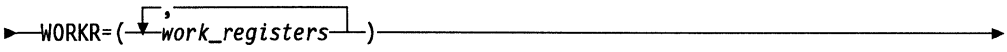
Default None.

Remarks The register you specify must not be the same as the one specified for the WORKR keyword.

Register 6 is not allowed with SUPV=YES.

Register 3 or 7 is preferred.

Register 1 is not allowed with SUPV=NO.



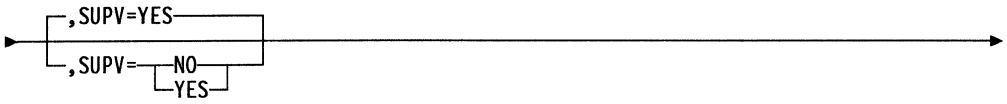
Function Specifies one or two work registers, the contents of which may be altered during execution of the macro.

Format You can specify a maximum of two registers from register 1, 2, 3, 4, 5, or 7.

Default None.

Remarks The registers you specify must not be the same as the one specified for the PIU keyword.

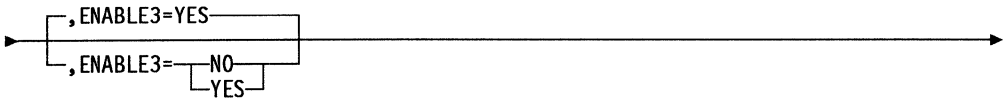
This parameter is not valid with SUPV=NO.



Function Specifies whether the macro is to issue a supervisor request to level 4 or link directly to the PIUDEALL service routine.

Format YES or NO

Default YES.



Function Specifies whether level 3 interupts are enabled at the completion of the PIU deallocation. If ENABLE3=YES, both level 2 and level 3 interrupts are enabled at the completion of the PIU deallocation. If ENABLE3=NO, a level 4 routine inhibits level 3 interrupts and issues a PIU deallocation. Only level 2 interrupts are enabled at the completion of the PIU deallocation.

Format YES or NO

Default YES.

PIUEND—Get the Last PIU Data Byte and a Buffer Pointer

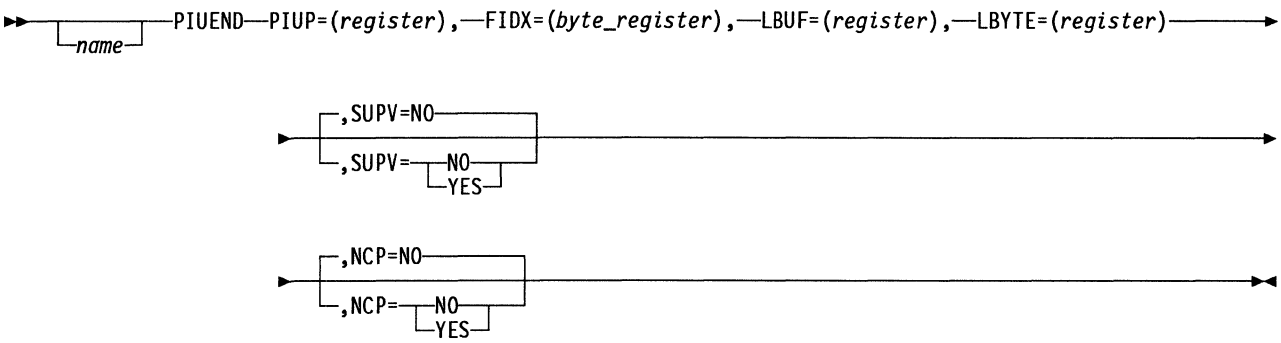
The PIUEND macro returns a pointer to the last buffer with data and the last byte with data in the path information unit (PIU). The macro requires a pointer to the first buffer of the PIU.

You must put a valid save area pointer in register 6. If the PIU is an FID1 with a correct transmission header data count, code the FIDX keyword as 1 for efficiency. In all other cases, code FIDX as 0.

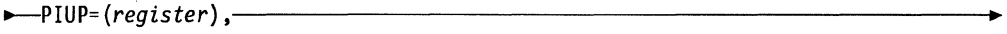
The buffer data counts in the PIU must be correct before you call the macro.

Register 0 is not allowed for register parameters.

Syntax



Parameters

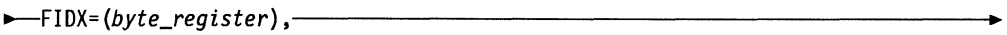


Function Specifies the register containing a pointer to the first buffer of the PIU.

Format Register notation.

Default None.

Remarks Register 6 is not allowed. The register you specify must not be the same as that specified for any other keyword. The value in the selected register is preserved.



Function Specifies a byte register containing one of the two values shown below:

X'01' Use this value *only* when the PIU is a FID1 with a correct data count in the transmission header.

X'00' Use this value for all other cases or when the FID type is not known.

Format Byte register notation.

Default None.

Remarks The register must be the high or low byte of an odd-numbered register and must be used only for this keyword. The value of this register is preserved.

FIDX=X'00' works for all cases. If you know that the PIU is FID1, use X'01' for efficiency.

Function Specifies the register that will contain the pointer to the last buffer in the PIU that contains data.

Format Register notation.

Default None.

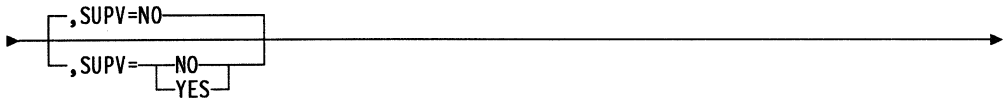
Remarks Register 6 is not allowed. The register you specify must not be the same as that specified for any other keyword.

Function Specifies the register that will contain the pointer to the last byte of data in the PIU.

Format Register notation.

Default None.

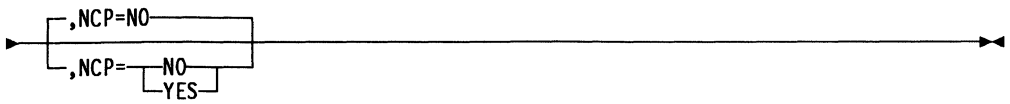
Remarks Register 6 is not allowed. The register you specify must not be the same as that specified for any other keyword.



Function Specifies the level in which the issuer is running. SUPV=NO indicates that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.



Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES indicates that the macro is to be expanded inline. NCP=NO indicates that an SVC call is to be used.

Format YES or NO.

Default NO.

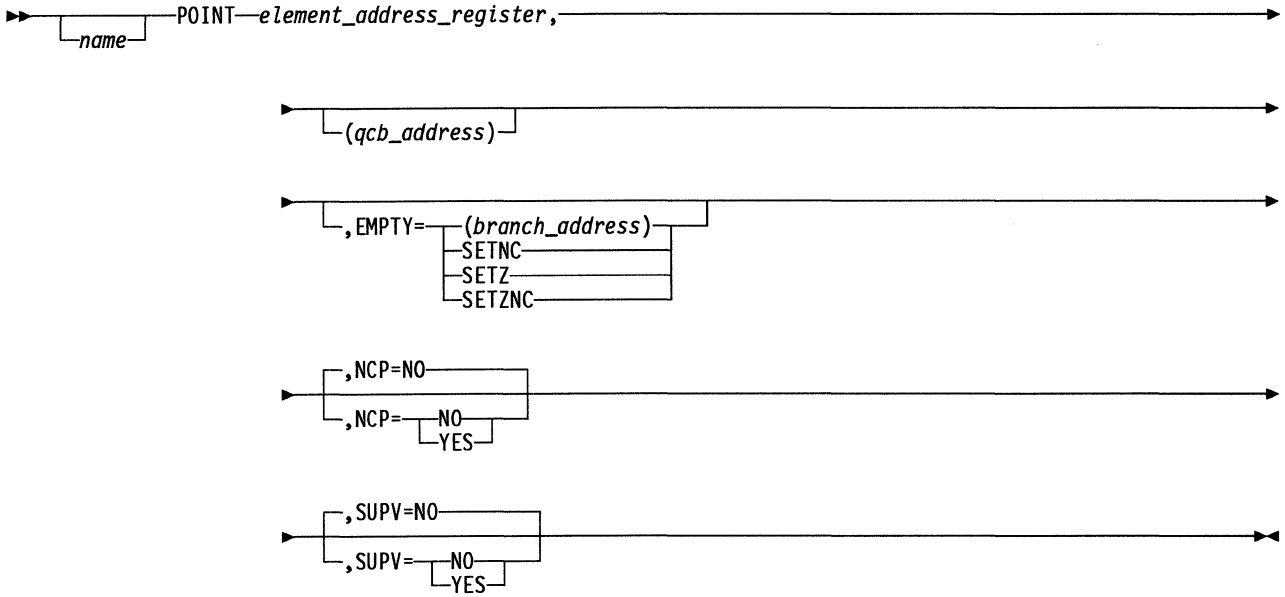
Remarks NCP is valid only when SUPV=NO.

POINT—Return the Address of the First Element in a Queue

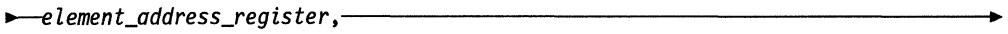
The POINT macro returns the address of the first element. This first element can be either a block control unit (BCU) or a queue control block (QCB) that is queued on a specified system queue without dequeuing the element. You can also use the POINT macro to determine if the specified system queue is empty without returning the address of the first element (when present). The Z latch is set to 1 if the queue is empty and to 0 if it is not empty. The C latch is set opposite to the Z latch.

Register 0 is not allowed for register parameters.

Syntax



Parameters



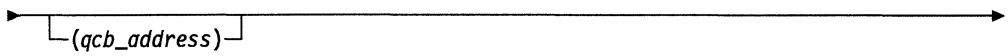
Function Specifies a register to receive the address of the located element.

Format Register notation.

Default None.

Remarks If there are currently no elements queued on the specified queue, the address in the specified register is set to 0.

Register 3 is standard.



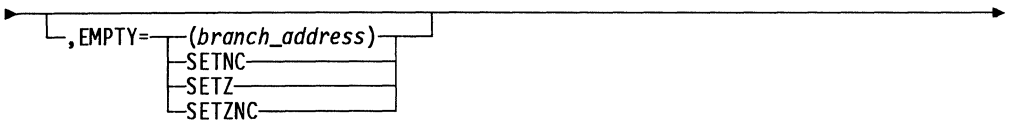
Function Specifies the address of the QCB governing the queue in which the element is located.

Format Register or label notation.

Default If SUPV=YES, there is no default. If SUPV=NO, the QCB that activated the issuing task is assumed.

Remarks If NCP=NO, register 1 is not allowed.

This keyword must be specified if SUPV=NO and NCP=YES.



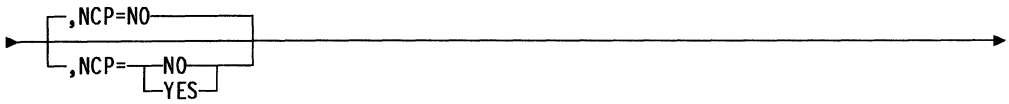
Function Either specifies the address to be given control or specifies whether the C latch, Z latch, or both are to indicate whether the specified queue is currently empty.

Format Label notation, register notation, SETZNC, SETZ or SETNC.

Default The next instruction after the macro instruction is given control.

Remarks Register 1 is not allowed.

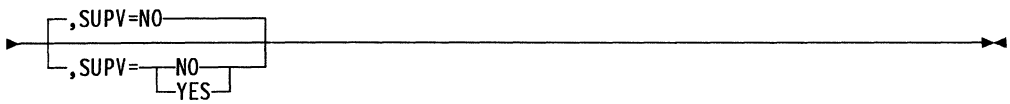
For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the specified queue is empty or to 0 if not empty. For SETZNC or SETNC, the C latch is set to 0 if the specified queue is empty or to 1 if not empty.



Function Specifies the type of code generated based on the issuer's storage protection key. Valid only when SUPV=NO. If NCP=YES, inline code is generated. If NCP=NO, an SVC is generated.

Format YES or NO.

Default NO.



Function Specifies the level in which the user is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

POSTUACB—Implement User Line Control

The POSTUACB macro in your level 2 interrupt handler implements user line control. It places the user adapter control block (UACB) directly into the NCP's communication interrupt control program (CICP) queue and causes a program-controlled interrupt to level 3. Use the FINDUACB macro to obtain the UACB pointer. The UACB pointer must be in the register specified by the UACBREG keyword. When your level 2 interrupt handler is finished executing, it must place the UACB pointer into register 2.

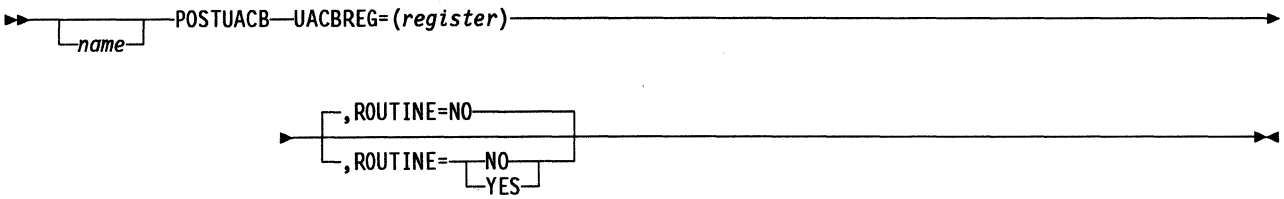
After the POSTUACB macro is executed, the level 3 interrupt processor does the following:

- Tests to see if the device is supported by NCP or by user code
- Retrieves the UACB
- Transfers control to the address in the GCBL3 field of the group control block (GCB).

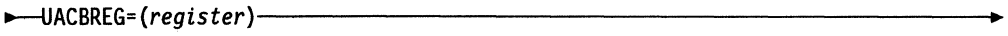
The level 3 routine obtains the UACB pointer from register 2.

Note: The use of this macro requires that the assembly include the DSECTs XCXTGCB and XCXTXDA. Addressing is handled internally by the macro.

Syntax



Parameters

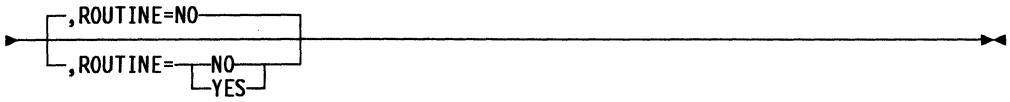


Function Specifies the register containing the UACB pointer to be passed to the level 3 routine.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



Function Specifies whether you want the code to expand inline to post the UACB (ROUTINE=NO) or branch-and-link to a common routine (CXBDQACE) that posts the UACB (ROUTINE=YES). ROUTINE=NO expands the macro inline.

Format YES or NO.

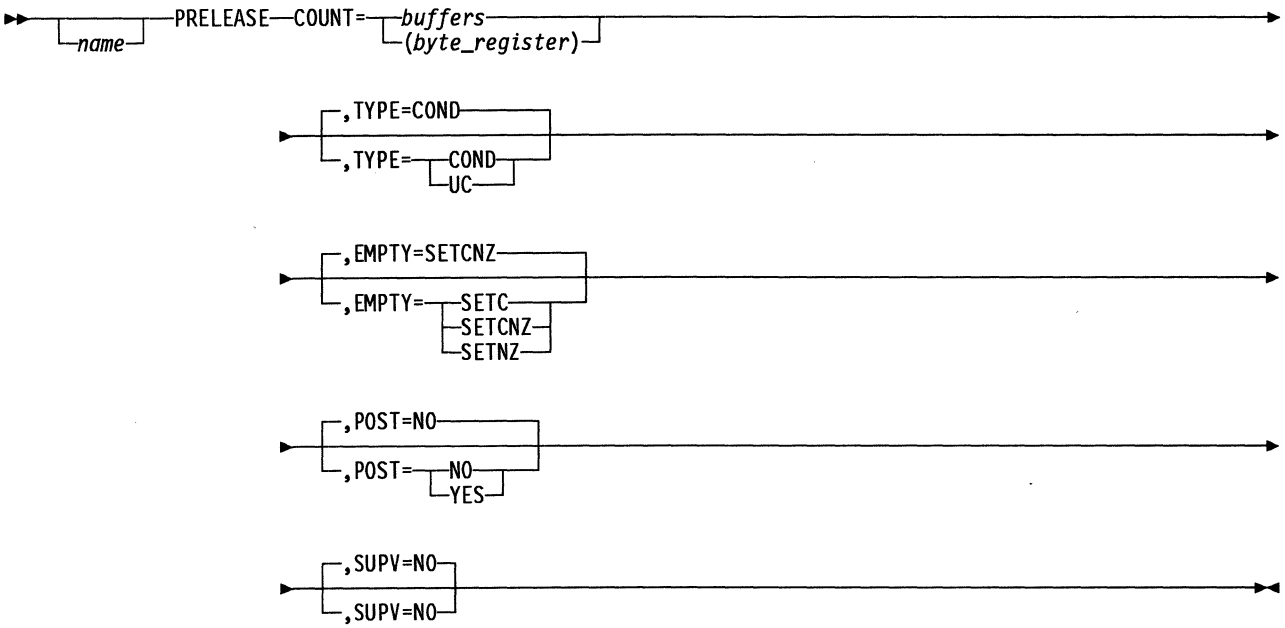
Default NO.

Remarks If you code ROUTINE=YES, the UACB must be in register 2 before the macro is invoked. Also, the contents of registers 1, 4, and 7 are destroyed.

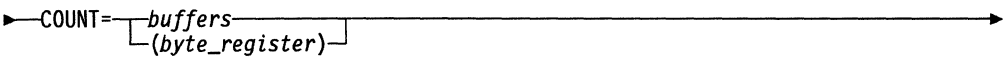
PRELEASE—Reserve Buffers for Future Use

The PRELEASE macro allows a program to reserve one or more buffers for future use. If the buffers requested are not available, the requesting program has the option of being posted until the buffers are available. Other level 5 tasks continue while the requesting program waits. When the buffers become available the task is dispatched.

Syntax



Parameters

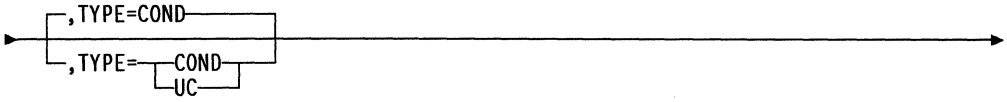


Function Specifies the number of buffers to be preleased.

Format Low-byte register (absolute notation) or numerical value (absolute notation).

Default None.

Remarks Registers 0 and 1 are not allowed.



Function Specifies whether the PRELEASE request is conditional or unconditional.

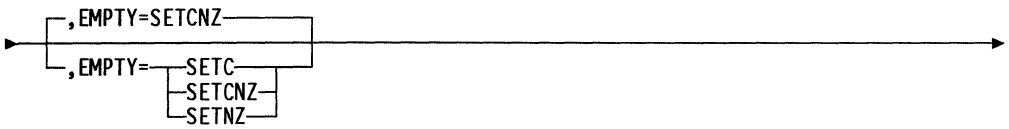
Format COND or UC.

Default COND.

Remarks In a conditional request (TYPE=COND), buffers will not be preleased if the number specified causes the free-buffer count to drop below the slowdown threshold.

An unconditional request (TYPE=UC) will be honored if enough buffers are available.

In earlier releases of NCP, TYPE=SLDN and TYPE=CWALL indicated that the PRELEASE request was bounded by the slowdown and communications wall thresholds, respectively. Revisions to the PRELEASE macro replace TYPE=SLDN with TYPE=COND, and TYPE=CWALL with TYPE=UC.



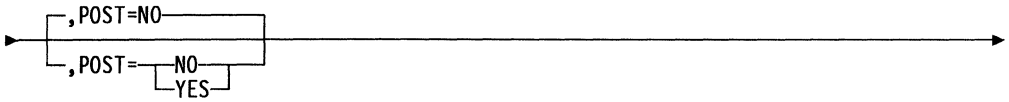
Function Specifies that either the C latch or Z latch is set if PRELEASE cannot be satisfied.

Format SETC, SETNZ, or SETCNZ (either C, or NOT Z).

Default SETCNZ.

Remarks If PRELEASE is satisfied, C latch is set to 0 and Z latch is set to 1.

If PRELEASE is not satisfied, C latch is set to 1 and Z latch is set to 0.



Function Specifies whether the current queue control block (QCB) is posted to wait until one or more buffers are available.

Format YES or NO.

Default NO.

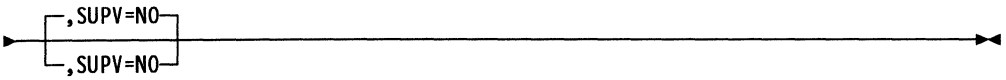
Remarks If POST=YES and the PRELEASE is not satisfied, the following occurs:

- This PRELEASE is flagged as being unsuccessful.
- The task is queued on a prelease dispatch queue for later dispatching.
- When buffers are available, they are preleased and the task is dispatched.

When a task is dispatched from a prelease dispatch queue (unconditional or conditional) the buffers will have been preleased before execution at the task entry point. Check the preleased triggered bit of the QCB (in the XYZSTATP field) to see if the buffers have already been preleased.

For example:

```
IF 'preleased triggered' bit is OFF
   THEN (dispatched off normal queue)
        PRELEASE POST =YES,...
        IF not successful,
           THEN
                SYSXIT QPOST=NO (if POST=YES on PRELEASE)
        ENDIF
ENDIF
```



Function Specifies that the issuer is running in level 5.

Format NO.

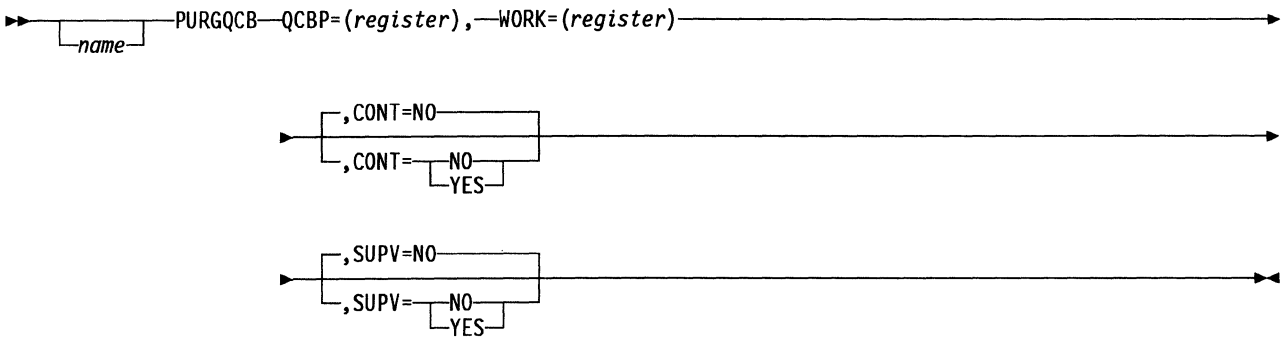
Default NO.

PURGQCB—Purge a Queue

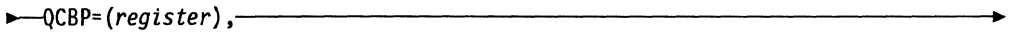
The PURGQCB macro purges a queue using the DEQUE and RELEASE macros when given a pointer to a queue control block (QCB).

Register 0 is not allowed for register parameters.

Syntax



Parameters

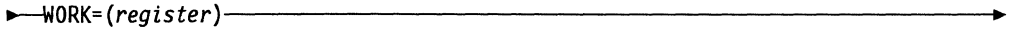


Function Specifies the register containing a pointer to the QCB for the queue to be purged.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 6 is allowed. The register cannot be the same as that used for WORK. The value of the register is preserved. Register 2 is standard. The elements on the QCB must contain event control blocks (ECBs).

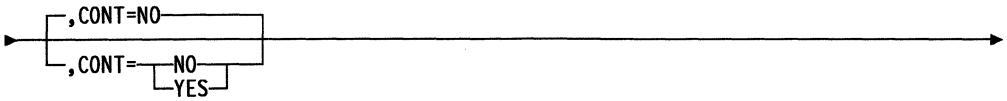


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

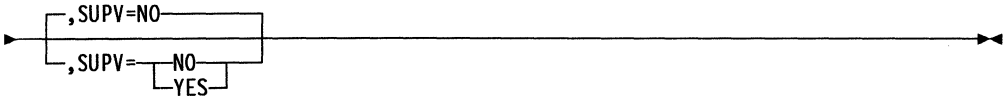
Remarks Neither register 1 nor register 6 is allowed. The register cannot be the same as that used for QCBP. The contents of the register are not preserved. Register 3 is standard.



Function Specifies whether the queue is a contention queue.

Format YES or NO.

Default NO.



Function Specifies the level in which the issuer is running. SUPV=NO indicates that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

PUTBYTE—Store a Data Byte from a Register into a BCU

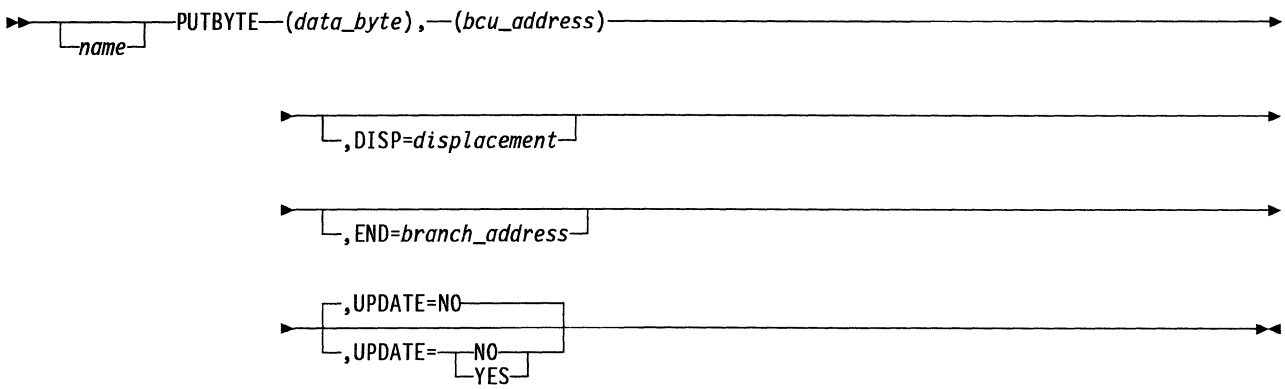
The PUTBYTE macro stores a specified data byte from a register into a specified position in a block control unit (BCU).

The position into which the register is to be stored is specified in two ways. The first way is to specify a displacement value in the DISP keyword. Using this displacement value, the supervisor calculates the appropriate position in the appropriate buffer of the BCU and stores the register byte. The supervisor then increments the displacement value by 1 and saves it. The second way is to omit the DISP keyword. The supervisor uses the displacement saved from the last PUTBYTE macro to calculate the position where the byte is to be stored. It then increments the displacement value by 1 and saves it.

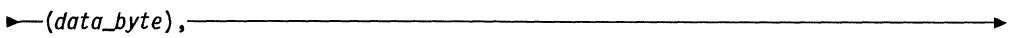
You can use the PUTBYTE macro in two modes, sequential and update. Use sequential mode when a BCU is being built 1 byte at a time. In this mode, the PUTBYTE macro causes the text count in the basic transmission unit (BTU) and the data count in the buffer to be incremented by 1. Use update mode when the data within an existing BCU is being modified. In update mode, the PUTBYTE macro alters neither the BTU text count nor the buffer data count.

Register 0 is not allowed for register parameters.

Syntax



Parameters

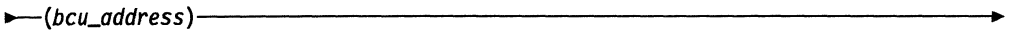


Function Specifies the register containing the data byte to be stored.

Format Byte register notation.

Default None.

Remarks Register 1(0) is standard.

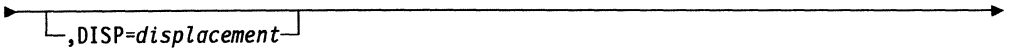


Function Specifies the register containing the address of the first byte of the first buffer of the BCU.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

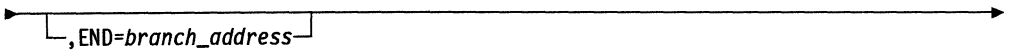


Function Specifies the displacement from the first byte in the BCU to the position where the data byte is to be stored.

Format Absolute or register notation.

Default Computed value is used.

Remarks Register 1 is not allowed.

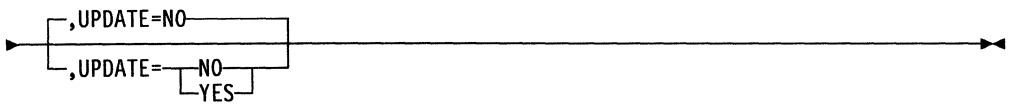


Function Specifies the address to be given control if the specified or computed displacement exceeds the total size of the BCU.

Format Register or label notation.

Default The issuing program tests register 1(1) bit 0. If the bit tested is 0, the byte was not stored successfully.

Remarks Register 1 is not allowed.



Function Specifies the mode in which the PUTBYTE macro is to be used. UPDATE=YES specifies update mode; UPDATE=NO specifies sequential mode.

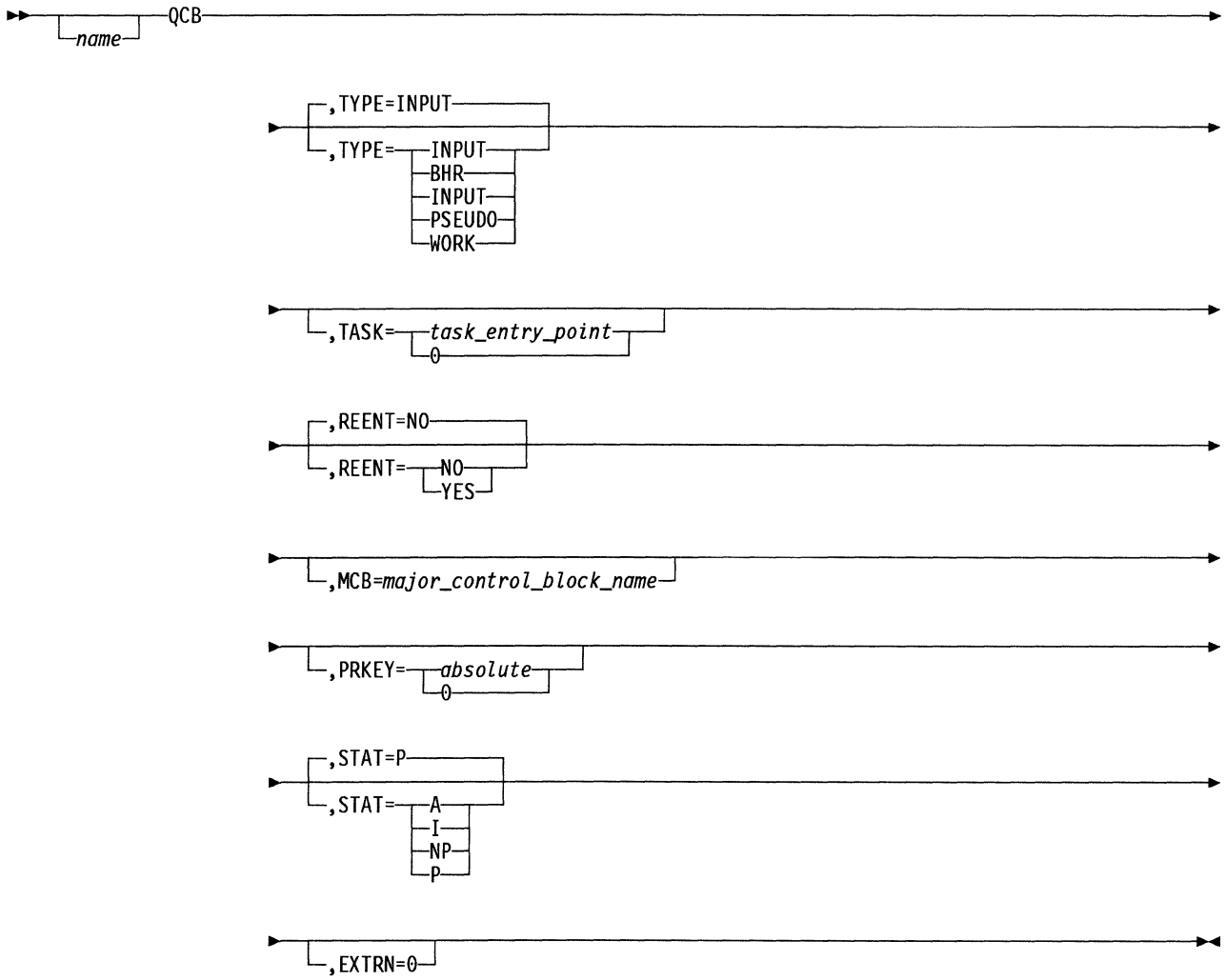
Format YES or NO.

Default NO.

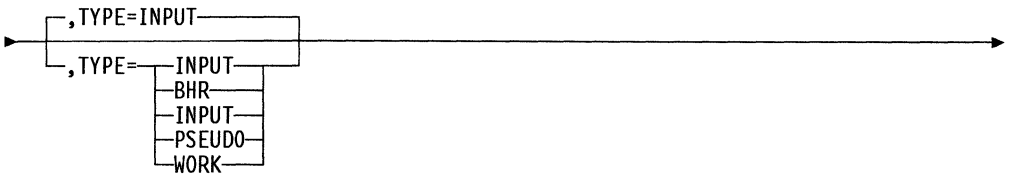
QCB—Define Storage for a System Queue Control Block

The queue control block (QCB) macro defines the storage required to establish a system queue control block. The name field is optional for work queues and is required for all other types of queues.

Syntax



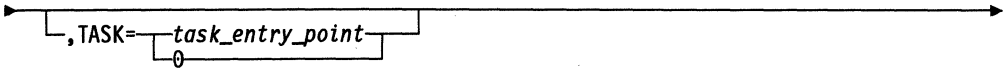
Parameters



Function Specifies the type of QCB to be generated. Code the block handling routine (BHR) if the associated task executes an EXECBHR macroinstruction.

Format INPUT, PSEUDO, WORK, or BHR.

Default INPUT.

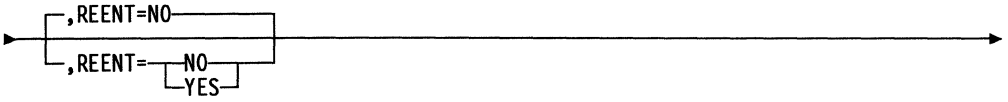


Function Specifies the entry point that is to be activated when data is enqueued to the QCB being defined.

Format Label notation. Entry point of the task to be activated, or 0 if it is not known at assembly.

Default None.

Remarks If TYPE=WORK, this keyword has no meaning; otherwise, it is required.

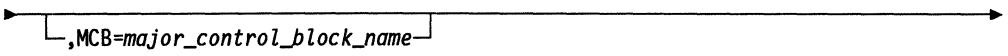


Function Specifies whether the task associated with the queue being defined is reentrant, that is, it is associated with more than one QCB and issues a WAIT macro.

Format YES or NO.

Default NO.

Remarks If TYPE=WORK, this keyword has no meaning.

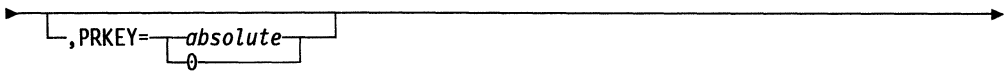


Function Specifies the name of the major control block within which the QCB is being assembled.

Format Label notation.

Default None.

Remarks If TYPE=WORK, this keyword has no meaning.

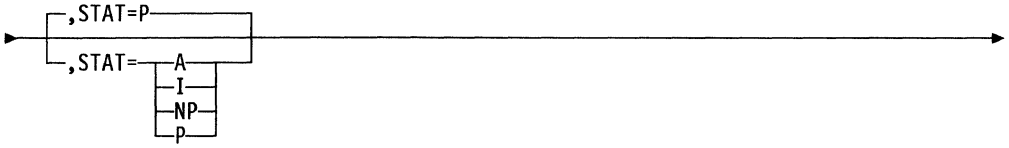


Function Specifies the protection key under which the task associated with the queue being defined executes.

Format Any value from 0 to 7.

Default 0.

Remarks If TYPE=WORK, this keyword has no meaning.



Function Specifies the initial status of the QCB as either appendage, immediate, productive, or nonproductive.

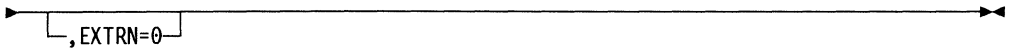
Format The format codes are:

A Appendage (highest priority)
I Immediate (second-highest priority)
P Productive (third-highest priority)
NP Nonproductive (lowest priority).

Default P.

Remarks The status of the QCB is the dispatching priority that is used when the QCB is triggered.

If TYPE=WORK, this keyword has no meaning.



Function Suppresses the generation of an external reference to the task entry point specified by TASK. Specify this keyword if the QCB macro is in the same module as its task entry point.

Remarks If TYPE=WORK, this keyword has no meaning.

QPOST—Flag a Task as Ready and Eligible for Reactivation

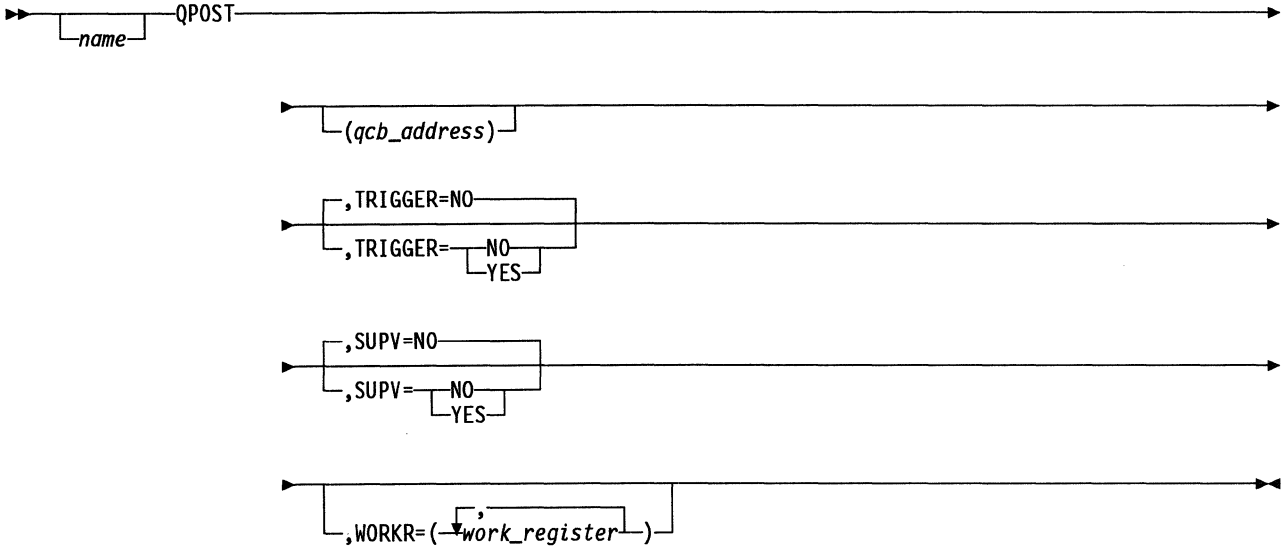
The QPOST macro flags a specified task as being in the ready state and eligible for reactivation. If any element is currently enqueued on the specified task input queue, the task is triggered and scheduled for execution.

For additional information on how the QPOST macro affects task states, refer to "Task Management" in *NCP and EP Reference*.

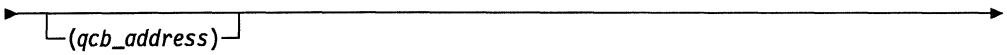
Register 0 is not allowed for register parameters.

Note: This macro uses the current save area and assumes that this save area can be overwritten.

Syntax



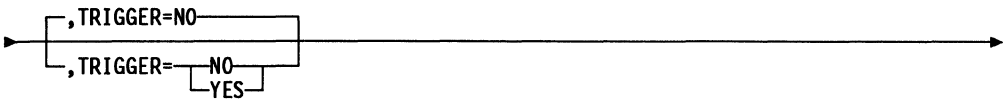
Parameters



Function Specifies the register containing the address of the input or pseudoinput queue control block (QCB) governing the task to be flagged.

Format Register or label notation.

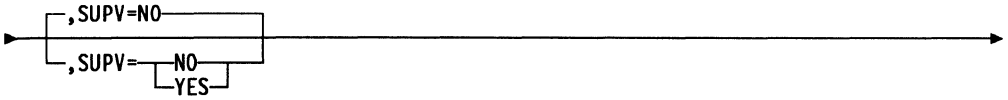
Default If SUPV=YES, there is no default. If SUPV=NO, the QCB that activated the issuing task is posted.



Function Specifies whether the specified task is to be rescheduled (YES) or merely flagged as ready (NO).

Format YES or NO.

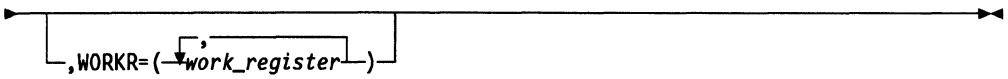
Default YES.



Function Specifies whether the macro is to issue a supervisor request to level 4 or is to link directly to the QPOST routine.

Format YES or NO.

Default NO.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default No work registers.

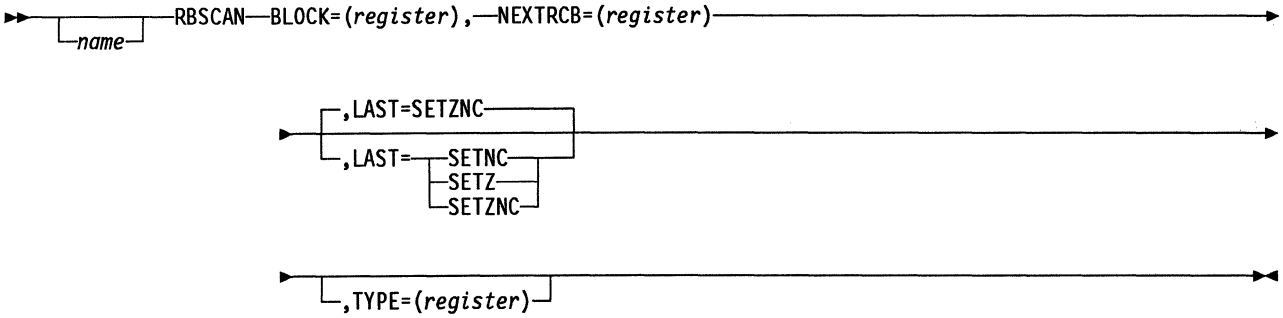
Remarks Register 6 is not allowed.

RCBSCAN—Get the Address of an RCB in a Chain Associated with a Virtual Route

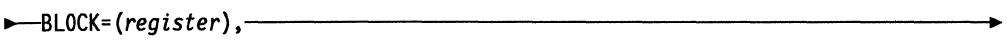
The RCBSCAN macro scans the chain of resource connection blocks (RCBs) associated with a virtual route and returns the address and type of the next RCB on the chain. This macro expands inline.

Register 0 is not allowed for register parameters.

Syntax



Parameters



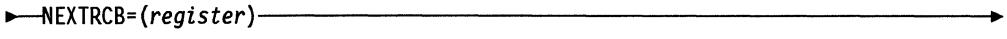
Function Specifies the register containing the starting RCB or VRB address in the chain to scan.

Format Register notation.

Default None.

Remarks The specified register must not be the same as that specified for the NEXTRCB or TYPE keywords.

If an address of a VRB is input, the first RCB in the chain is returned.



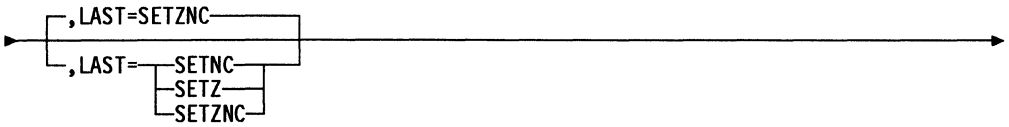
Function Specifies the register that returns the address of the next RCB on the chain of RCBs associated with a virtual route.

Format Register notation.

Default None.

Remarks The content of the specified register is set to 0 if there are no more RCBs on the chain.

The specified register must not be the same as that specified for the BLOCK or TYPE keywords.



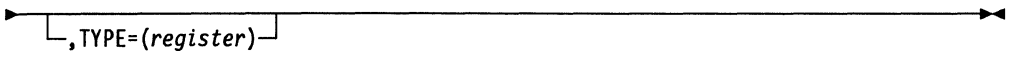
Function Specifies whether the C latch, the Z latch, or both are to indicate if the end of the chain is found.

Format SETZ, SETNC, or SETZNC.

Default SETZNC.

Remarks For SETZNC or SETZ, the Z latch is set to 1 if this is not the end of the chain or to 0 if this is the end. For SETZNC or SETNC, the C latch is set to 0 if this is not the last RCB on the chain or to 1 if this is the last.

The conditions for setting the C and Z latches as described above apply when the VRB has no RCBs chained to it.



Function Specifies the register in which the RCBSCAN macro returns the block type from the RCBBLK field.

Format Byte register notation.

Default None.

Remarks If the chain is empty, the contents of this register is set to 0.

The specified register must not be the same as that specified for the NEXTRCB or BLOCK keywords.

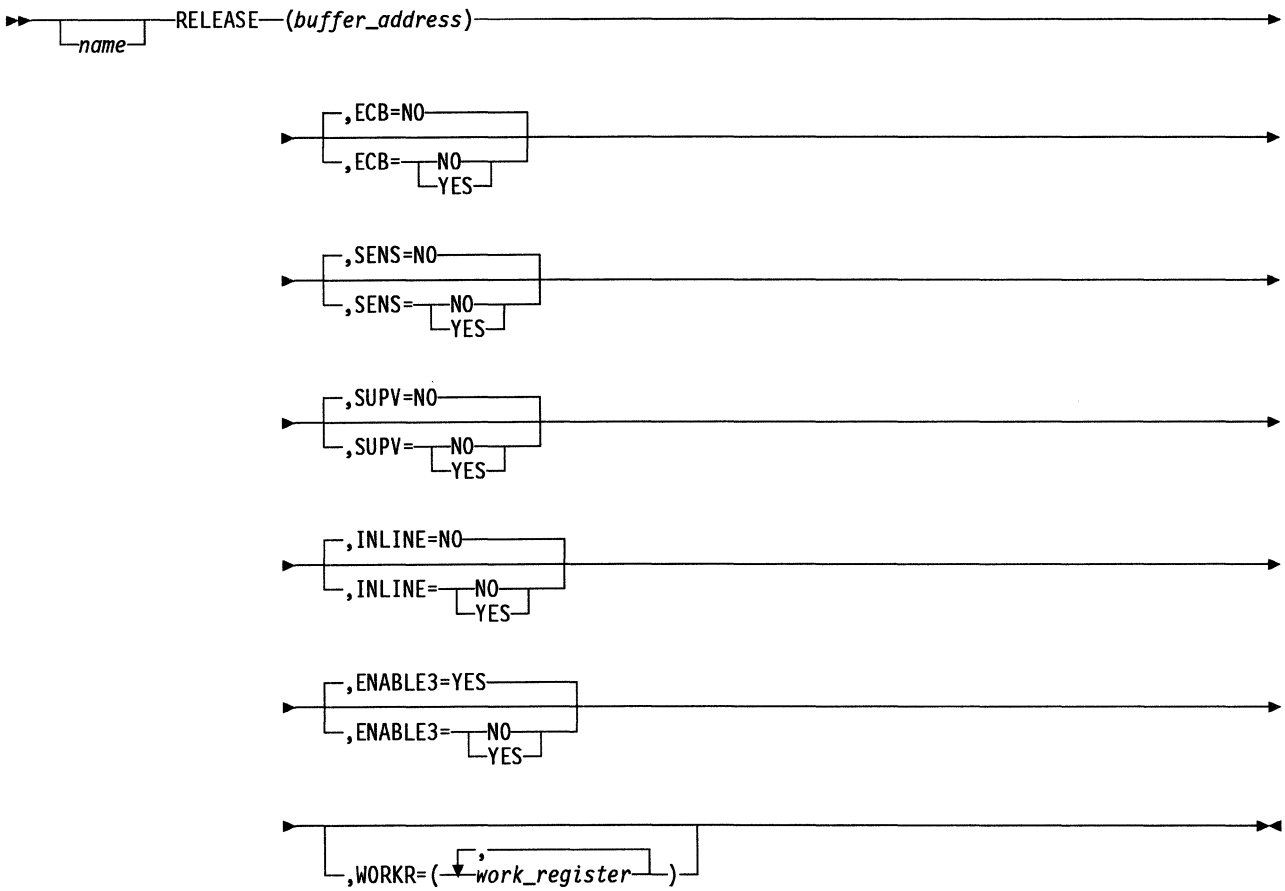
RELEASE—Return a Buffer or Buffer Chain to the System Free-Buffer Pool

The RELEASE macro returns a buffer or buffer chain to the system free-buffer pool.

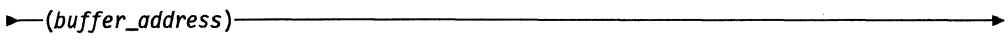
Register 0 is not allowed for register parameters.

Note: This macro uses the current save area and assumes that this save area can be overwritten.

Syntax



Parameters



Function Specifies the register containing the address of the buffer or buffer chain to be released.

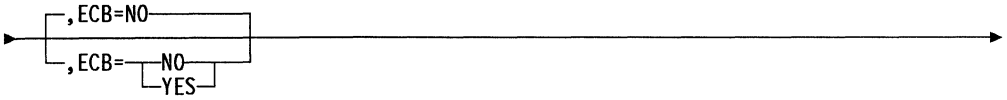
Format Register notation.

Default None.

Remarks Register 1 is not allowed.

If SUPV=YES, register 3 is standard and register 6 is not allowed.

The buffer address must be specified as register 3 if INLINE=YES is specified.



Function Specifies whether the first or only buffer being released contains an event control block (ECB).

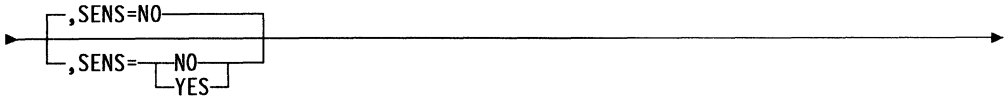
Format YES or NO.

Default NO.

Remarks This keyword is ignored if INLINE=YES, unless the system is in slow-down mode.

For ECB=YES, an ECB initialization function must have been carried out before the RELEASE macro is invoked.

The ECB keyword does not affect the number of buffers being released.



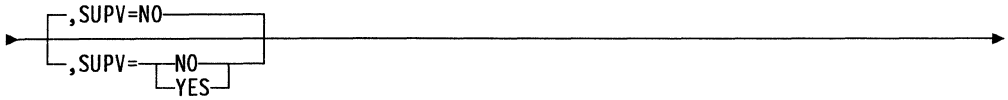
Function Specifies whether the data in the buffers is sensitive.

Format YES or NO.

Default NO.

Remarks YES specifies that the data is sensitive and that the supervisor is to set the buffers to 0 before returning them to the system free-buffer pool.

This keyword is ignored if INLINE=YES, unless the system is in slow-down mode.

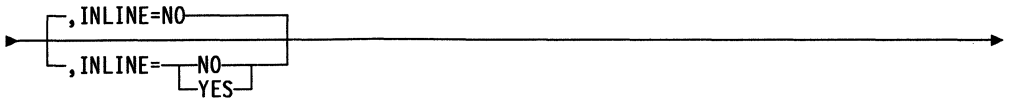


Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

Remarks You must specify SUPV=YES if you specify INLINE=YES.



Function YES specifies that a fast inline release will be generated. The inline code checks whether the system is in slowdown or pseudoslowdown mode. If the system is in slowdown mode, existing release code will be executed; otherwise, a "fast" release will do the following:

- Chain the chain of buffers passed in register 3 to the free-buffer pool chain
- Update the buffer tag in each buffer of the chain that is passed
- Update the current free-buffer pool count.

Format YES or NO.

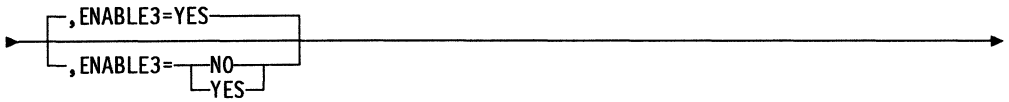
Default NO.

Remarks SENS and ECB are ignored if the fast inline code is executed.

You must specify SUPV=YES.

The buffer address must be in register 3.

If INLINE=YES is coded, dsect XCXTABN must be included in the assembly.

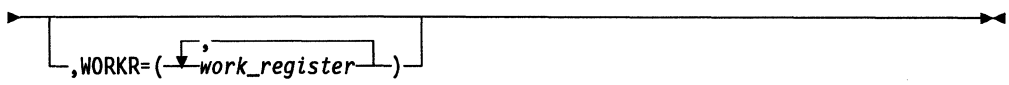


Function Specifies whether level 3 interrupts are to be enabled at the completion of the RELEASE. If ENABLE3=YES, both level 2 and level 3 interrupts are enabled at the completion of the RELEASE. ENABLE3=NO allows a level 4 routine to inhibit level 3 interrupts and then issue a RELEASE, and only level 2 interrupts will be enabled at the completion of the RELEASE.

Format YES or NO.

Default YES.

Remarks This parameter is valid only when you code SUPV=YES.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default No work registers.

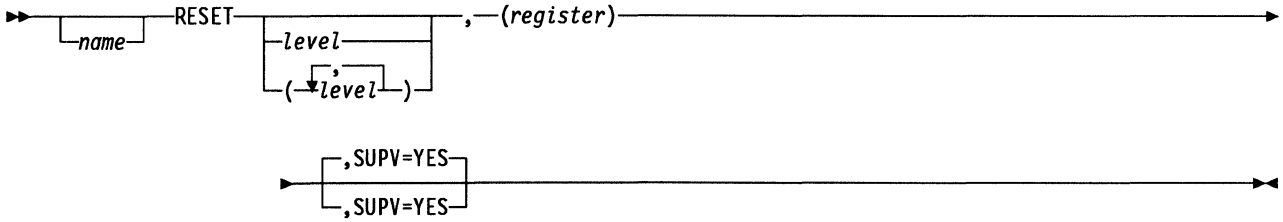
Remarks Register 6 is not allowed. WORKR is valid only if INLINE=YES.

RESET—Enable an Interrupt Level

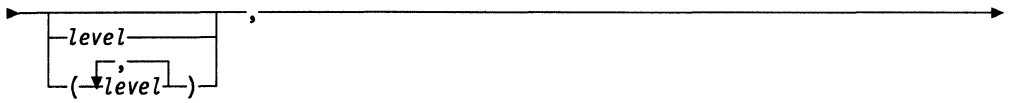
The RESET macro loads a specified register with the appropriate masks and issues an output instruction to allow interrupts at the specified interrupt levels. This macro can be used in program levels 1, 2, 3, and 4.

Note: This macro can be executed by interrupt-level code only.

Syntax



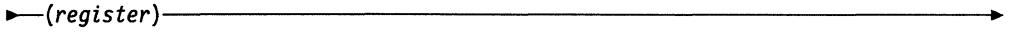
Parameters



Function Specifies the interrupt level to be unmasked.

Format Absolute notation. If more than one level is to be unmasked, the list of levels is enclosed in parentheses.

Default The supervisor assumes that the mask has been preloaded into a specified register.

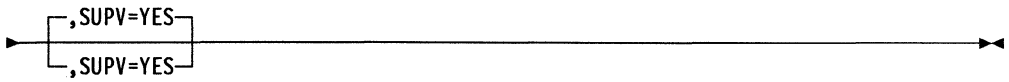


Function Specifies a register to contain the mask to enable the desired interrupt level.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



Function Specifies that the issuer is running in an interrupt level.

Format YES.

Default YES.

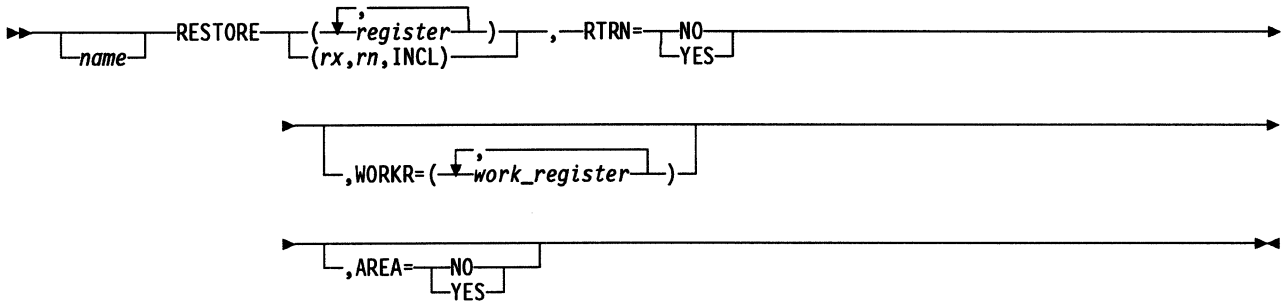
RESTORE—Restore Registers

The RESTORE macro does the following:

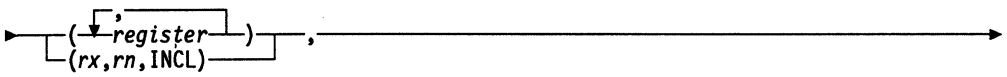
- Updates register 6 to point to the previous save area in the save area chain.
- Restores the specified registers from the save area addressed by register 6.
- Loads the instruction address register (IAR) with the address of the return point specified in the save area addressed by register 6.

Register 0 is not allowed for register parameters.

Syntax



Parameters



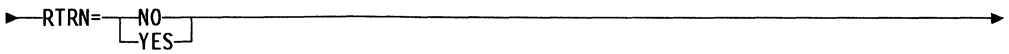
Function Specifies the registers to be restored. (*register*,...) specifies the individual registers to be restored. (*rx*,*ry*,INCL) specifies a range of registers to be restored; INCL must be included.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7.

Default No registers are restored.

Remarks If you specify a register with the (*register*,...) format and you specify the same register for WORKR, the contents of the register will not be restored.

If you specify (*rx*,*ry*,INCL), the WORKR keyword is not valid.

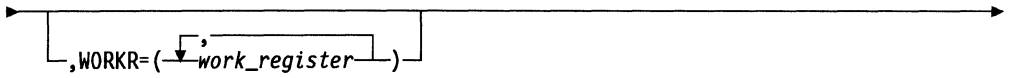


Function Specifies whether the instruction address register (IAR) is to be loaded. RTRN=YES indicates that the IAR is to be loaded with the address of a return point specified in the save area addressed by register 6. RTRN=NO indicates that the IAR is not to be loaded.

Format YES or NO.

Default None.

Remarks You must code this keyword.

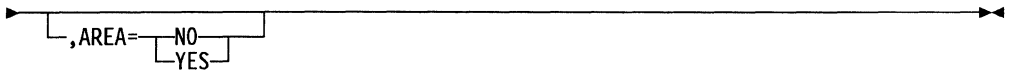


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format You can use registers 1, 2, 3, 4, 5, and 7. You cannot use equated values.

Default No work registers. All registers specified in the first parameter are restored.

Remarks If you specify a register in the first parameter with the (*register*,...) format, and you specify the same register for WORKR, the contents of the register will not be restored. If you specify (*rx*,*ry*,INCL) for the first parameter, WORKR is not valid.



Function Specifies whether register 6 is to be updated. AREA=YES indicates that before the registers are restored, register 6 is to be updated to point to the previous save area in the chain. AREA=NO indicates that register 6 is not to be updated.

Format YES or NO.

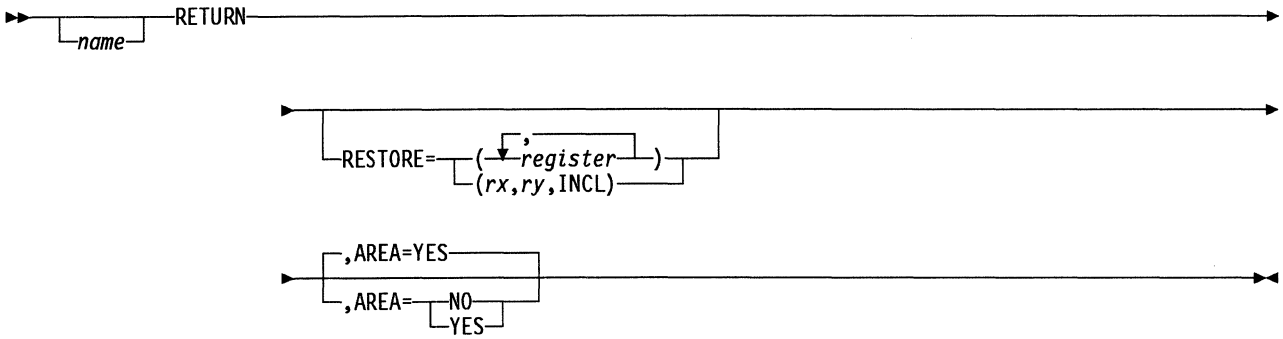
Default If you code a SAVE macro with AREA=YES and do not code a SAVE or RESTORE macro between that SAVE and this RESTORE, the default is YES; otherwise, the default is NO.

RETURN—Restore Registers and Return Control after a ROUTINE Macro

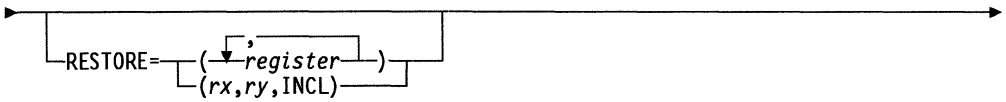
The RETURN macro restores the registers saved by the ROUTINE macro and returns control to the instruction following the PERFORM macro in the PERFORM, ROUTINE, RETURN macro sequence. Register 6 is updated to point to the previous save area. If no other ROUTINE macro is executed between a ROUTINE macro and its RETURN, you can code more than one RETURN for a single ROUTINE macro in the assembly of the routine. The RETURN macro must be executed following a ROUTINE macro and must follow its associated ROUTINE macro in the assembly of the routine.

The values saved by the ROUTINE macro and used by RETURN can be overridden by the RESTORE macro.

Syntax



Parameters



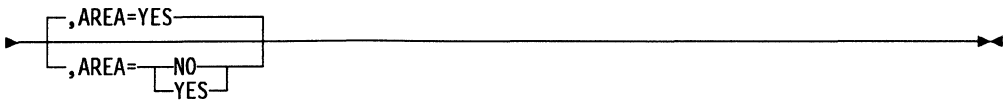
Function Specifies the registers to be restored. (*register*,...) specifies the individual registers to be restored. (*rx*,*ry*,INCL) specifies a range of registers to be restored.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7.

Default No registers are restored.

Remarks The value of the SAVE keyword on the ROUTINE macro is passed to the RETURN macro. Any value specified in the RESTORE keyword overrides the value passed from the ROUTINE macro.

Register 0 is not allowed.



Function Specifies whether register 6 is to be updated. AREA=YES indicates that register 6 is to be updated to point to the previous save area in the chain before the registers are restored. AREA=NO indicates that the registers are to be restored from the save area currently pointed to by register 6.

Format YES or NO.

Default YES.

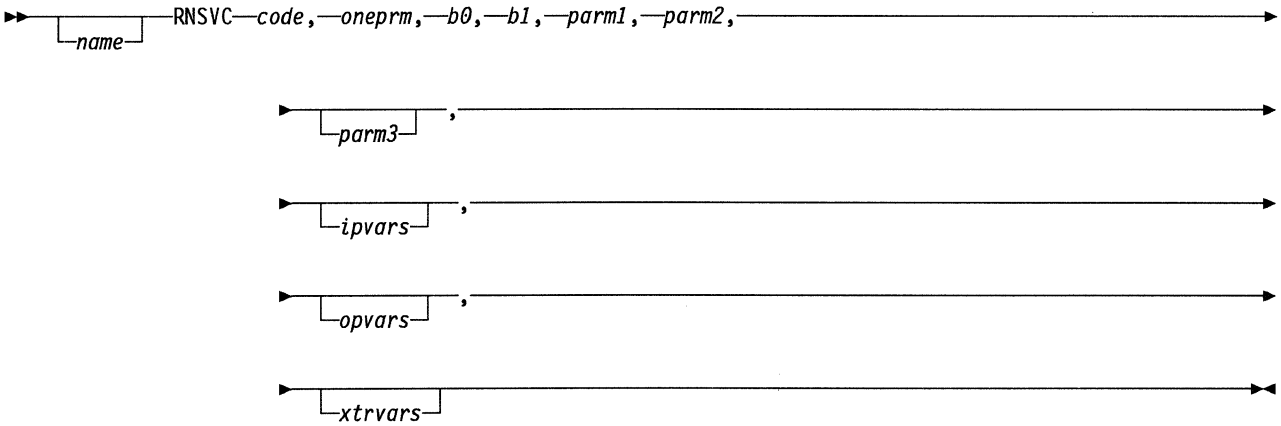
Remarks The value of the AREA keyword on the ROUTINE macro is passed to the RETURN macro. Any value specified in the AREA keyword overrides the value passed from the ROUTINE macro.

Use AREA=NO only in a routine that performs no other routines.

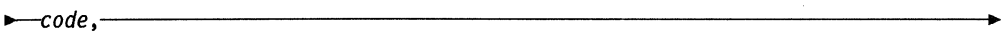
RNSVC—Generate Linkage from Level 5 to the Supervisor SVC-Handling Routine

The RNSVC macro, within a supervisor call (SVC) macro, generates the standard linkage from level 5 to the supervisor SVC-handling routine. See Chapter 1 for a list of the SVC codes and associated macros and for a description of the linkage between levels 5 and 4.

Syntax



Parameters

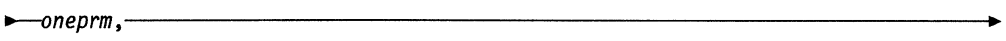


Function Specifies the SVC code.

Format Numeric value.

Default None.

Remarks This value is put in byte 2, bits 0 to 6,

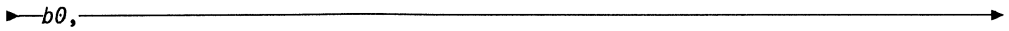


Function Specifies the value of the one-parm flag (byte 2, bit 7) in the SVC linkage.

Format 0 or 1.

Default None.

Remarks If this value is 1, byte 3 contains one input parameter in bits 2 to 4. The rest of the bits in byte 3 are then available for use as communication bits.



Function Specifies the value of the three-parms flag (byte 3, bit 0) of the SVC linkage.

Format 0 or 1.

Default None.

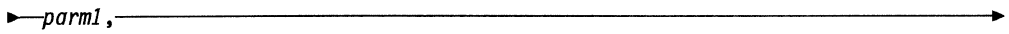
Remarks This bit should be 1 if there are more than one input parameter and more than one communication bit. If this bit is 1, all communication bits are in byte 5.

Function Specifies the value of the communication bit (byte 3, bit 1) of the SVC linkage.

Format 0 or 1.

Default None.

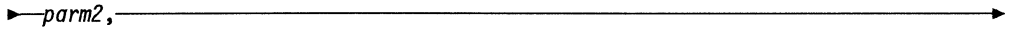
Remarks This bit is the only bit available in byte 3 for communication bits if the one-parm flag (byte 2 bit 7) is off.



Function Specifies the value of parameter 1 (byte 3, bits 2 to 4).

Format Numeric bit value.

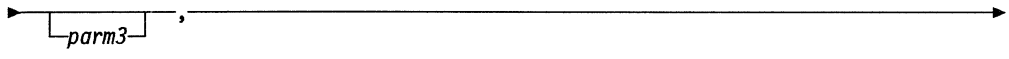
Default None.



Function Specifies the value of parameter 2 (byte 3, bits 5 to 7).

Format Numeric bit value.

Default None.

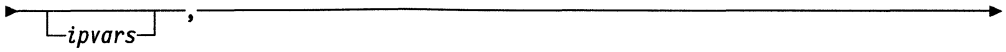


Function Specifies the value of parameter 3 (bytes 4 and 5).

Format Numeric bit value.

Default None.

Remarks If the pseudoparm 3 flag is on (byte 5, bit 0), byte 5 contains only communication bits (that is, does not contain a parameter 3).

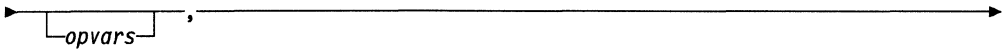


Function Specifies the first two inline input variables.

Format Variables separated by commas and enclosed in parentheses.

Default None.

Remarks For each variable specified, a 4-byte address constant is generated.

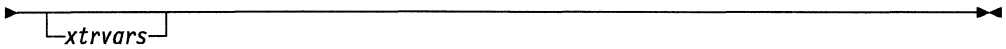


Function Specifies the inline output variables.

Format Variables separated by commas and enclosed in parentheses.

Default None.

Remarks For each variable specified, a fullword DC and RESTORE is generated.



Function Specifies the inline parameters to be placed after byte 5 of the communication bytes.

Format Variables separated by commas and enclosed in parentheses.

Default None.

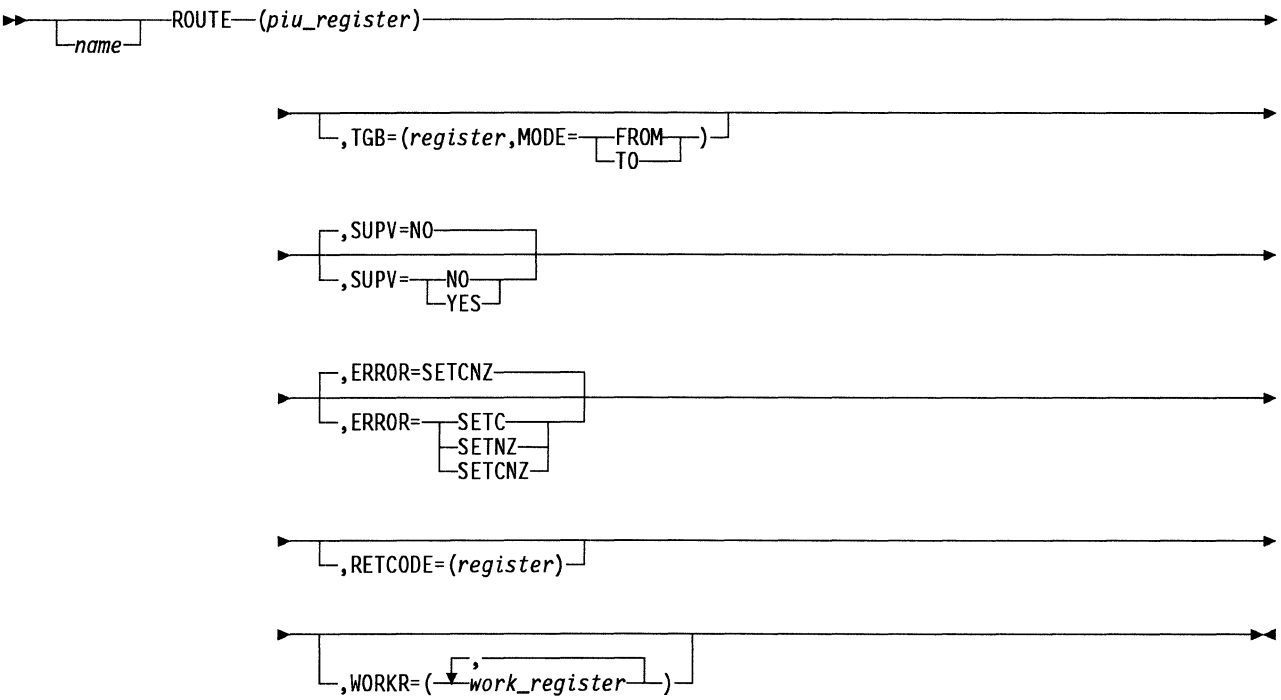
Remarks For each variable specified, a 4-byte address constant is generated.

ROUTE—Pass a PIU to the Explicit Route Control Component of Subarea Path Control

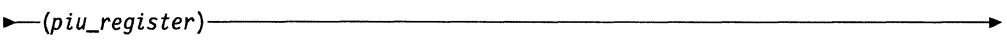
The ROUTE macro passes a path information unit (PIU) to the explicit route control component of subarea path control (subarea node component). Destination routing is performed on the PIU on the basis of the information contained in its transmission header.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register that points to the PIU to be routed.

Format Absolute register notation.

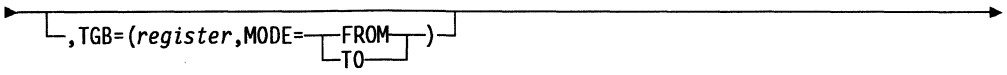
Default None.

Remarks If SUPV=YES, neither register 1 nor register 6 is allowed.

Register 3 is preferred.

The register cannot be the same as that specified on the transmission group control block (TGB) or RETCODE keyword.

On completion of the ROUTE function, the content of the register that specifies the PIU to be routed is set to 0.



Function When MODE=TO, the ROUTE macro expects the register to contain a pointer to the TGB that the PIU is to be sent over. When MODE=FROM, the ROUTE macro expects the register to contain a pointer to the TGB that the PIU arrived over.

Format (*register*,MODE=TO) or (*register*,MODE=FROM).

Default None.

Remarks If SUPV=YES, neither register 1 nor register 6 is allowed.

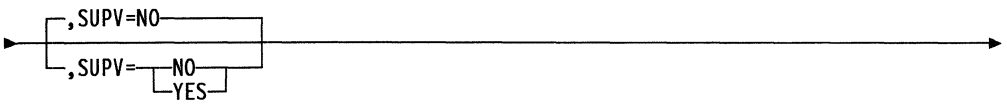
Register 2 is preferred.

Code MODE=TO when the subarea node already knows the transmission group that is to be used to route the PIU to its destination.

Code MODE=FROM when the PIU destination is not known and the ROUTE macro must determine the destination based on the destination routing information in the transmission header of the PIU.

By coding MODE=TO, the ROUTE macro infers that the PIU originated within NCP or within its boundary; thus, a null (0) TGB address is implied for the FROM keyword.

The register cannot be the same as that specified on the PIU register keyword or the RETCODE keyword.

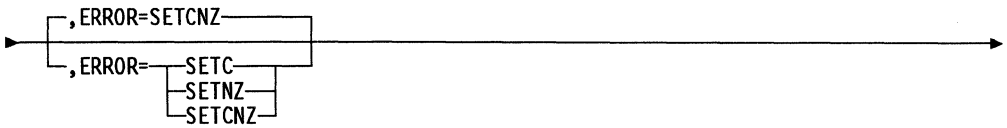


Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

Remarks If you code SUPV=YES, you must provide the address of an unused save area in register 6.



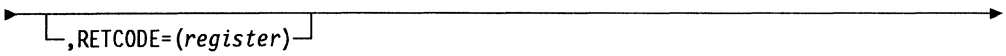
Function Specifies whether the C latch, the Z latch, or both are to indicate an error.

Format SETC, SETNZ, or SETCNZ.

Default SETCNZ.

Remarks Regardless of the keyword specified, the C latch is set to 1 if the PIU could not be routed; otherwise it is set to 0. The Z latch is set to 0 if the PIU could not be routed; otherwise it is set to 1.

If the PIU cannot be routed, it is either released to the free-buffer pool or is placed on the system exception queue.



Function Specifies the register to get the return code after execution of the ROUTE function.

Format Register notation.

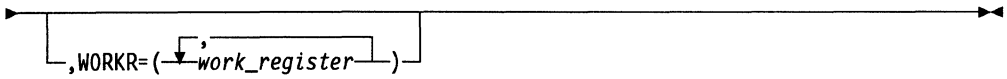
Default Register 1.

Remarks If the PIU could not be routed, the return code value is 1; otherwise, the return code value is 0.

If the PIU cannot be routed, it is either released to the free-buffer pool or placed on the system exception queue.

This keyword is valid only for SUPV=YES.

The register cannot be the same as that specified on the PIU register keyword or the TGB keyword.



Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying work registers makes execution of the macro more efficient.

Format Register notation.

Default No work registers.

Remarks This keyword is valid only for SUPV=YES.

You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

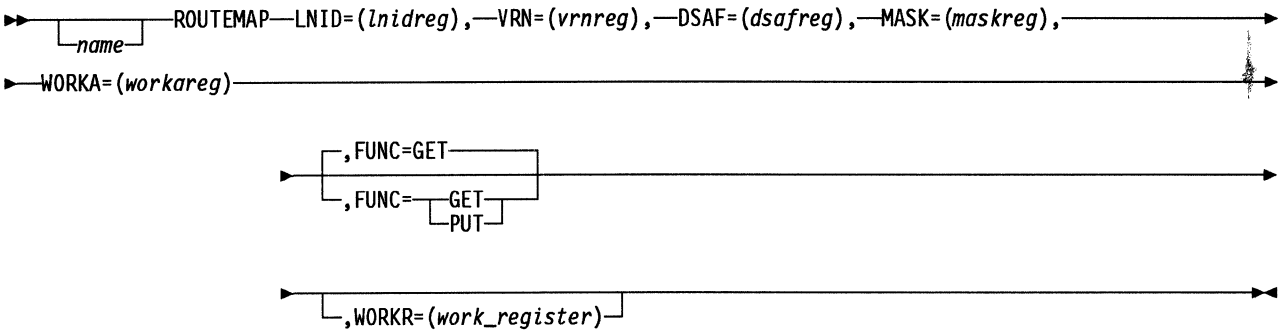
ROUTEMAP—Access the Explicit Route to Virtual Route Mapping List

The ROUTEMAP macro produces inline code that accesses the explicit route to virtual route mapping list (EML). The local network identification (LNID) and destination subarea address field (DSAF) are used to find the correct row of the EML. The EML row is indexed with the virtual route number to determine the correct entry. Each EML entry is an explicit route mask that indicates the explicit route on which the virtual route number will flow. Either the entry is retrieved and put into the MASK variable, or the value from the MASK variable is stored into the entry, depending on the encoding of the FUNC keyword.

For proper expansion of the ROUTEMAP macro, the NVT structure (XCXTNVT), the RMB structure (XCXTRMB), and the system equates (XCXTEQU) must be made available by the invoking module. Also, the invoking module must provide an EXTRN for external label CXTNVT0.

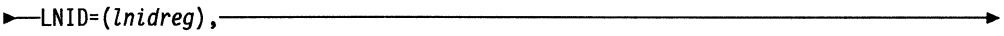
Register 0 is not allowed for register parameters.

Syntax



Note: No two keywords can specify the same register.

Parameters

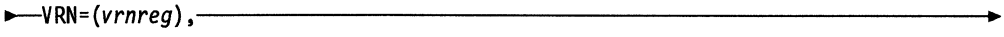


Function Specifies the register that contains the local network identification.

Format Byte register notation.

Default None.

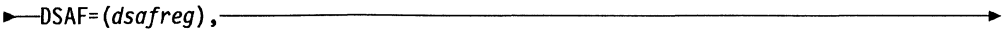
Remarks Register contents are destroyed.



Function Specifies a register containing the virtual route number that will be used in the virtual route to explicit route mapping.

Format Byte register notation.

Default None.

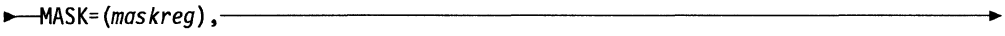


Function Specifies a register that contains the destination subarea address of the virtual route.

Format Register notation.

Default None.

Remarks The register contents are destroyed.

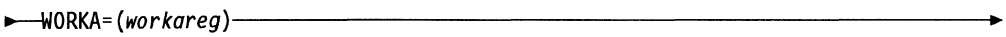


Function Specifies the register to be used to hold the explicit route mask.

Format Register notation.

Default None.

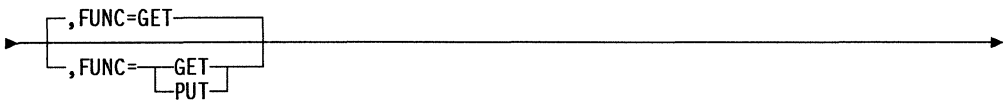
Remarks A bit turned on in the mask indicates the corresponding explicit route. Bit 0 on indicates ER 0, bit 7 on indicates ER 7, and so forth, through bit 15. The explicit route mask is always a halfword, even though the EML entries may be 1-byte masks. If the EML entries are 1-byte masks and FUNC=PUT, only the high byte (bits 0 to 7) of the halfword is stored into the EML entry. If the EML entries are 1-byte masks and FUNC=GET, the high byte of the halfword explicit route mask is set to the EML entry's value and the low byte is set to 0. The register specified must be an odd-numbered register.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

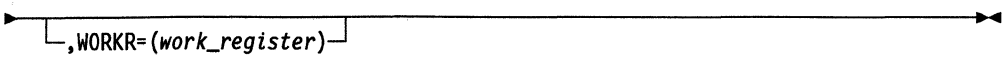


Function Specifies whether the macro is to retrieve the explicit route mask from the EML or store the explicit route mask in the EML.

Format Register notation.

Default GET.

Remarks FUNC=GET will set the Z-latch ON if the mask returned is 0, and will set the Z-latch OFF if the mask returned is not 0. The register specified must be an odd-numbered register.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks This keyword is required if FUNC=PUT. You can specify register 1, 3, 5, or 7. Do not use equated symbols.

ROUTINE—Define the Entry Point for a Routine

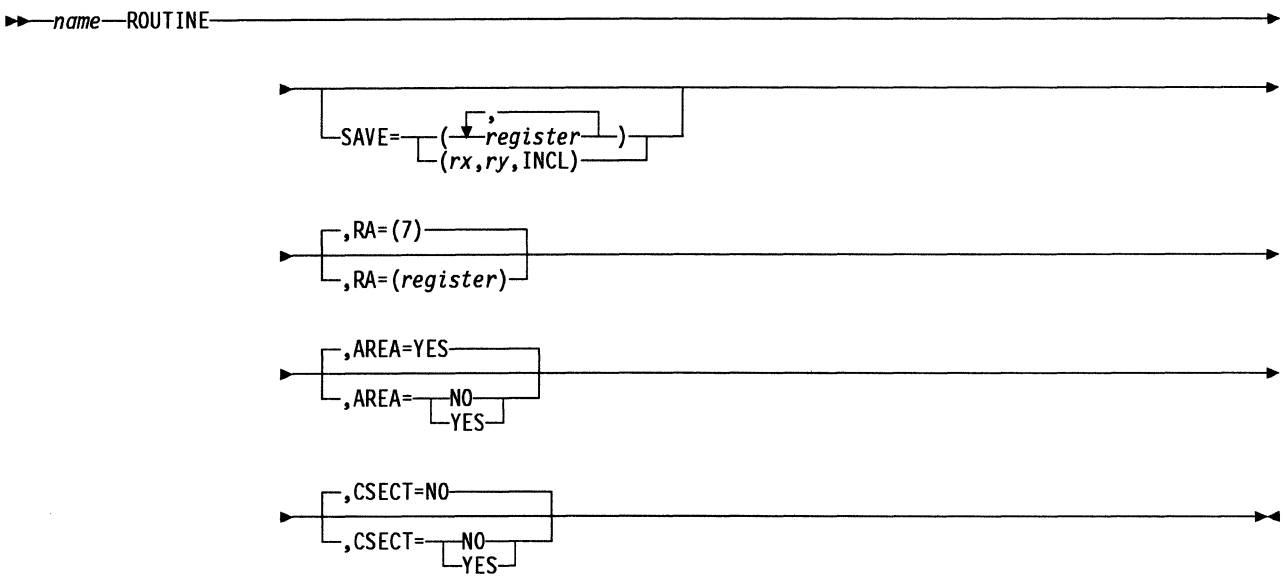
The ROUTINE macro defines the entry point for a routine in a structured process block created by a ROUTINE/RETURN pair. The routine is invoked by the PERFORM macro. Refer to "Structured Process Blocks" in *NCP and SSP Customization Guide* for more information.

Before the ROUTINE macro executes, register 6 must point to a save area in the save area chain. The ROUTINE macro saves all designated registers in the save area and can update register 6 to point to the next save area in the chain. When the routine called by the ROUTINE macro is finished, use the RETURN macro to restore the saved registers and update register 6 with the pointer to the previous save area.

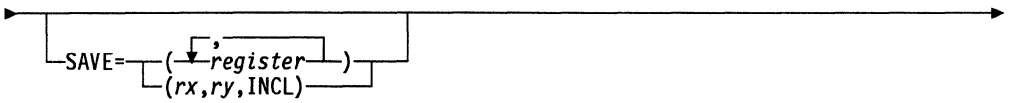
You can change the save area pointer in register 6 by using the RESTORE macro to point at a save area different from the one set by the ROUTINE macro.

Register 0 is not allowed for register parameters.

Syntax



Parameters

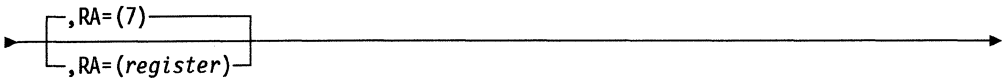


Function Specifies the registers to be saved. (*register*,...) specifies the individual registers to be saved. (*rx*,*ry*,INCL) specifies the limits of a range of registers to be saved.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7.

Default No registers are saved.

Remarks The value of the SAVE keyword on the ROUTINE macro is passed to the corresponding RETURN macro.



Function Specifies the register containing the return address. The register specified must match the linkage register used by every PERFORM associated with this ROUTINE macro. This return address is stored in the save area addressed by register 6 when the routine is entered. When the RETURN macro is executed, the return address in the save area is used to return control to the instruction following the completed PERFORM macro.

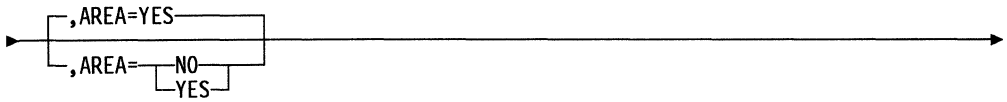
Format Register notation.

Default Register 7.

Remarks If the return address register is also coded in the SAVE keyword, the register contents will be duplicated in the save area field for that register.

The return address register is generally the best choice for a return code or a returned parameter.

The ROUTINE and PERFORM macros cannot detect that ROUTINE was invoked by a PERFORM macro that used a different linkage register. Incorrect operation will occur during execution if care is not taken when the network control program is assembled.



Function Specifies whether register 6 is to be updated. AREA=YES indicates that register 6 is to be updated to point to the next save area in the chain after the registers are saved. AREA=NO indicates that register 6 is to remain pointing to the current save area after the registers are saved.

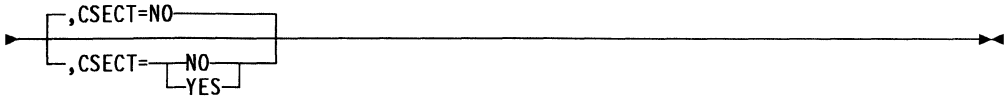
Format YES or NO.

Default YES.

Remarks The value of the AREA keyword on the ROUTINE macro is passed to the corresponding RETURN macro.

NO is for use only in the lowest-level routine that performs no other routines.

SAVE and RESTORE macros can be used within a routine if each AREA=YES keyword on a SAVE macro is balanced by an AREA=YES keyword coded on the corresponding RESTORE macro.



Function Specifies whether a CSECT statement is to be generated. CSECT=NO indicates that a CSECT statement is not generated. CSECT=YES indicates that a CSECT statement, using the name field, is generated.

Format NO or YES.

Default NO.

RSLVCAP—Get SSCP-NCP Session Capability Data

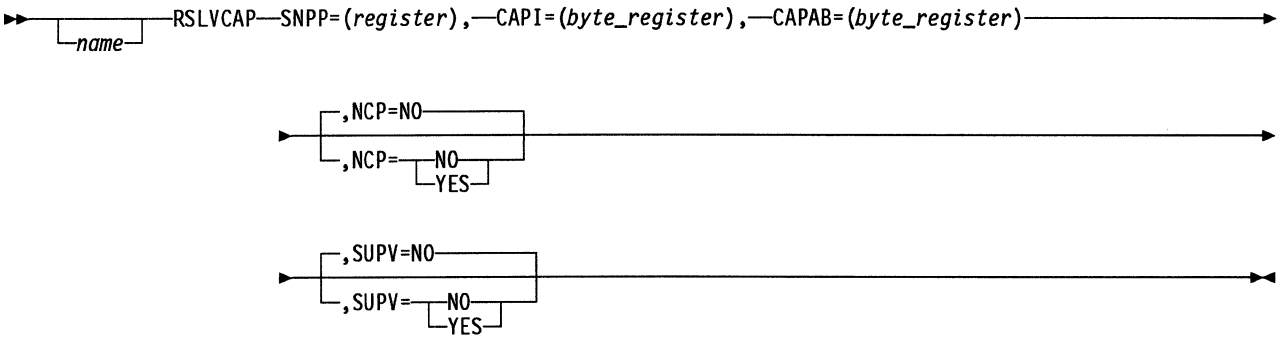
The RSLVCAP macro, given a pointer to an SSCP-NCP session control block (SNP), returns 1 byte of the SNP's capability field. You indicate which of the 4 capability bytes you want.

RSLVCAP returns the capability byte in the byte register specified, as follows:

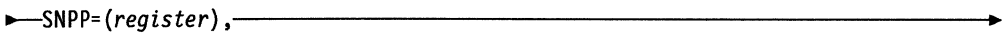
CAPI Value	SNP Capability Byte
X'01'	SNPSCAP1
X'02'	SNPSCAP2
X'03'	SNPSCAP3
X'04'	SNPSCAP4

Register 6 must contain a valid save area pointer. Register 0 is not allowed.

Syntax



Parameters

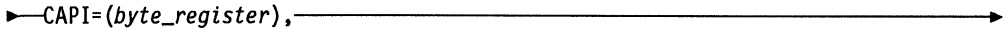


Function Specifies a register containing the pointer to the SSCP-NCP SNP.

Format Register notation.

Default None.

Remarks Register 6 is not allowed.



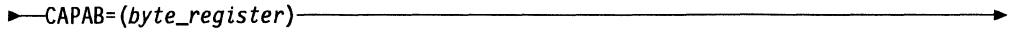
Function Specifies a byte register containing the number of the capability byte you want.

Format Byte register notation.

Default None.

Remarks The valid range for CAPI is 0 to 04.

The register specified cannot be the same as the register specified for SNPP.



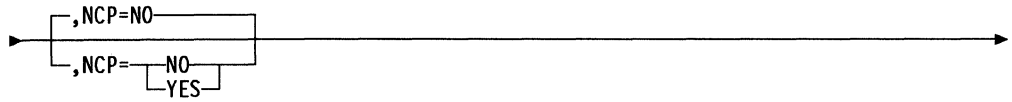
Function Specifies a byte register that contains the appropriate byte of the SNP's capabilities.

Format Byte register notation.

Default None.

Remarks The register specified may be the same as CAPI.

The register may be the same as SNPP but the SNP address will be lost.

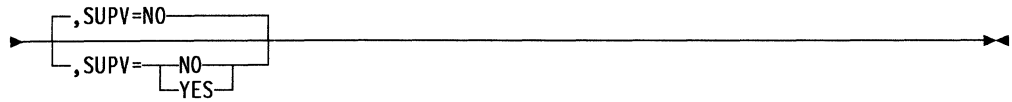


Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES specifies that inline linkage code is generated. NCP=NO specifies that an SCV is generated.

Format YES or NO.

Default NO.

Remarks Specify this parameter only when SUPV=NO.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

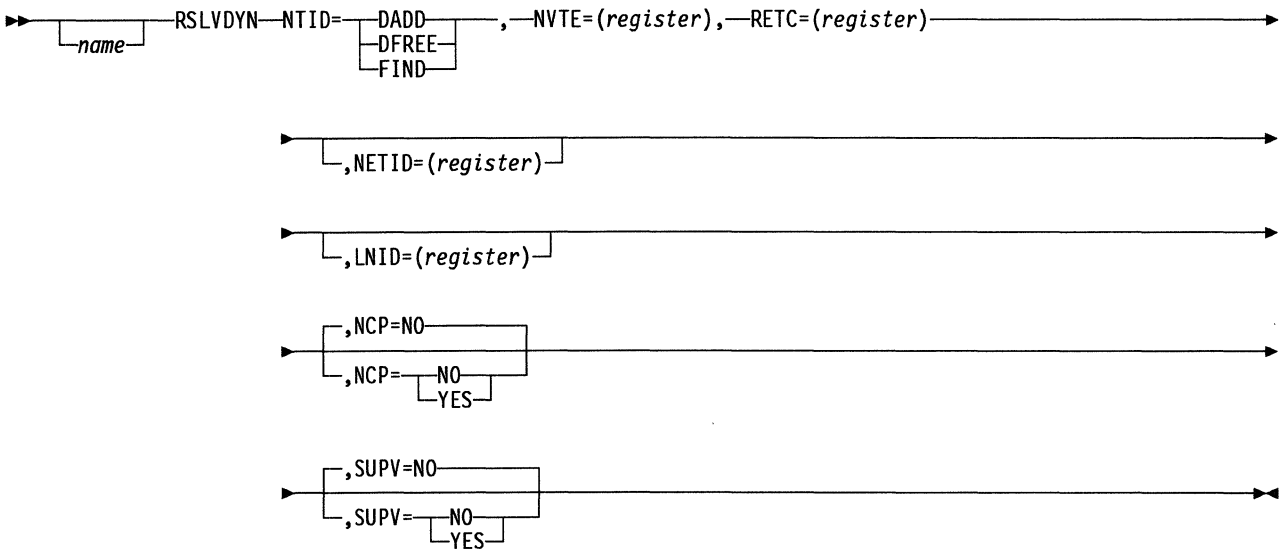
RSLVDYN—Allocate or Deallocate a Network from the NVT

The RSLVDYN macro allocates or deallocates a network from the network vector table (NVT). The desired network is identified by its network ID or local network identification (LNID).

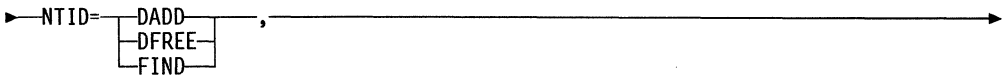
Note the following considerations:

- If NTID=FIND, SUPV=NO, and NCP=NO are coded, register 1 is not valid.
- If NTID=DFREE is coded, DSECT XCXTNVT is necessary for the proper expansion of the RSLVDYN macro.
- Register 6 must point to a save area.
- A different register must be used for each keyword.
- Register 0 is not allowed.

Syntax



Parameters



Function Identifies which NETID operation is to be performed.

NTID=FIND specifies that a NETID is to be found from the NVT. This keyword will be used only when NETID=(register) is specified on the RSLVDYN macro.

Note: This function should only be used when finding the network for a station during XID2 negotiation time for leased subarea stations or during XID2 pre-negotiation time for switched subarea stations. This function increments the count of users for the found network.

NTID=DADD specifies that an available dynamic entry is to be allocated in the network vector table for the input NETID. This can only be used when NETID=(*register*) is specified on the RSLVDYN macro.

Note: This function should be used when allocating the network for a station during XID2 negotiation time for leased subarea stations or during XID2 pre-negotiation time for switched subarea stations. This function sets the count of users for the found network to 1.

NTID=DFREE specifies that dynamic NETID entries from the NVT are to be freed when they are no longer needed. This should be used only when LNID=(*register*) is specified on the RSLVDYN macro.

Note: This function should be used when deallocating the network for a station. This function decrements the count of users for the found network. The entry is made available for reuse only if the count of users for the found network reaches 0.

Format FIND, DADD, or DFREE.

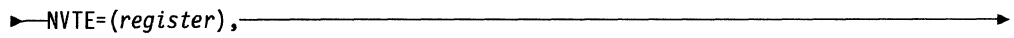
Default None.

Remarks Only one NTID type can be specified at a time.

You must code NVTE. It will be used to return a pointer to the found, allocated, or deallocated entry.

You must code RETC. It will be used to indicate if the entry was found, allocated, or deallocated.

If you code NTID=FIND, SUPV=NO, and NCP=NO, register 1 is not valid.

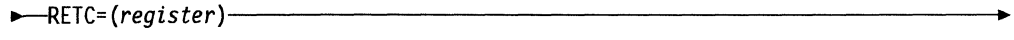


Function Specifies a register containing the pointer to the NVT entry.

Format Absolute register notation.

Default None.

Remarks A full register is required. You must specify an odd-numbered register if NTID=DFREE.



Function Specifies a register that indicates:

X'00' An in-use network ID that matched the input network ID was found in the NVT. This status is generated when NTID=FIND or NTID=DFREE.

A new dynamic entry was allocated for the input network ID. This output status is generated when NTID=DADD.

X'01' No in-use network ID that matched the input network ID was found in the NVT. This status is generated when NTID=FIND.

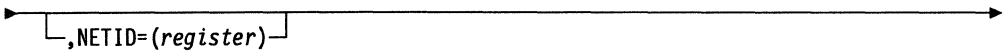
X'02' An in-use dynamic network ID that matched the input network ID was found in the NVT but was no longer usable because the number of stations using that dynamic network ID had already reached the maximum of 255. This status is generated when NTID=FIND.

X'03' No new dynamic entry was available in the NVT. (All 255 are in use.) This status is generated when NTID=DADD.

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification 0 or 1 is required.



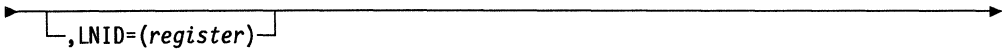
Function Specifies a register containing a pointer to the applicable NETID.

Format Absolute register notation.

Default None.

Remarks A full register is required.

If you code NTID=DFREE, NETID is not valid.



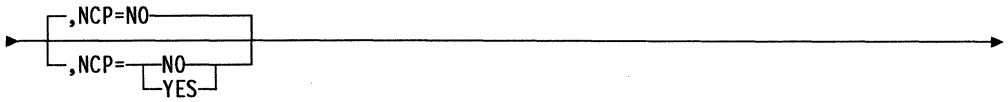
Function Specifies a register containing the local network identification.

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification 0 or 1 is required.

If you code NTID=FIND or NTID=DADD, LNID is not valid.



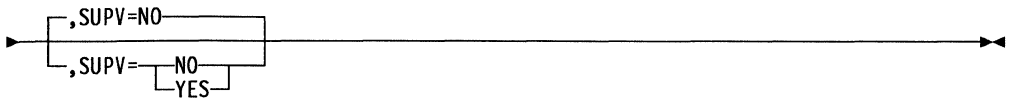
Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES indicates that the macros are to expanded inline. NCP=NO indicates that an SVC call is to be used.

Format YES or NO.

Default NO.

Remarks If NCP=NO, Register 1 for NETID is not valid.

Specify this parameter only when SUPV=NO is coded.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in interrupt level.

Format YES or NO.

Default NO.

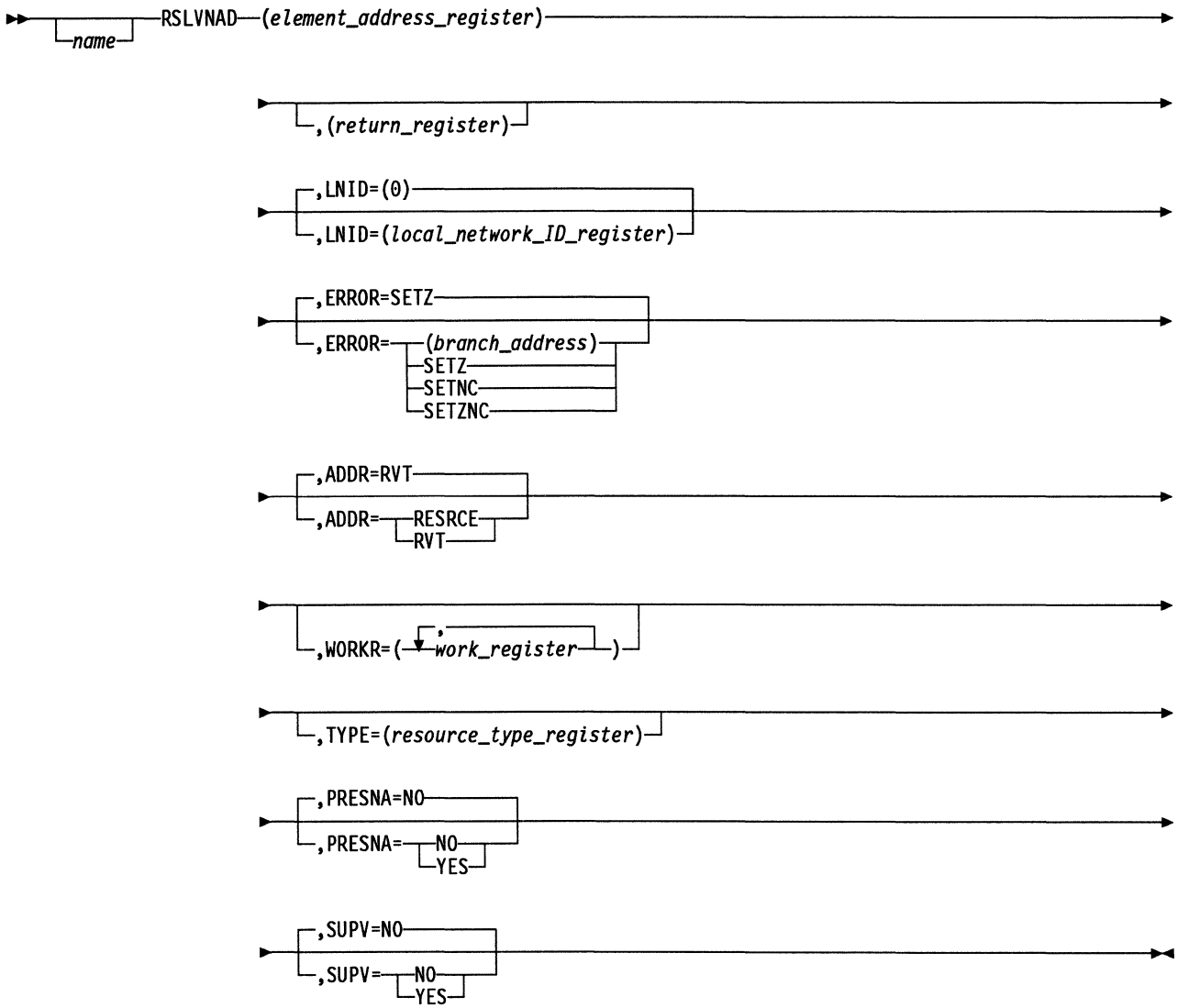
Remarks If SUPV=YES, you cannot specify register 1 for NETID.

RSLVNAD—Locate a Network Element Resource Control Block

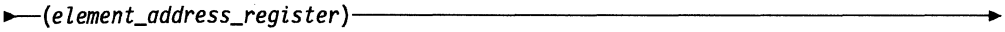
The RSLVNAD macro transforms a given element address either into the address of the resource control block for that element address or into the address of the resource vector table (RVT) entry for that address. RSLVNAD can also return the 2-byte type field from the RVT if you code the TYPE keyword. If NCP is a gateway NCP, it may be necessary to specify the network in which the given address is valid by supplying the local network identification (LNID). If you do not specify LNID, the native network (LNID=0) is assumed.

Register 0 is not allowed for register parameters.

Syntax



Parameters



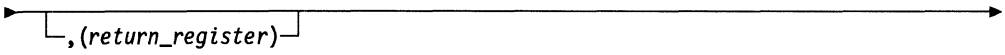
Function Specifies the register containing the element address to be resolved.

Format Register notation.

Default None.

Remarks If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed.

Register 2 is standard.

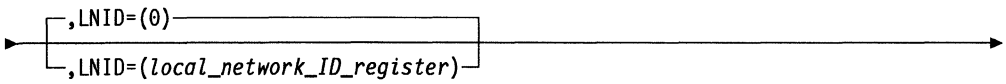


Function Specifies the register in which the address is to be returned.

Format Register notation.

Default The element address register is also used as the return register.

Remarks If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed. Register 3 is standard.



Function Specifies the register containing the local network identification of the network in which the specified element address is valid.

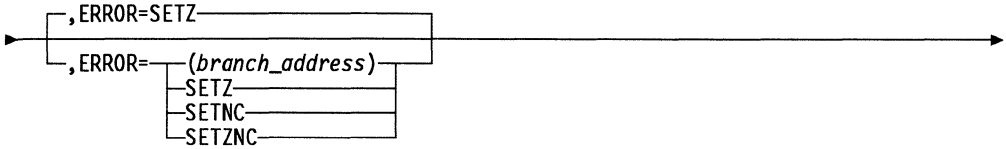
Format Register notation.

Default The native network (LNID=0) is assumed.

Remarks Although the LNID keyword is a 1-byte value, you must specify a full register with the LNID in the low-order byte. The contents of the high-order byte are not read.

The contents of the complete LNID register are preserved unless the register is used as an output register.

If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed. Register 4 is standard.



Function Either specifies the address to be given control or specifies whether the C latch, Z latch, or both are to indicate whether the supplied element address is valid.

Format Label notation, register notation, SETZNC, SETZ, or SETNC.

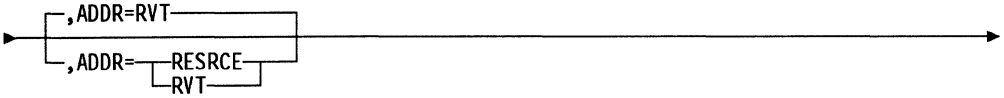
Default SETZ.

Remarks For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the element address is not valid or to 0 if it is valid. For SETZNC or SETNC, the C latch is set to 0 if the element address is not valid or to 1 if it is valid.

SETZ gives the most efficient macro expansion.

If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed.

Do not list this register as a work register in the WORKR keyword.



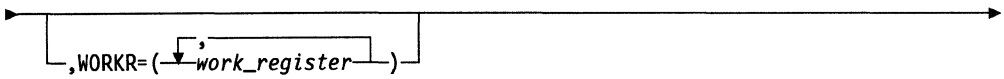
Function Specifies whether the address of the resource vector table entry or the address of the resource control block is to be returned (SUPV=NO only). If you specify ADDR=RESRCE, this macro returns the address of the resource control block (DVB, LCB, SCB, or LKB) pointed to by that RVT entry.

If you specify ADDR=RVT, this macro returns the address of the RVT entry.

Format RVT or RESRCE.

Default RVT.

Remarks If SUPV=YES, ADDR=RESRCE is not allowed.

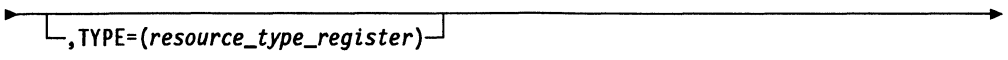


Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying such registers makes execution of the macro more efficient.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default No work registers.

Remarks If SUPV=NO, this keyword is ignored. Do not use the register specified on the ERROR keyword as a work register.



Function Specifies the register to return the resource type or the RVT flags field.

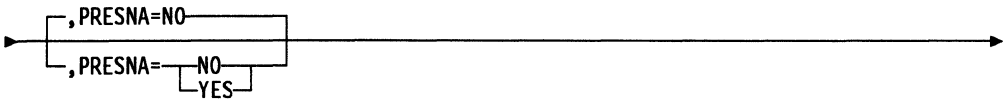
Format Register notation.

Default Type not returned.

Remarks If SUPV=NO, register 1 is standard.

If SUPV=YES, this keyword is not allowed.

Register 1 is modified when this keyword is specified.

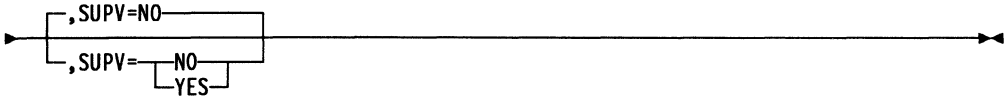


Function For local subarea addresses, specifies whether a BSC or start-stop resource is valid. For PRESNA=YES, no checking is performed. For PRESNA=NO, a check is done to ensure that the network address is not a BSC or start-stop network address.

Format YES or NO.

Default NO.

Remarks Checking is done to ensure that the network address is within the RVT, and that the network address refers to an SNA entry.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

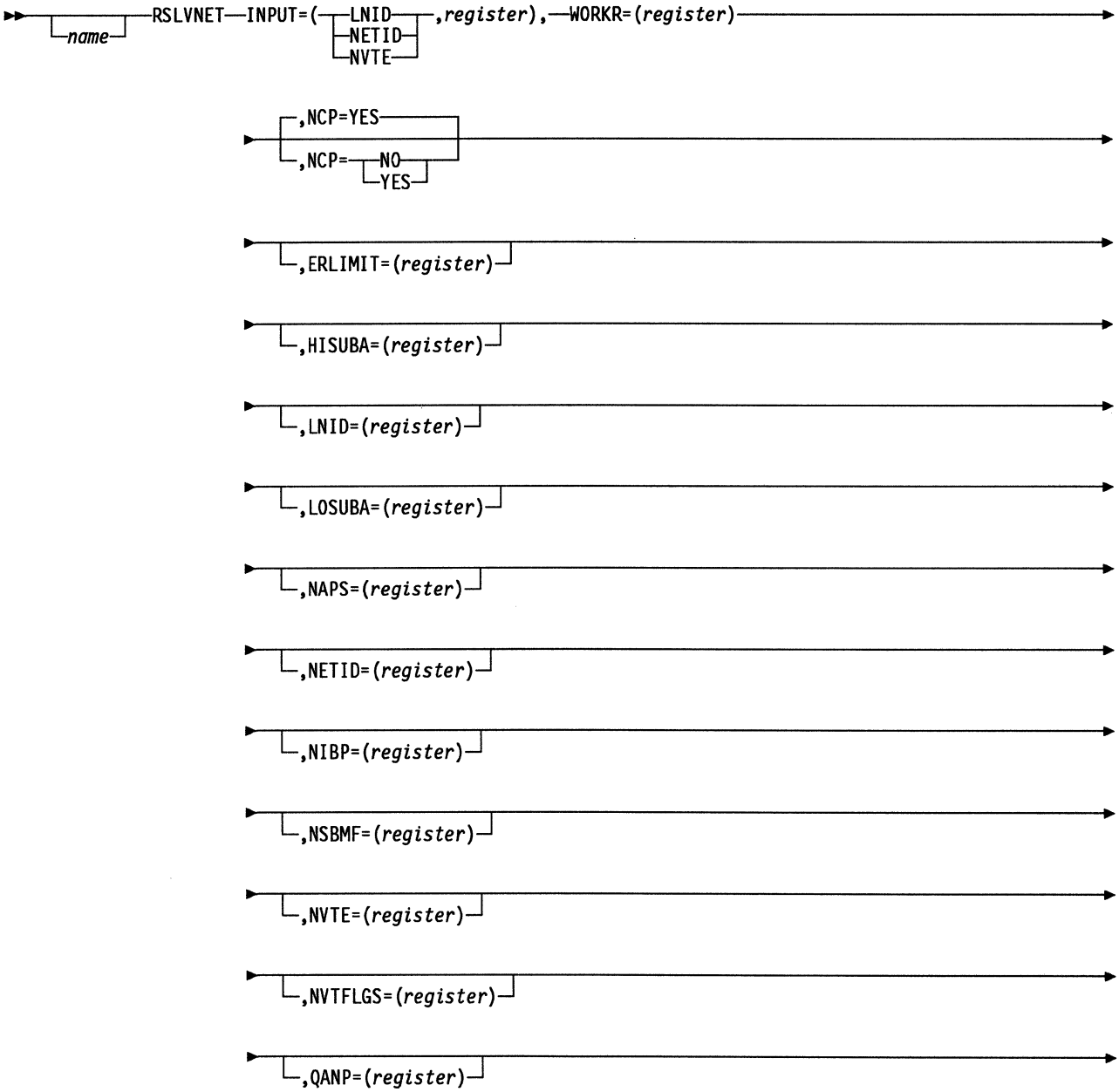
Default NO.

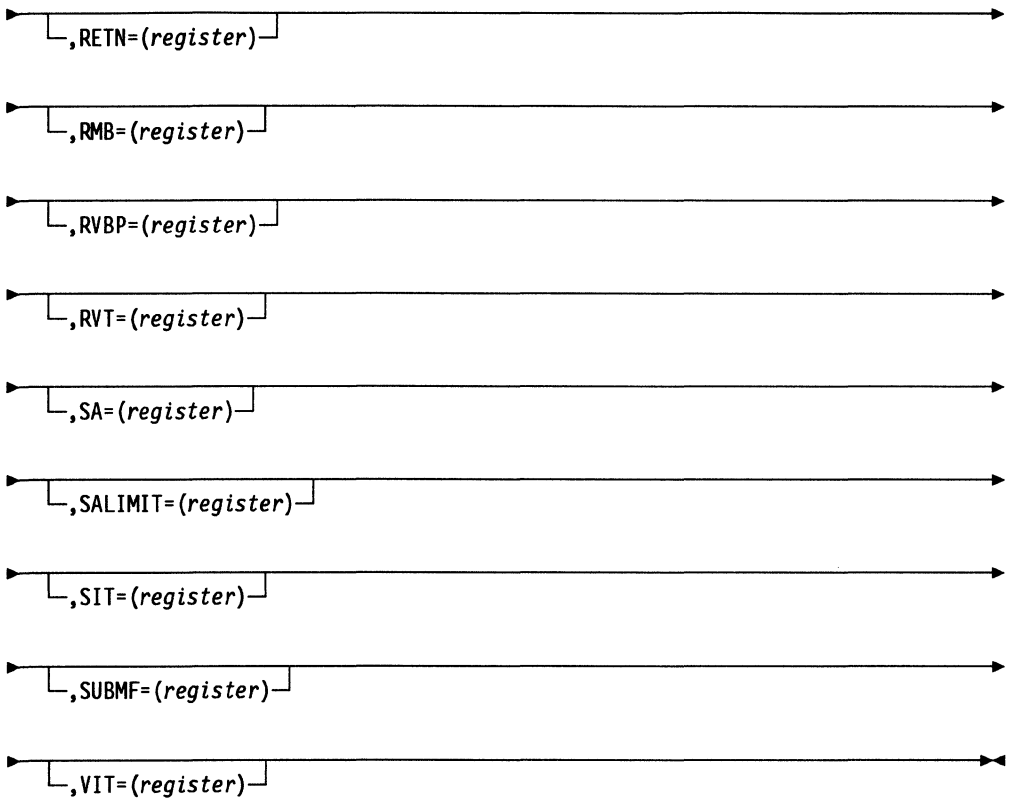
RSLVNET—Get Network Vector Table Data

The RSLVNET macro furnishes information about a given network attached to a gateway NCP. The desired network is identified by its network ID, its local network identification (LNID), or a pointer to its entry in the network vector table (NVT). The XCXTNVT DSECT is necessary for the proper expansion of the RSLVNET macro. Also, if the input is NETID, register 6 must point to a usable save area and the issuing routine must issue an EXTRN for CXDNETID before issuing the macro. If the input is LNID, the issuing routine must issue an EXTRN for CXTNVT0.

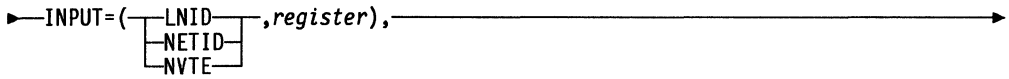
Register 0 is not allowed for register parameters.

Syntax





Parameters



Function Identifies the network about which information is to be supplied.

Format (Keyword,*register*), where *register* is specified in absolute register notation.

Default None.

Remarks Only one input can be specified.

If INPUT=LNID, the specified register contains the local network identification.

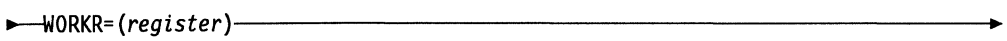
If INPUT=NETID or NVTE, the specified register contains a pointer to the network ID or NVT entry.

You must specify a full register if INPUT=NETID or NVTE and 1 byte of an odd-numbered register if INPUT=LNID.

Except as discussed in the next paragraph, you can also specify an input register as an output register.

When INPUT=LNID, the byte register containing the local network identification is altered. When INPUT=NETID or NVTE, the input register is not altered.

When INPUT=NETID or LNID, code RETN. RETN indicates whether or not the given network ID appears in the network vector table. If INPUT=NETID, the information requested through other output parameters is valid only if the network is in use; if INPUT=LNID, the information requested through other output parameters is valid even if the network is not in use.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Absolute register notation.

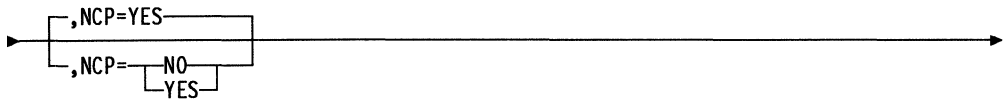
Default None.

Remarks An odd-numbered full register is required.

If a single output is requested and that output is to be supplied in a full register, the work register may be used as an output register.

If multiple outputs are requested, the work register cannot be used as an output register.

The input register cannot be used as a work register.



Function Specifies whether the macro is generated by user-written code or by NCP product code.

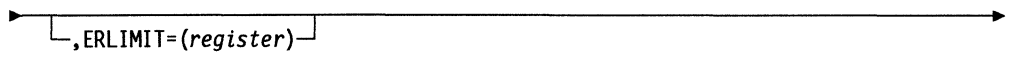
NCP=YES specifies that this macro was called by an NCP user and inline code is generated.

NCP=NO specifies that the call comes from a user-generated code and a SVC call is generated.

Format YES or NO.

Default YES.

Remarks When NCP=NO, you cannot specify register 1 for NETID.

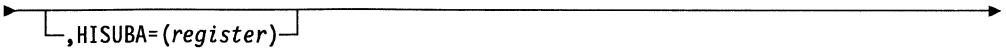


Function Specifies the register containing the highest allowable number of explicit routes supported (8 or 16) between subarea pairs in the specified network.

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification (0 or 1) is required.

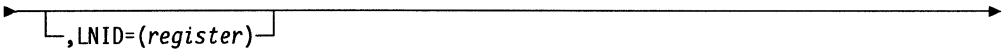


Function Specifies the register containing the higher 2 bytes of the subarea address of NCP in the specified network.

Format Absolute register notation.

Default None.

Remarks A full register is required.

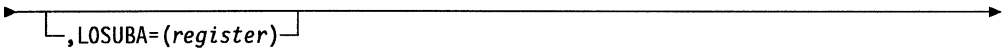


Function Specifies the register containing the local network identification.

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification (0 or 1) is required.



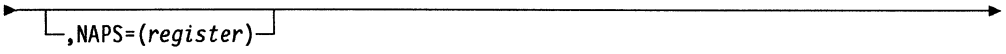
Function Specifies the register containing the lower 2 bytes of the subarea address of NCP in the specified network.

Format Absolute register notation.

Default None.

Remarks This keyword is equivalent to the SA keyword except that it allows more accurate coding of keywords for new occurrences of this macro. If both SA and LOSUBA are coded, the SA keyword is ignored and a warning message is given.

A full register is required.

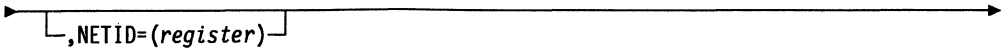


Function Specifies the register containing the 2-byte network address of the NCP physical services in the specified network.

Format Absolute register notation.

Default None.

Remarks A full register is required.

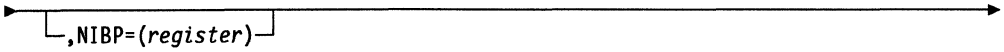


Function Specifies the register containing a pointer to the applicable network ID.

Format Absolute register notation.

Default None.

Remarks A full register is required.

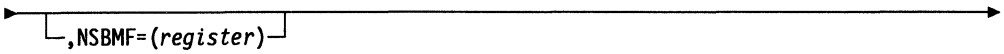


Function Specifies the register containing a pointer to the applicable network interconnect control block.

Format Absolute register notation.

Default None.

Remarks A full register is required.

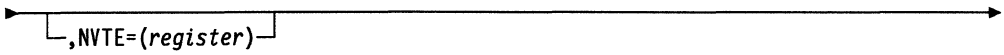


Function Specifies the register containing the applicable not-subarea mask (the complement of the subarea mask).

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification (0 or 1) is required.

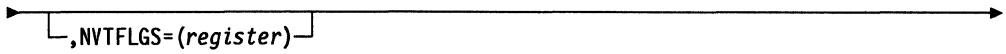


Function Specifies the register containing a pointer to the applicable network vector table entry.

Format Absolute register notation.

Default None.

Remarks A full register is required.

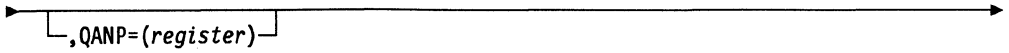


Function Specifies the register containing the NVT flag byte for the specified network.

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification (0 or 1) is required.

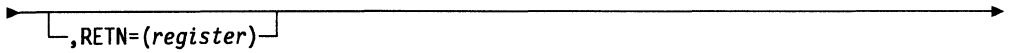


Function Specifies the register containing a pointer to the applicable queue anchor block for a network.

Format Absolute register notation.

Default None.

Remarks A full register is required.



Function If INPUT=NETID or LNID, specifies a register that indicates whether or not the given network ID appears in the network vector table. If the ID does appear, the specified byte register will contain X'00'; otherwise, the register will contain X'01'.

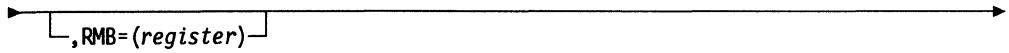
Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification (0 or 1) is required.

Specify RETN when INPUT=NETID or LNID.

The RETN output is valid with the NETID input and LNID input.

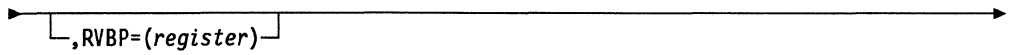


Function Specifies the register containing a pointer to the applicable route management block.

Format Absolute register notation.

Default None.

Remarks A full register is required.

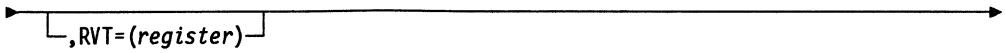


Function Specifies the register containing a pointer to the applicable resource vector control block.

Format Absolute register notation.

Default None.

Remarks A full register is required.

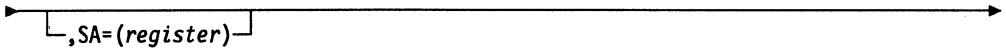


Function Specifies the register containing a pointer to the highest element network address field (not to the first entry) in the applicable resource vector table.

Format Absolute register notation.

Default None.

Remarks A full register is required.



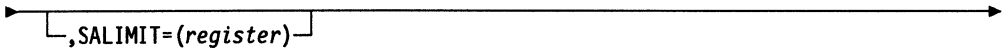
Function Specifies the register containing the lower 2 bytes of the subarea address of NCP in the specified network.

Format Absolute register notation.

Default None.

Remarks This keyword is equivalent to the LOSUBA keyword. For new occurrences of this macro, code LOSUBA in place of SA.

A full register is required.

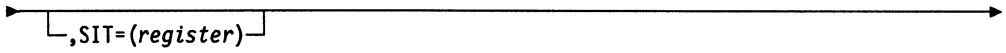


Function Specifies the register containing the highest valid subarea address in the specified network.

Format Register notation.

Default None.

Remarks A full register is required.

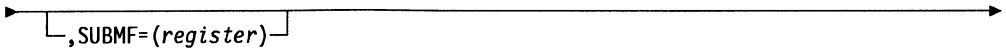


Function Specifies the register containing a pointer to the applicable subarea index table.

Format Absolute register notation.

Default None.

Remarks A full register is required.

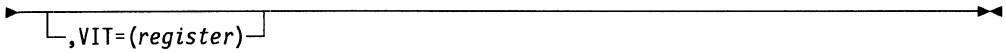


Function Specifies the register containing the applicable subarea mask.

Format Absolute register notation.

Default None.

Remarks An odd-numbered register with byte specification (0 or 1) is required.



Function Specifies the register containing a pointer to the applicable virtual route subarea index table.

Format Absolute register notation.

Default None.

Remarks A full register is required.

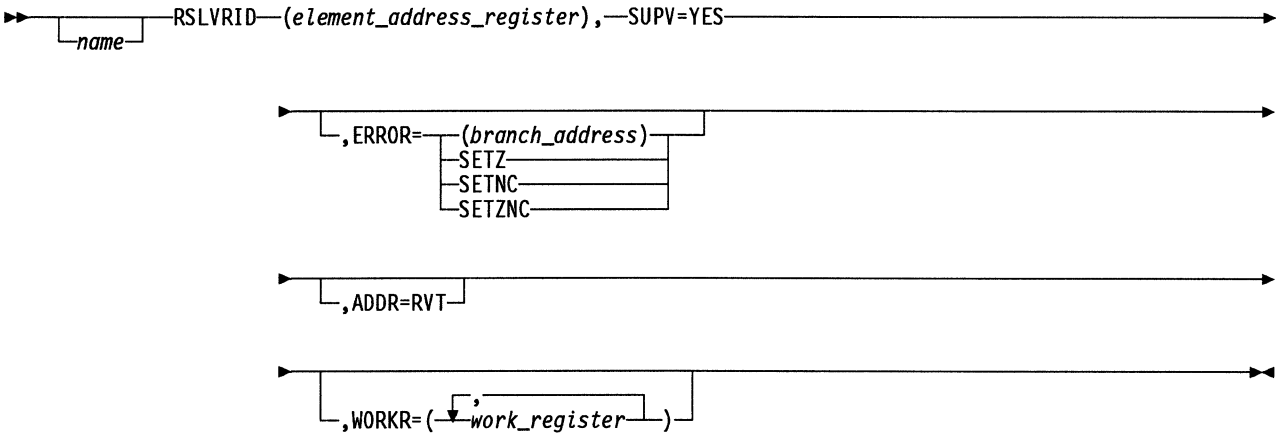
RSLVRID—Locate a BSC or Start-Stop Resource Control Block

The RSLVRID macro converts a BSC or start-stop element address into either the address of the resource control block or the address of the resource vector table (RVT) entry corresponding to that element address.

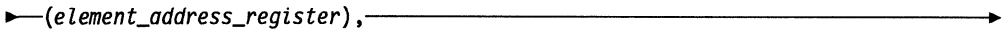
Register 0 is not allowed for register parameters.

SUPV=YES Format (Generates Inline Code)

Syntax



Parameters

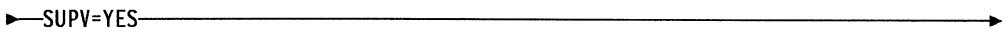


Function Specifies the register containing the element address to be resolved.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed.



Function Specifies that the issuer is running in an interrupt level.

Format YES.

Default NO.

Function Specifies either the address to be given control, or whether the C latch, Z latch, or both are to indicate whether the supplied element address is valid.

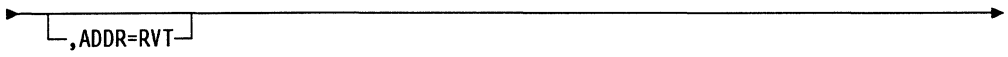
Format Label notation, register notation, SETZNC, SETZ, or SETNC.

Default If SUPV=YES, no validity checking is done. There is no default.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed.

For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the element address is not valid or to 0 if it is valid. For SETZNC or SETNC, the C latch is set to 0 if the element address is not valid or to 1 if it is valid.

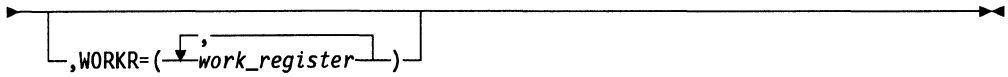
SETZ gives the most efficient macro expansion.



Function Specifies whether the address of the RVT entry or the address of the resource control block is to be returned.

Format RVT.

Default RVT.



Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying work registers makes the macro more efficient.

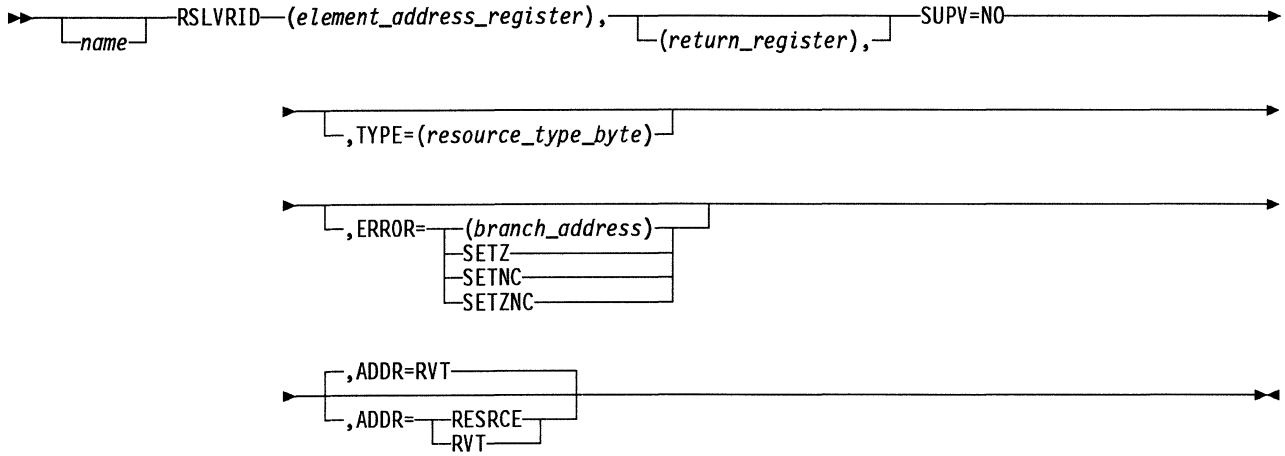
Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default No work registers.

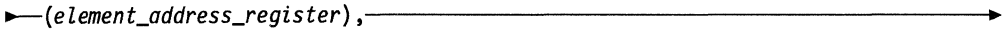
Remarks This keyword is valid only when SUPV=YES.

SUPV=NO Format

Syntax



Parameters

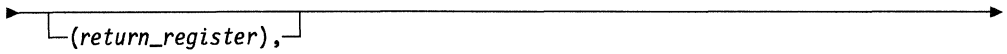


Function Specifies the register containing the element address to be resolved.

Format Register notation.

Default None.

Remarks Register 1 is not allowed.

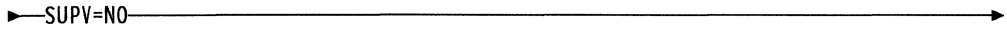


Function Specifies the register in which the address is to be returned.

Format Register notation.

Default The element address register is assumed also to be the return register.

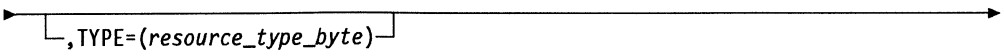
Remarks Register 2 is not allowed. Register 3 is standard.



Function Specifies that the issuer is running in level 5.

Format NO.

Default NO.



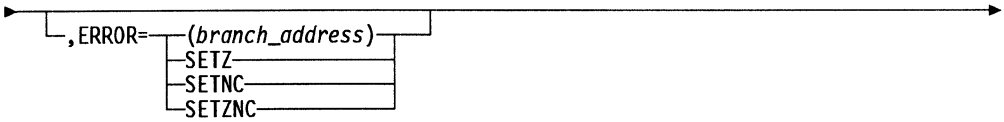
Function Specifies the register in which the resource type byte is to be returned.

Format Byte register notation.

Default No resource byte is returned.

Remarks The high-order half (0) of register 1 is standard.

Register 1 is modified when this keyword is specified.



Function Specifies either the address to be given control, or whether the C latch, Z latch, or both are to indicate whether the supplied element address is valid.

Format Label notation, register notation, SETZNC, SETZ, or SETNC.

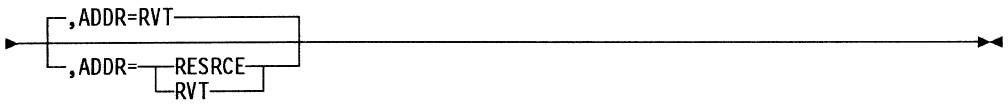
Default None.

Remarks Register 1 is not allowed.

For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the element address is not valid or to 0 if valid. For SETZNC or SETNC, the C latch is set to 0 if the element address is not valid or to 1 if valid.

SETZ gives the most efficient macro expansion.

If SUPV=NO, you must code this keyword.



Function Specifies whether the address of the RVT entry or the address of the resource control block is to be returned.

Format RVT or RESRCE.

Default RVT.

RSLVSNP—Locate an SSCP-NCP Session Control Block

The RSLVSNP macro converts the following items into the address of the corresponding SSCP-NCP session control block (SNP):

- SNP mask
- SSCP name
- Element address and virtual route vector table index (VVTI) pair.

The address is either stored in the physical services block (PSB) or returned to the user, depending on how the the SNPADDR keyword is specified. If a mask was used as input, the mask bit corresponding to the SSCP is turned off, and the residual mask is returned.

Register 0 is not allowed for register parameters. Register 1 is not allowed for any keyword if SUPV=NO. Register 6 is not allowed for any keyword if SUPV=YES.

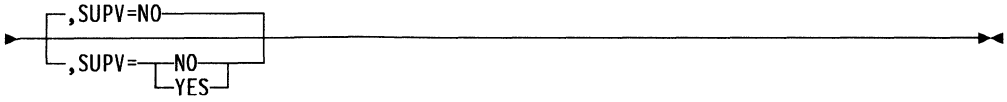
If you specify NETADDR or ELEMADR, you must specify the VVTI keyword.

MASK, SNPADDR, and CAPAB cannot share registers.

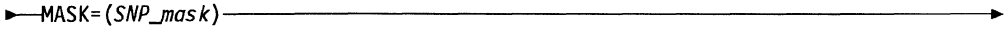
The ELEMADR or NETADDR and VVTI registers must be different.

You can use only one of the following input keywords on a given call: MASK, SSCPNME, NETADDR, or ELEMADR.

Syntax



Parameters



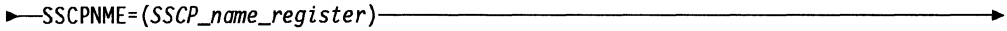
Function Specifies the register containing the SNP mask to be resolved.

Format Absolute byte register notation. Must be the low-order half of an odd-numbered register.

Default None.

Remarks Register 1 is not allowed.

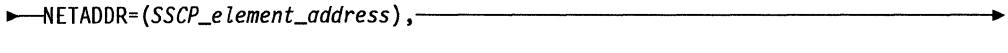
The residual mask is returned in the same manner as the register.



Function Specifies the register containing a pointer to the start of an SSCP name contained in storage. This name is compared to the SSCP name in the SNP. The SSCP name is left-justified, padded with trailing blanks, and halfword aligned.

Format Register notation.

Default None.



Function Specifies the register containing the element address to be resolved.

Format Register notation.

Default None.

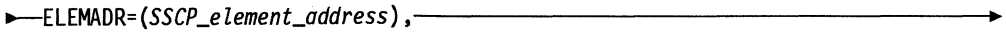
Remarks The keyword NETADDR is not recommended for new occurrences of this macro.

If SUPV=YES, register 3 is not allowed.

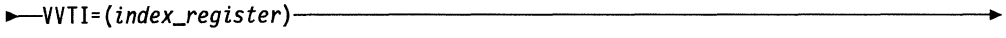
Register 2 is standard.

The register specified for NETADDR cannot be the same as that for the VVTI keyword.

Do not code NETADDR if you code ELEMADR. If you do code both NETADDR and ELEMADR, NETADDR is ignored, and a warning message is issued during assembly.



Remarks ELEMADR is synonymous with NETADDR except that it allows more accurate coding of keywords for new occurrences of this macro. All constraints on registers for NETADDR apply to this keyword. If both NETADDR and ELEMADR are used, NETADDR is ignored, and a warning message is given.

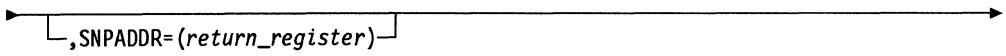


Function Specifies the VVTI associated with the element address to be resolved.

Format Register notation.

Default None.

Remarks If SUPV=YES, register 2 is not allowed. The VVTI keyword is required when NETADDR or ELEMADR is coded.

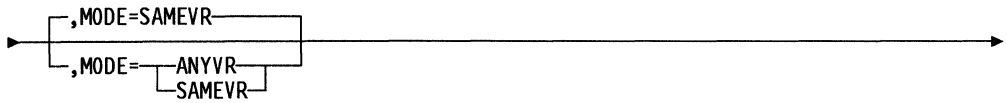


Function Specifies the register to return the SNP address. This register is set to 0 if the SNP is not resolved.

Format Absolute register notation.

Default If this keyword is omitted, the SNP address is placed into the PSBCSNPP field of the PSB.

Remarks If SUPV=YES, this keyword is required.

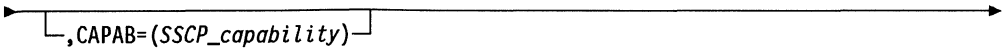


Function Specifies whether the SNP to be returned must be associated with the same virtual route as the specified VVT index or whether it can be on any virtual route ending in the subarea of the input VVT index.

Format SAMEVR or ANYVR.

Default If this keyword is omitted, SAMEVR will be assumed.

Remarks Specify this keyword only when the input is ELEMADR, VVTI.



Function Specifies the register in which the capability of the SSCP is to be returned (SNPSCAP1 field of the SNP).

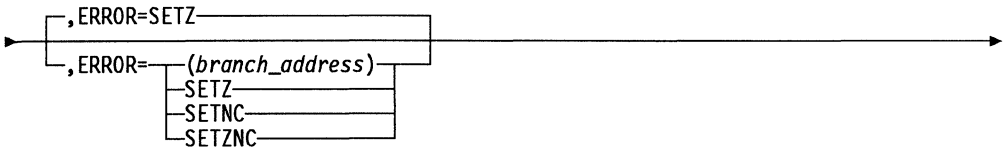
Format Byte register notation.

Default None.

Remarks The register specified for this keyword cannot be the same as those specified for SNPADDR or MASK.

The low byte of register 1 (0(1)) is not allowed.

Contents of the byte not specified for CAPAB will not be preserved.



Function Either specifies the address to be given control, or specifies whether the C latch, Z latch, or both are to indicate whether the supplied SSCP name or element/VVT index pair is valid.

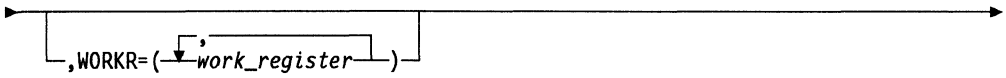
Format Label notation, register notation, SETZNC, SETZ, or SETNC.

Default SETZ.

Remarks For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the element address is valid. For SETZNC or SETNC, the C latch is set to 0 if the element address is not valid or to 1 if the element address is valid.

SETZ gives the most efficient macro expansion.

Do not use this register for the WORKR keyword.

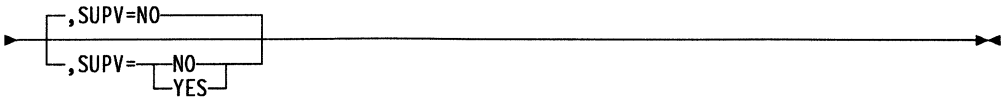


Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying these registers makes this macro more efficient.

Format You can use any or all of the following registers: 1, 2, 3, 4, 5, and 7. You cannot use equated values.

Default No work registers.

Remarks If SUPV=NO, this keyword is ignored.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in interrupt level.

Format YES or NO.

Default NO.

Remarks If SUPV=NO, an EXIT instruction is generated. If SUPV=NO, register 1 is not allowed, but the contents of register 1 are not destroyed.

RSLVSSCP—Convert an SNP Mask to a Network Address or VVT Index

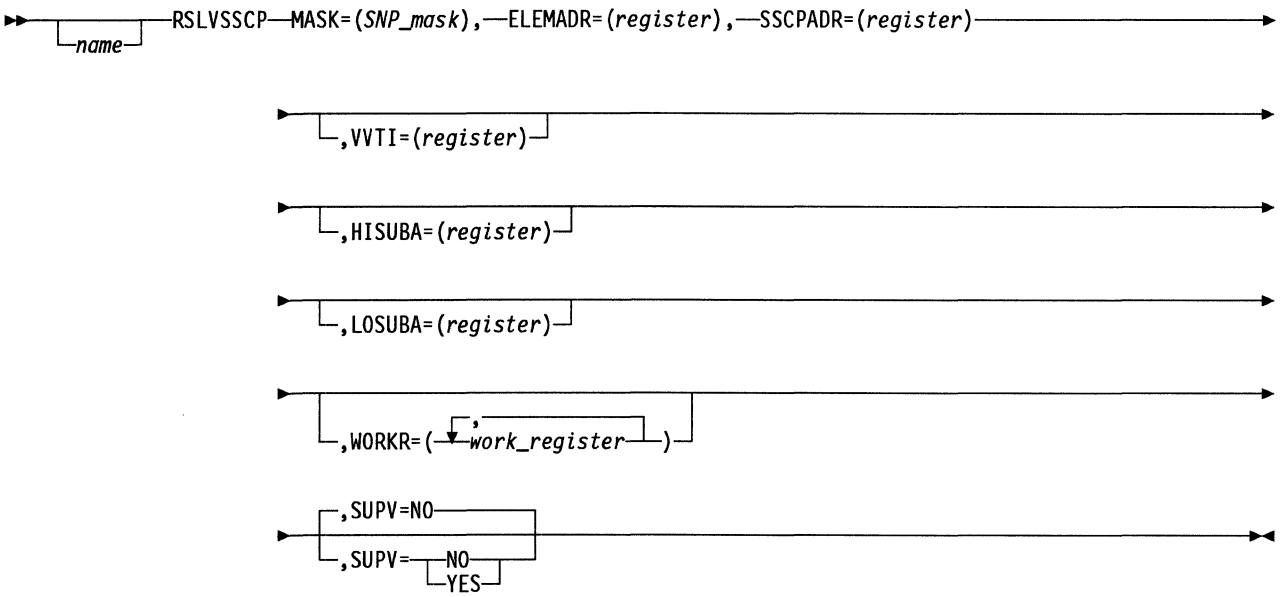
The RSLVSSCP macro converts a given mask or the highest order bit of an SNP mask to any one or combination of the following items for the corresponding SSCP:

- The high portion and the low portion of the subarea address
- The subarea address
- The element address
- The virtual route vector table (VVT) index.

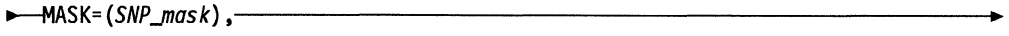
The corresponding bit in the mask register is turned off, and the resulting mask is returned.

Register 0 is not allowed. Register 1 is not allowed if SUPV=NO; register 6 is not allowed if SUPV=YES. Do not specify the same register for more than one keyword.

Syntax



Parameters



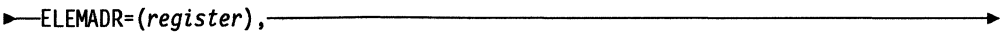
Function Specifies the register containing the SNP mask to be resolved.

Format Byte register notation. Must be the low-order half of an odd-numbered register.

Default None.

Remarks Register 1 is not allowed.

The residue mask is returned in the same manner as the register.

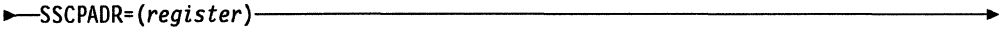


Function Specifies the register in which the element address of the SSCP is to be returned.

Format Register notation.

Default None.

Remarks Avoid using ELEMADR or SSCPADR alone as output and code at least one other output keyword.

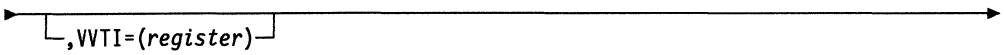


Function Specifies the register in which the address of the SSCP is to be returned.

Format Register notation.

Default None.

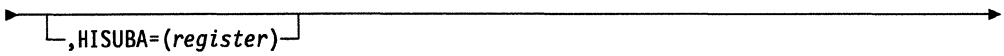
Remarks This keyword is equivalent to ELEMADR except that ELEMADR allows more accurate and meaningful coding of keywords for new occurrences of this macro. SSCPADR is retained to minimize source changes required to the programs written before extended network addressing. If both SSCPADR and ELEMADR are specified, SSCPADR is ignored and a warning message is issued.



Function Specifies the register for the virtual route vector table index (VVTI) associated with the SSCP requested.

Format Register notation.

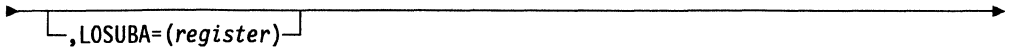
Default None.



Function Specifies the register to return the high portion of the subarea address of the SSCP. This part of the address is right-justified when it is returned.

Format Register notation.

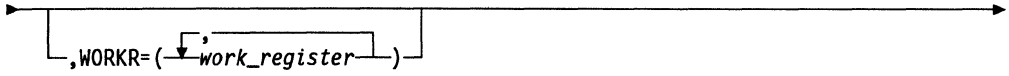
Default None.



Function Specifies the register to return the low halfword of the subarea address of the SSCP. The address is right-justified when it is returned.

Format Register notation.

Default None.

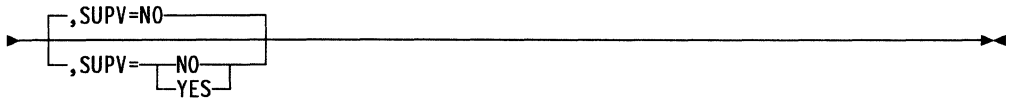


Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying this keyword makes this macro more efficient.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Equated values cannot be used.

Default No work registers.

Remarks If SUPV=NO, this keyword is ignored.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in interrupt level.

Format YES or NO.

Default NO.

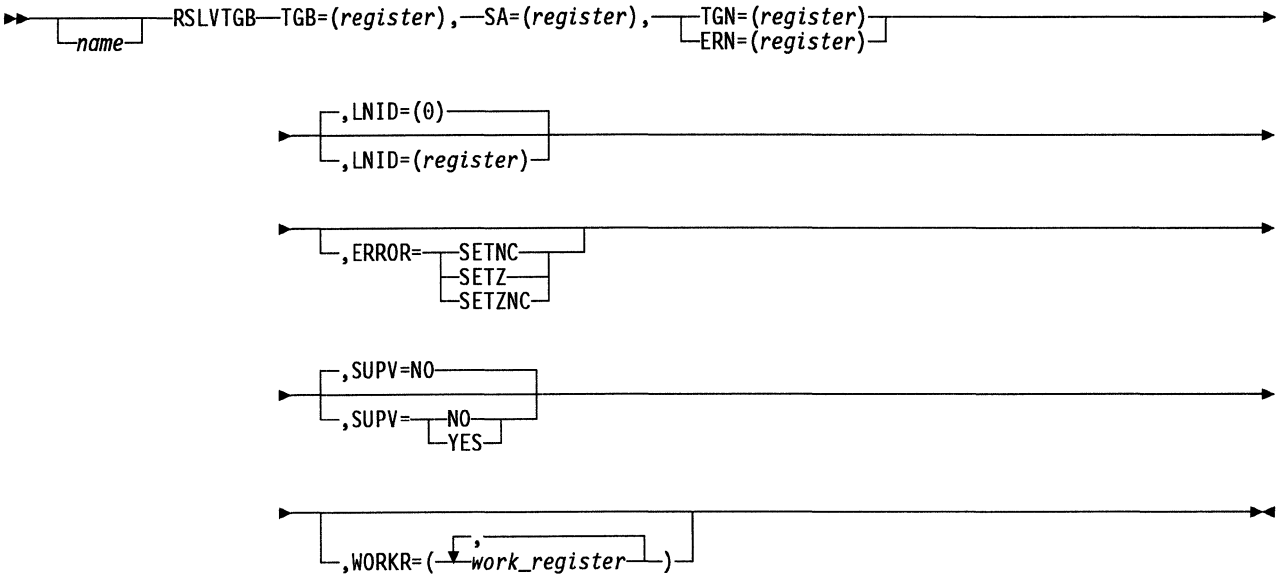
Remarks If SUPV=NO, an exit instruction is generated.

RSLVTGB—Locate an Outbound Transmission Group Control Block

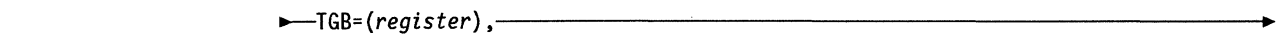
The RSLVTGB macro converts either a subarea address (SA) and transmission group number (TGN) or a subarea address and an explicit route number (ERN) into a pointer to the corresponding transmission group control block (TGB) used for outbound routing. In a gateway environment, the identifier of the network to be searched must be specified. Unless you specify local network identification (LNID), the native network is assumed.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register in which the TGB address is to be returned.

Format Absolute register notation.

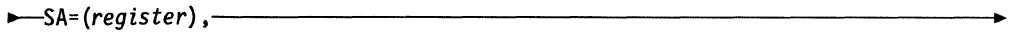
Default None.

Remarks The register cannot be the same as the register specified on the SA, TGN, ERN, or WORKR keyword.

If SUPV=YES, register 6 is not allowed.

Register 7 is standard.

A value of 0 is returned if the TGB address could not be returned.



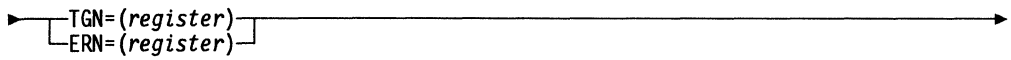
Function Specifies the register containing the network subarea address to be resolved. The subarea address is right-justified and not shifted.

Format Absolute register notation.

Default None.

Remarks The register cannot be the same as the register specified on the TGN, ERN, LNID, WORKR, or TGB keyword.

If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed. Register 2 is standard.



Function Specifies the register containing the TGN or ERN to be resolved.

Format Absolute register notation.

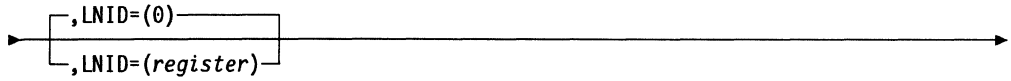
Default None.

Remarks You can code the TGN or ERN keyword, but not both.

The register cannot be the same as the register specified on the SA, WORKR, LNID, or TGB keyword.

TGN and ERN cannot both be blank.

If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed. Register 3 is standard.



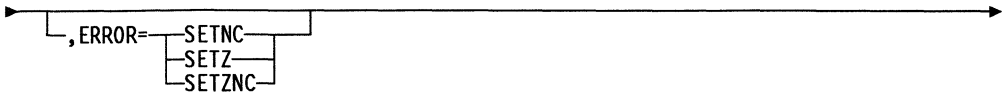
Function Specifies the byte register containing the local network identification to be used to resolve the TGB.

Format Byte register notation. It must be in the low-order half of an odd-numbered register.

Default Native network, LNID=0.

Remarks If SUPV=NO, register 1 is not allowed.

The register used cannot be the same as the register specified on the TGN, ERN, WORKR, TGB, or SA keyword.



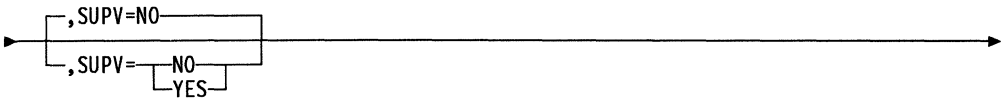
Function Specifies whether the C latch, the Z latch, or both are to indicate whether the requested resolution can be done.

Format SETZ, SETNC, or SETZNC.

Default None.

Remarks For all three specifications, the Z latch is set to 1 if the resolution cannot be done or to 0 if the resolution can be done. The C latch is set to 0 if the resolution cannot be done or to 1 if the resolution can be done.

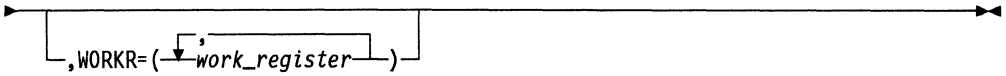
For SUPV=NO, the C and Z latches are set as above whether the keyword is coded or not. For SUPV=YES, latch settings are indeterminate if ERROR is not coded.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7.

Default No work registers.

Remarks This keyword is valid only with SUPV=YES.

The registers cannot be the same as the registers specified on the TGB, ERN, LNID, SA, or TGN keyword.

RSLVVTI—Get Virtual Route Control Block Data

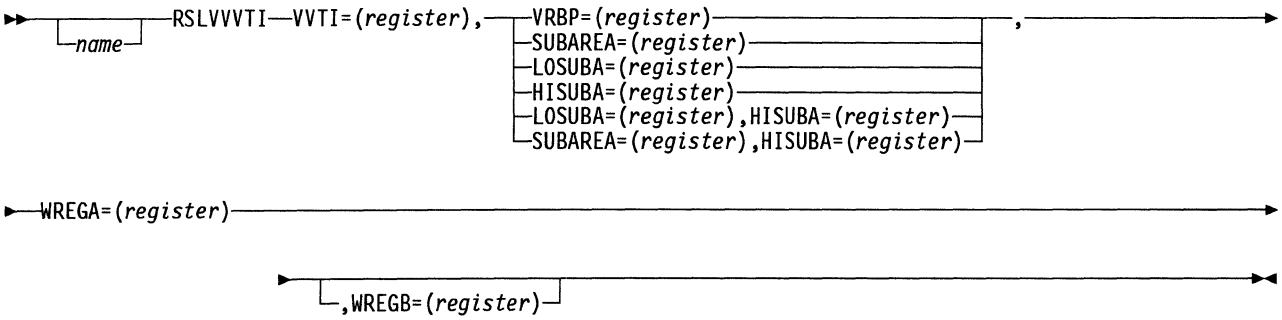
The RSLVVTI macro indexes the virtual route vector table (VVT) to get either the address of a virtual route control block (VRB) or the low- or high-order portion of the subarea address of the other end of the virtual route. If the VVT index is not valid or if the VRB pointed to by the VVT is not assigned to a virtual route, the VRB address or the portion of the subarea address returned is 0, and the Z latch is set.

This macro generates inline code.

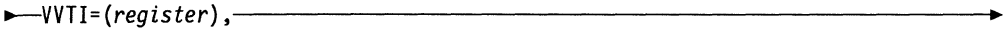
The VVT DSECT, XCXTVVT, must be available and addressability must be established to the VVT using EXTRN CXTVVT. If SUBAREA, LOSUBA, or HISUBA is to be returned, the VRB DSECT, XCXTVRB must be available.

Note that all registers must be different. Register 0 is not allowed.

Syntax



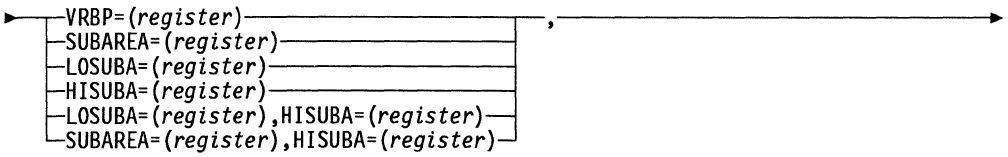
Parameters



Function Specifies the register containing the VVT index (VVTI).

Format Register notation.

Default None.



Function Specifies the register in which either the VRB address or the other-end SUBAREA, LOSUBA, or HISUBA of the virtual route is to be returned.

Format Register notation.

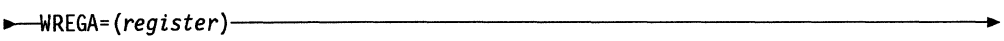
Default None.

Remarks If the VVTI is not valid or if the VRB is not assigned, the VRB address or SUBAREA, LOSUBA, or HISUBA returned will be 0.

The registers must not be the same as the register specified on the VVTI keyword.

HISUBA returns the high-order portion of the other end of the virtual route.

Do not specify both LOSUBA and SUBAREA. They each return the low-order portion of the subarea address of the other end of the virtual route. LOSUBA is preferred.



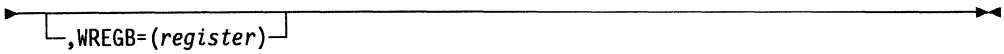
Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register must not be the same as the register specified on the VVTI, VRBP, SUBAREA, LOSUBA, or HISUBA keywords.

The code will be more efficient if you use an odd-numbered register.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks If you do not specify WREGB but do specify VRBP, the contents of the register specified by the VVTI keyword are destroyed.

If you do not specify WREGB but do specify only the SUBAREA, LOSUBA, or HISUBA keyword, the contents of the register specified by the VVTI keyword are destroyed.

If you do not specify WREGB but you specify HISUBA in addition to SUBAREA or LOSUBA, the contents of the register specified by the VVTI keyword are not destroyed.

The register must not be the same as the register specified on the VVTI, VRBP, SUBAREA, LOSUBA, HISUBA, or WREGA keywords.

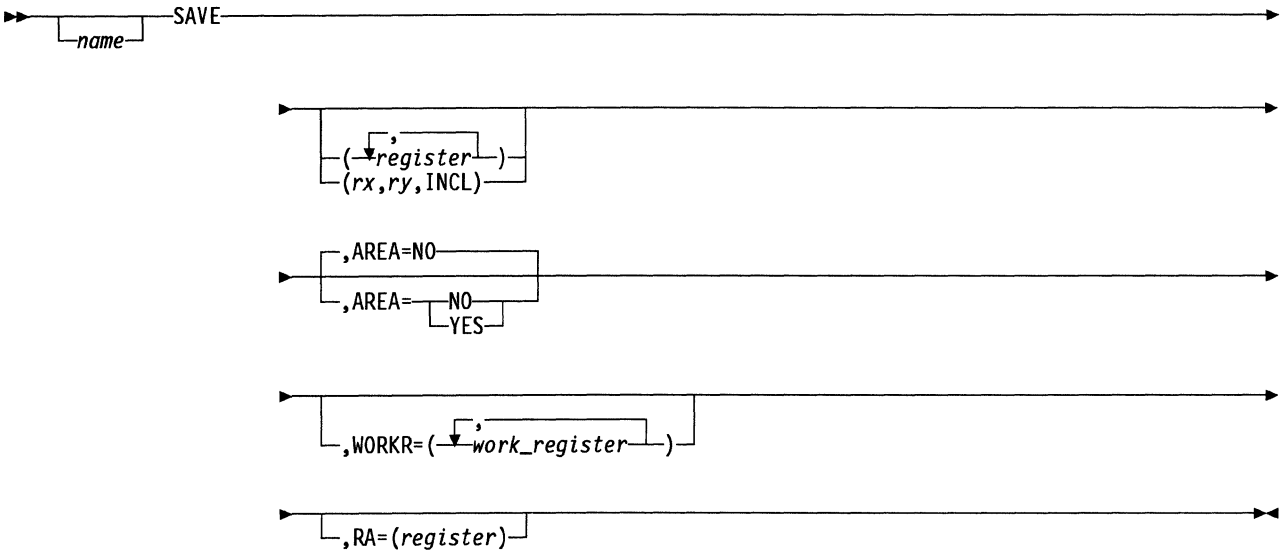
SAVE—Store Registers

The SAVE macro stores the specified registers in the save area pointed to by register 6 and updates register 6 to point to the next save area in the save area chain. When you use the SAVE macro with the optional LINK macro RA=B keyword, you can save the return address in the previous save area.

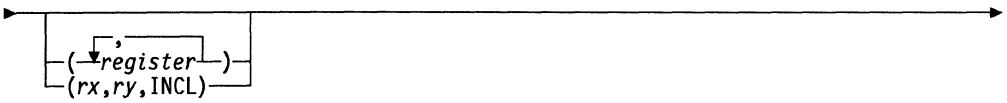
The SAVE macro does not save registers designated as work registers on the WORKR keyword. Use this function for macros that allow you to specify work registers that do not need to be saved.

Register 0 is not allowed for register parameters.

Syntax



Parameters



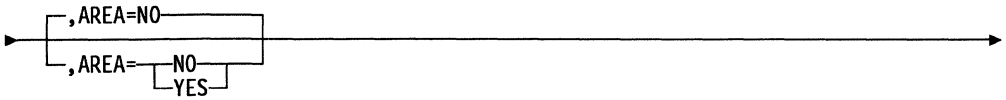
Function Specifies the registers to be saved. (*register*) specifies the individual registers to be saved. (*rx*,*ry*,INCL) specifies a range of registers to be saved.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7.

Default No registers are saved.

Remarks If you specify the (*register*) format, and you specify the same register for WORKR, the register will not be saved.

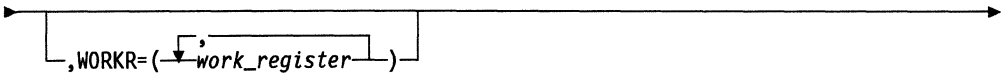
If you specify with (*rx*,*ry*,INCL), WORKR=(*register*) is not valid.



Function Specifies whether register 6 is to be updated. YES specifies that register 6 is to be updated to point to the next save area in the chain after the registers are saved. NO specifies that register 6 is not to be updated.

Format YES or NO.

Default NO.

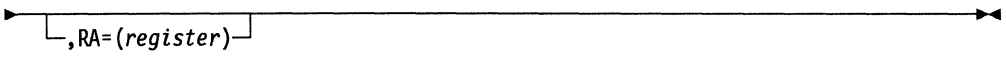


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. You cannot use label register values.

Default No work registers; all registers specified in the first keyword are saved.

Remarks If you specify a register in the first parameter with the (*register,...*) format and you specify the same register for WORKR, the contents of the register are not saved. If you specify a register in the first parameter with (*rx,ry*,**INCL**), the WORKR keyword is not valid.



Function Specifies the register that contains the return address. The contents of this register are stored at offset 8 in the save area pointed to by register 6. This register is then available for use by the RESTORE macro. If this register is also in the first keyword position, it will be saved there.

Format You can specify register 1, 2, 3, 4, 5, or 7.

Default No register saved.

SAVEAREA—Create Inline Storage for Registers

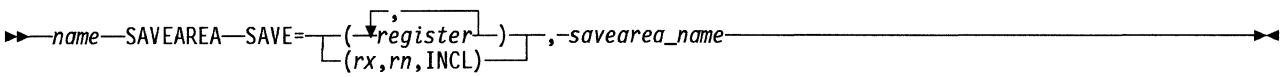
The SAVEAREA macro uses load address instructions to create inline storage for your registers. Use the SAVEAREA macro when you do not have register 6 pointing to a regular save area.

The SAVEAREA macro generates the following:

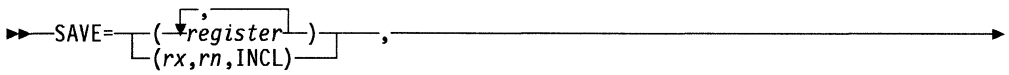
- An 8-byte chain field for forward and backward chaining of the save area
- A load address (LA) instruction for each register to be saved when this save area is referred to a CALL macro
- A branch to the next sequential instruction after the CALL macro that last referred to this save area
- A global set symbol telling the supervisor which registers are to be saved in this save area.

When a CALL macro is issued, the supervisor stores the appropriate register in each LA instruction. It then stores the address of the next sequential instruction after the CALL macro in the last instruction generated by this SAVEAREA macro. When a subroutine returns control to the calling program, execution begins at the first LA instruction in the save area generated by the SAVEAREA macro. Execution proceeds through each LA instruction until all specified registers are restored. At this point the program branches to the mainline instruction, and normal processing continues.

Syntax



Parameters



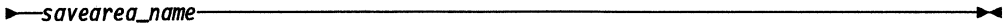
Function Specifies the registers to be saved.

Format Absolute notation.

Default None.

Remarks If a series of registers is to be saved, *rx* and *rn* specify the limits of the range of registers. The range is specified in ascending sequence, and the symbol INCL is coded as the third parameter to specify a series. Register 1 cannot be saved. If a random grouping of registers is to be saved, specify each register.

Register 0 is not allowed.



Function Specifies the name of the save area, and is identical to the name specified on the SAVE keyword of the corresponding CALL macro.

Format Any symbol with up to 7 characters.

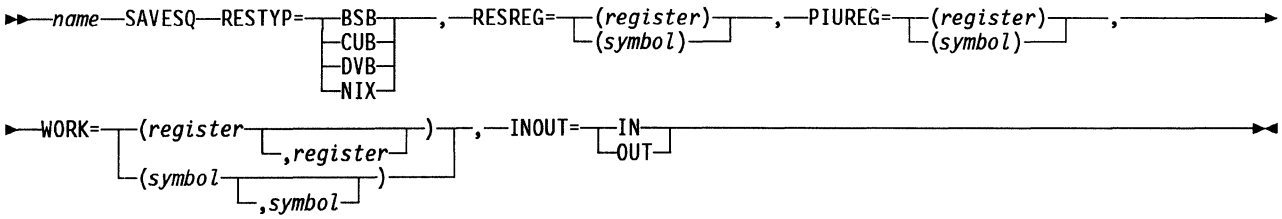
Default None.

SAVESQ—Record PIU Sequence Numbers when the Session Trace Is Active

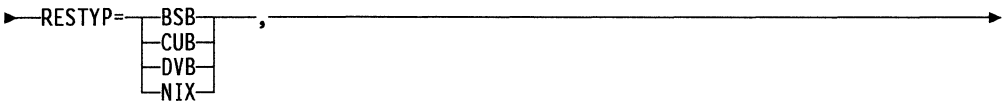
The SAVESQ macro is used to record path information unit (PIU) sequence numbers when the session trace is active. The numbers are stored in the specified control block: DVB, CUB, BSB, or NIX.

Register 0 is not allowed for register parameters.

Syntax



Parameters



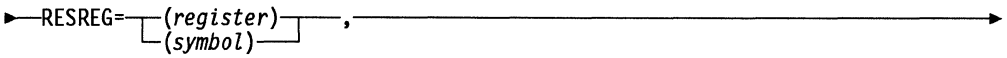
Function Specifies the type of control block into which the sequence numbers are to be stored.

Format DVB, CUB, BSB, or NIX only.

Default None.

Remarks Include the DSECT for the specified control block, as follows:

Control Block	Macro
DVB	XCXTDVB
CUB	XXCXTCUB
BSB	XCXTBSB
NIX	XCXTNIX



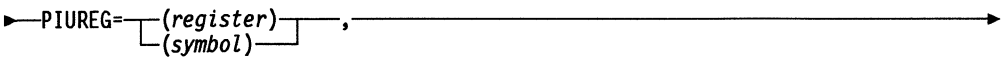
Function Specifies the register containing the address of the resource control block to be updated.

Format Register or symbol notation. You can specify register 1, 2, 3, 4, 5, or 7.

Default None.

Remarks If CUB is specified in RESTYP, a second work register must be specified.

The register must not be the same as that specified for any other keyword.

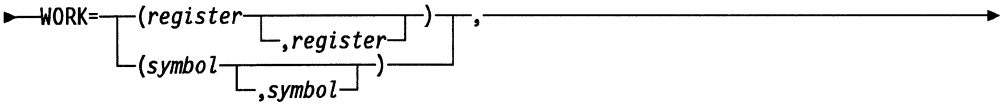


Function Specifies the register containing the address of the PIU being traced.

Format Register or symbol notation. You can specify register 1, 2, 3, 4, 5, or 7.

Default None.

Remarks The register must not be the same as that specified for any other keyword.



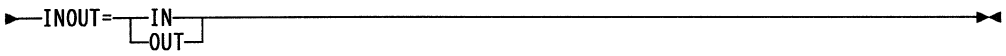
Function Specifies one or more registers, the contents of which may be altered during execution of the macro.

Format Register or symbol notation. For the first keyword value, you can specify register 1, 3, 5, or 7. For the second keyword value, you can specify register 1, 2, 3, 4, 5, or 7.

Default None.

Remarks If you specify RESTYP=CUB, you must code the second keyword value; otherwise it is ignored.

The register must not be the same as that specified for any other keyword.



Function Specifies whether the PIU being traced is an incoming or outgoing PIU.

Format IN if the PIU is incoming; OUT if the PIU is outgoing.

Default None.

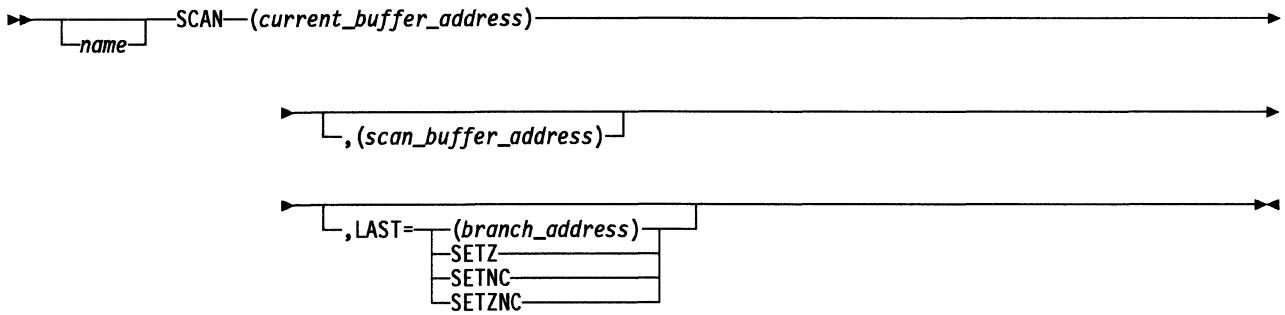
SCAN—Get the Address of the Next Buffer in a Chain

The SCAN macro returns the address of the buffer following a specified buffer in a buffer chain. This macro can also be used to determine whether the specified buffer is the last one in a chain of buffers. The Z latch is set to 1 if the buffer is last and to 0 if it is not last; the C latch is set conversely.

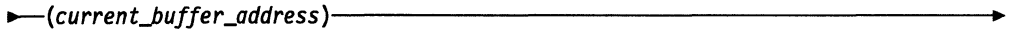
The SCAN macro always expands inline.

Register 0 is not allowed for register parameters.

Syntax



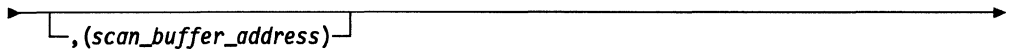
Parameters



Function Specifies the register containing the currently accessed buffer address.

Format Register notation.

Default None.

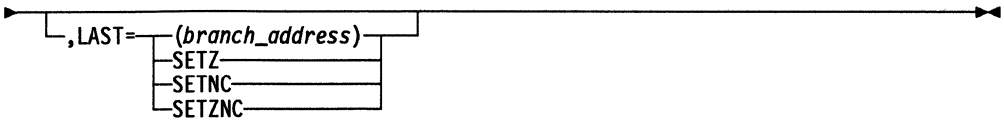


Function Specifies a register to receive the address of the next buffer in the chain.

Format Register notation.

Default The address of the next buffer is returned in the current-buffer register.

Remarks The register is set to 0 if the current buffer is the last in the chain.



Function Either specifies the address to be given control, or specifies whether the C latch, Z latch, or both are to indicate whether the current buffer is the last buffer in the chain.

Format Register notation, label notation, SETZNC, SETZ, or SETNC.

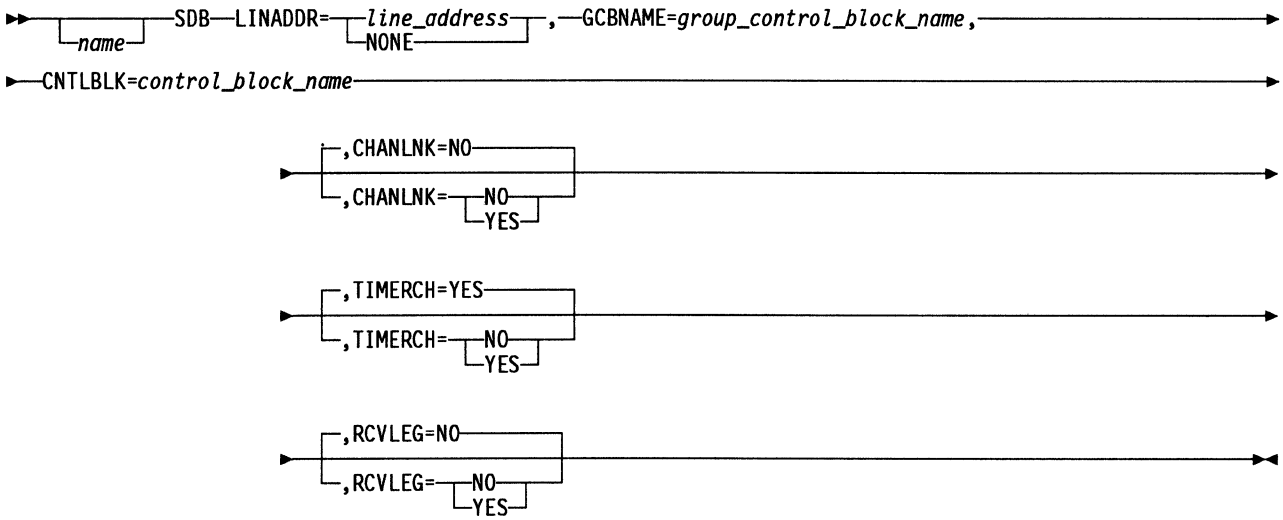
Default The next instruction after the macro is given control.

Remarks For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the positional buffer is the last buffer in the element or to 0 if it is not the last. For SETZNC or SETNC, the C latch is set to 0 if the current buffer is the last buffer in the chain or to 1 if it is not the last.

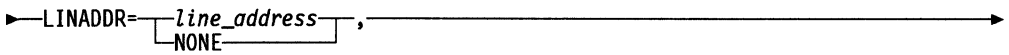
SDB—Build IBM-Required UACB Fields

The SDB macro builds the IBM-required user adapter control block (UACB) fields. This macro must immediately follow each UACB in the control block assembly source code.

Syntax



Parameters



Function *line address* specifies the line address associated with this UACB. NONE indicates that the UACB is not associated with a physical line.

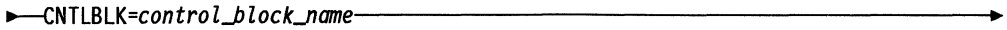
Format Absolute notation.

Default None.

Function Specifies the group control block (GCB) associated with this UACB. The GCBNAME must match the symbol for a GROUP definition statement.

Format Any symbol.

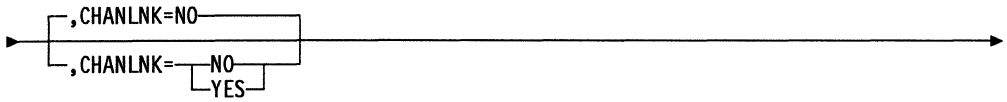
Default None.



Function Specifies the name of the UACB control block generated by this macro. The name must be unique for each SDB macro used. NCP uses this macro for timer chaining purposes.

Format Any valid symbol.

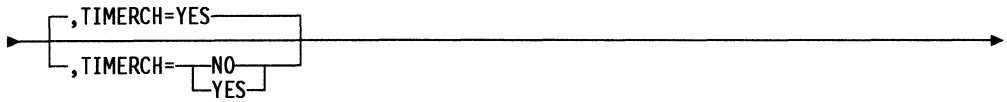
Default None.



Function Specifies whether this UACB is for channel or link operations. CHANLNK=YES means the UACB is for a channel, while CHANLNK=NO means the UACB is for a link.

Format YES or NO.

Default NO.

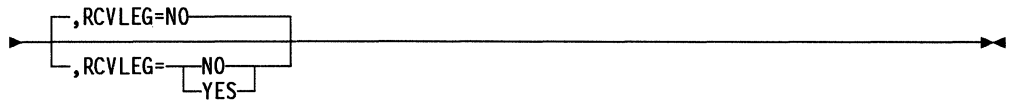


Function Specifies whether this UACB is to be included in the level 3 timer chain. If TIMERCH=NO, this particular UACB will not be available for timer-dependent operations and will not be included in the level 3 timer chain.

Format YES or NO.

Default YES.

Remarks You can specify TIMERCH=NO for virtual resources that will never be associated with timer-dependent events (such as NPSI virtual circuits) to obtain a significant performance improvement. However, specifying TIMERCH=NO when the resource does have timer-dependent events will lead to a resource failure.



Function Specifies whether this line address is for the receive leg of a duplex line.

Format YES or NO.

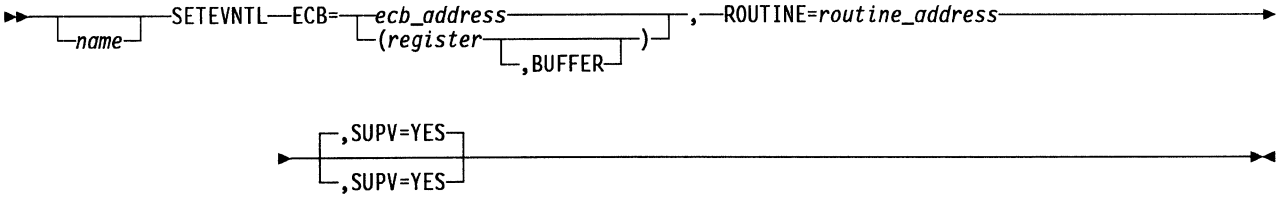
Default NO.

SETEVNTL—Create a Link between an Event and a Processing Routine

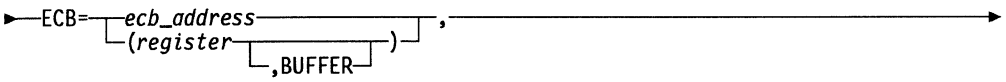
The SETEVNTL macro establishes the linkage between an event and a processing routine to be activated upon completion of the event. This macro is never used by level 5 code.

Register 0 is not allowed for register parameters.

Syntax



Parameters



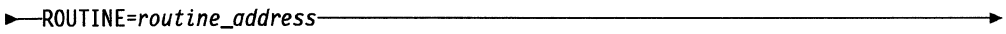
Function Specifies the address of the event control block.

Format Register or label notation.

Default None.

Remarks Neither register 1 nor register 6 is allowed.

If the event control block (ECB) is not defined in storage but is part of a block control unit (BCU), you must use register notation and specify BUFFER. The register must point to the beginning of the first buffer of the BCU.

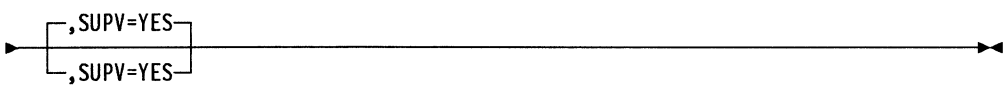


Function Specifies the address of the supervisor program that is to receive control on event completion.

Format Register or label notation.

Default None.

Remarks Neither register 1 nor register 6 is allowed.



Function Specifies that the issuer is running in an interrupt level.

Format YES.

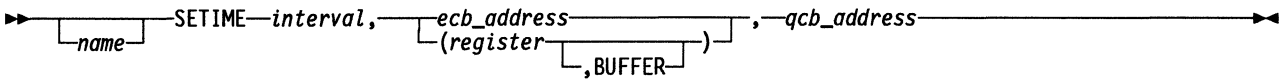
Default YES.

SETIME—Schedule an Interrupt and a Task for a Time Interval

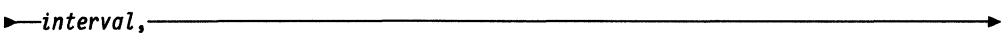
The SETIME macro schedules an interrupt for a specified time interval and schedules a specified task at the expiration of the timer interval.

Register 0 is not allowed for register parameters.

Syntax



Parameters



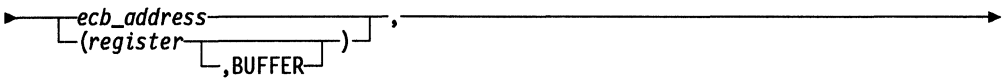
Function Specifies an integral number of seconds for the time interval.

Format Absolute, label, or register notation.

Default None.

Remarks This interval cannot exceed 5400 seconds (1/16 of a day).

Register 1 is not allowed.



Function Specifies the event control block (ECB) to be used by the supervisor for the duration of the interval.

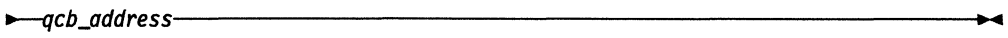
Format Register or label notation.

Default None.

Remarks A separate ECB is specified for each unexpired interval to be scheduled.

Register 1 is not allowed.

If the ECB is not defined in storage but is part of a block control unit (BCU), you must use register notation, and you must specify BUFFER. The register must point to the beginning of the first buffer of the BCU.



Function Specifies the address of the input or pseudo-input queue control block governing the task to be started when the interval elapses.

Format Register or label notation.

Default None.

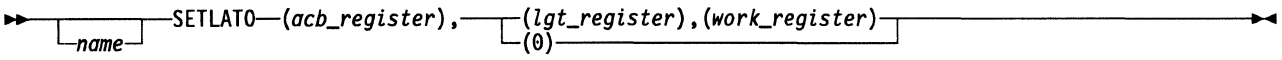
Remarks Register 1 is not allowed.

SETLATO—Set the Link Activity Time-Out Field in the CCB

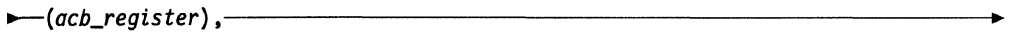
The SETLATO macro is used to set up the link activity time-out (LATO) field in the character control block (CCB) for secondary SDLC links. This macro is used in program levels 2 and 3 and requires that the XCXTDDB, XCXTACB, and XCXTLGT DSECTs be present, and that addressability be established.

Register 0 is not allowed for register parameters.

Syntax



Parameters

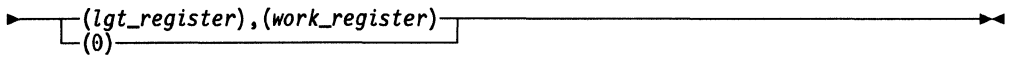


Function Specifies the register containing the address of the adapter control block (ACB).

Format Register notation.

Default None.

Remarks Addressability to the ACB must be set up by the program invoking the macro.



Function Specifies whether the contents of the LATO field should be set to 0, or if the system-generated LATO field from the line group table (LGT) should be copied to the CCB.

Format Register notation.

Default None.

Remarks Code 0 if the contents of the LATO field should be set to 0.

lgt reg is the register containing the address of the LGT.

work reg is any register (other than *acb reg* or 0) that can be used as a work register (coded only if *lgt reg* is not equal to 0).

Addressability to the LGT must be set up by the invoking program only if *lgt reg* is not equal to 0.

The inline code generated by this macro is either code to store a halfword of zeros in CCBLATO if *lgt reg* is equal to 0, or code to move LGTLATO into CCBLATO.

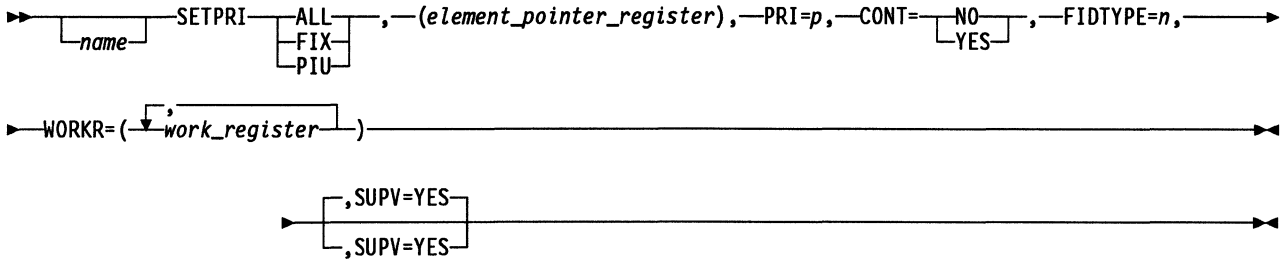
SETPRI—Set the Priority of an Element in a Queue

The SETPRI macro sets the priority of any element, except a queue control block (QCB), that can be placed on a queue. The priority is set in the ECBCSTAT field of the element's event control block (ECB). You must provide a save area that can be overwritten, and the QCB DSECT (XXQCB) must be available. SETPRI requires that the XSYSEQU macro be invoked.

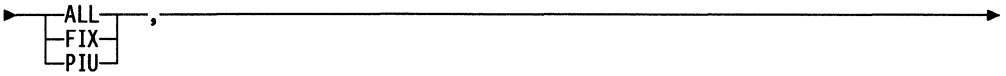
Register 0 is not allowed for register parameters.

Note: This macro can be executed by interrupt-level code only.

Syntax



Parameters



Function Specifies the keyword to be used.

If you specify the PIU keyword, the priority of the specified path information unit (PIU) is set to the designated priority. If the PIU has been queued, its priority is not set.

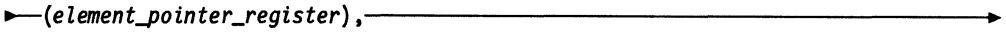
If you specify the FIX keyword, the priority of the first element on the specified queue is set to the highest priority (15).

If you specify the ALL keyword, the priority of all elements on the specified queue is set to the designated priority.

Format PIU, FIX, or ALL.

Default None.

Remarks Execution of this macro is terminated when a QCB is found on the queue, regardless of the keyword.



Function Specifies the address of a PIU or a QCB.

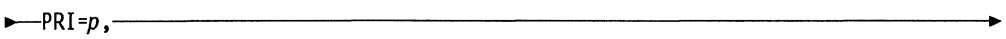
Format Register notation.

Default None.

Remarks If you specify the FIX or ALL keyword, you must use a QCB address. If you specify the PIU keyword, you must use a PIU address.

Register 3 is standard. Neither register 1 nor register 6 is allowed.

The register content is not altered.



Function Specifies the priority value to be set in the element's ECB.

Format Absolute or label notation. You can use any value from 0 to 15.

Default None.

Remarks If the element has not been enqueued, the priority value set by SETPRI is used when the element is enqueued with PRI=YES on the ENQUE macro instruction.

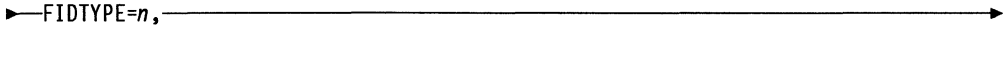
You cannot specify both the FIX keyword and the PRI keyword.

Function Specifies whether the queue is a contention queue (one that can be manipulated simultaneously by two or more interrupt levels).

Format YES or NO.

Default None.

Remarks If CONT=YES, level 3 interrupts are disabled. This keyword is ignored when PIU option is used.

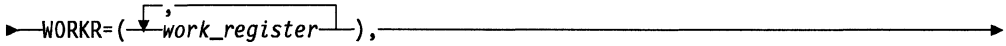


Function Specifies the FID type of the PIU.

Format 0, 1, 2, 3, or 4. Only absolute values may be used.

Default None.

Remarks When you specify the FIX or ALL keyword, this keyword is ignored.



Function Specifies one or more work registers, the contents of which may be altered during execution of the macro.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default No work registers.

Function Specifies that the issuer is running in an interrupt level.

Format YES.

Default YES.

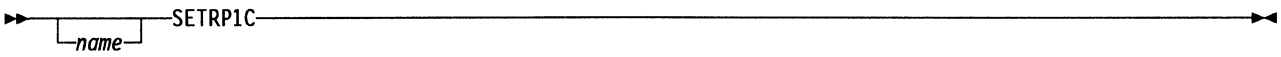
Remarks This parameter is ignored when the PIU keyword is coded.

SETRP1C—Set the System Response Phase to Phase 1

The SETRP1C macro sets the system response phase to phase 1 if it is currently phase 0. If the system response phase is not phase 0, no change is made.

Include and address the block control unit (BCU) DSECT (XXCXTBCU).

Syntax

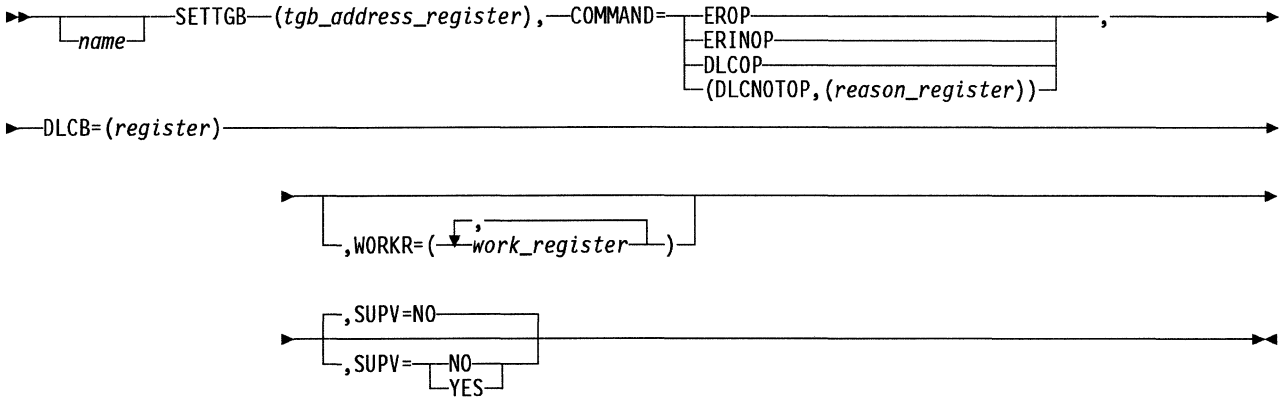


SETTGB—Control the Activation of a Transmission Group

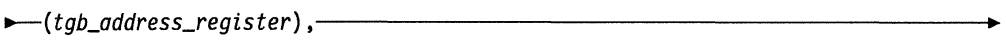
The SETTGB macro controls the process of activating and deactivating a specified transmission group.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register containing the address of the transmission group control block (TGB).

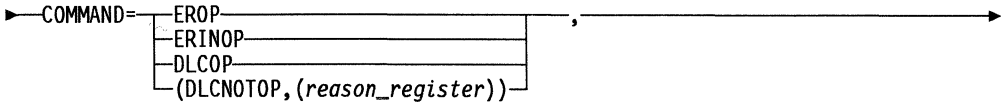
Format Register notation.

Default None.

Remarks If SUPV=NO, register 1 is not allowed.

If SUPV=YES, register 2 is standard and register 6 is not allowed.

The register must not be the same as the register specified on the DLCB or WORKR keywords.



Function Specifies the event that will take place and causes an action to be taken by the corresponding transmission group.

EROP indicates that the explicit route manager function has identified all NCP operative routes in the transmission group's network to the adjacent node.

ERINOP indicates that the explicit route manager function has completed all processing associated with terminating the operative state for all explicit routes associated with the specified TGB.

DLCOP indicates that the data link control (DLC) currently associated with the specified TGB is operational (capable of exchanging normal data traffic with the adjoining node).

(DLCNOTOP, (*reason register*)) indicates that the DLC currently associated with the specified TGB is no longer operational (incapable of exchanging normal data traffic with the adjoining node). (*reason register*) contains the reason code for the DLC going inoperative. The reason code is saved to make it accessible through the TESTTGB macro.

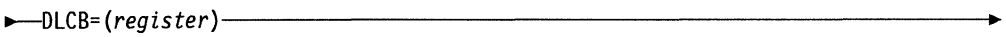
EROP and ERINOP commands must not be issued if the TGB is on a system queue, for example, RCQ.

Format EROP, ERINOP, DLCOP, (DLCNOTOP,(*reason register*)). The reason register must be a byte register.

Default None.

Remarks If SUPV=NO and COMMAND=(DLCNOTOP,(*reason register*)) are specified, the contents of register 1(0) are not preserved by this macro.

Register 1(0) is standard for (*reason register*).



Function Specifies the data link control block associated with the DLC event that will take place.

Format Register notation.

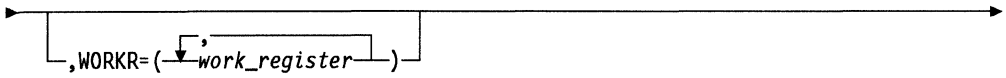
Default None.

Remarks This keyword is required when COMMAND=DLCOP or (DLCNOTOP,(*reason register*)); you cannot code it when COMMAND=EROP or ERINOP is coded.

If SUPV=NO, register 1 is not allowed.

If SUPV=YES, register 3 is standard and register 6 is not allowed.

The register must not be the same as the register specified for the TGB address register or on the WORKR keyword.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

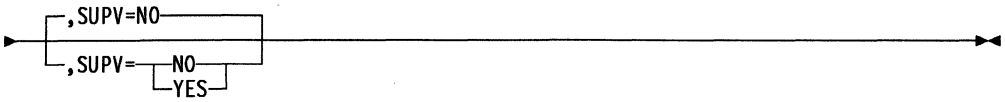
Format Register notation.

Default No work register.

Remarks This keyword is valid only with SUPV=YES.

The register must not be the same as the register specified for the TGB address register or on the DLCB keyword.

You can code up to five of the following registers: 1, 2, 3, 4, 5, and 7.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

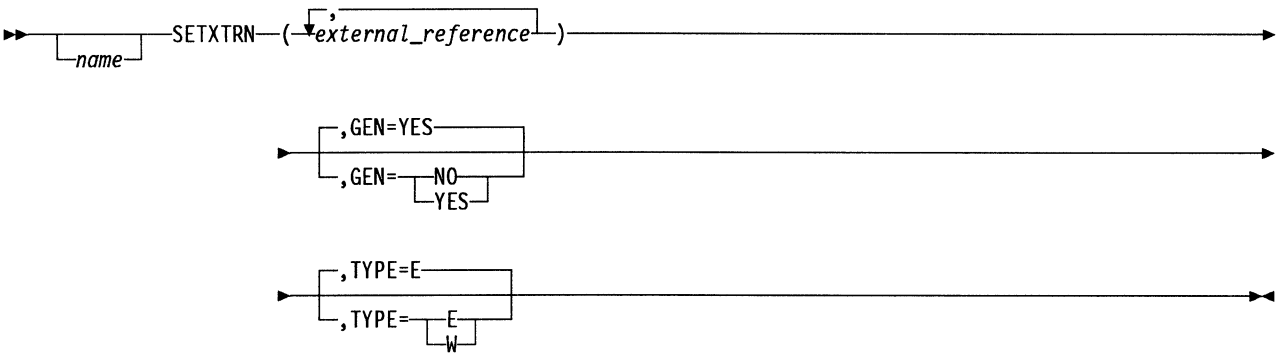
Format YES or NO.

Default NO.

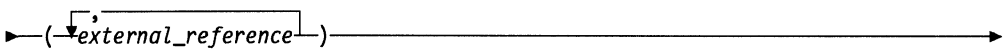
SETXTRN—Control EXTRN and WXTRN Assembler Statements

The SETXTRN macro controls the generation of EXTRN and WXTRN assembler statements. An EXTRN statement is generated only on the first occurrence of a given reference in a SETXTRN macro. For all other occurrences in the same assembly, the EXTRN or WXTRN statement is omitted.

Syntax



Parameters

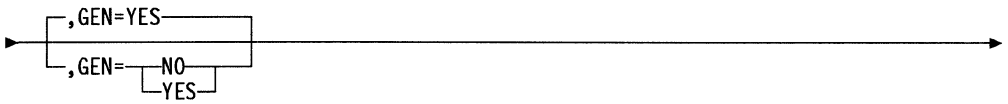


Function Specifies the labels that are external references. If appropriate, the macro generates an EXTRN or WXTRN statement.

Format One or several external references. The sublist form is allowed.

Default None. Specify at least one reference.

Remarks Specify a maximum of 399 different external references for each assembly run.

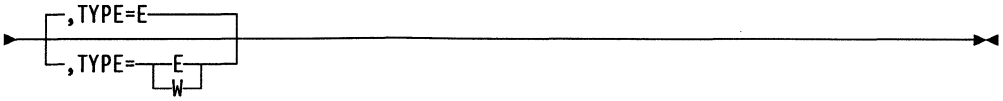


Function YES specifies that an EXTRN or WXTRN statement will be generated at the first occurrence of a reference on a SETXTRN macro in the assembly.

Format YES or NO.

Default YES.

Remarks Use SETXTRN with GEN=NO for references that are entry points in the actual module. Code the ENTRY statements separately. Succeeding SETXTRN macros with GEN=YES do not generate EXTRN or WXTRN statements for these entry points.



Function Specifies the type of statement to be generated. TYPE=E specifies that an EXTRN statement will be generated; TYPE=W specifies that a WXTRN statement will be generated.

Format E or W.

Default E.

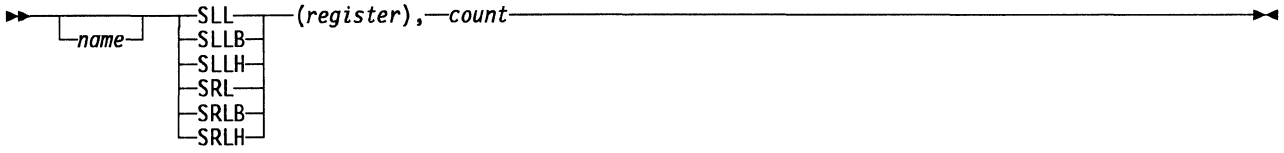
Remarks Use SETXTRN with TYPE=E for references to entry points that are always included. The link-edit will fail if the entry point cannot be found. Use SETXTRN with TYPE=W for reference points that are not always included. The link-edit will not fail if the entry point cannot be found.

Shift Macros—Shift a Register to the Left or Right

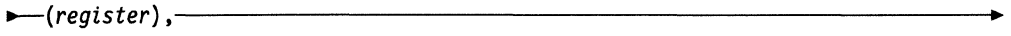
The shift macros generate the statements that shift a register a specified number of bit positions to the right or left. The format is the same for all shift macros. The shift macros are as follows.

Macro	Function
SLL	Shifts a 22-bit register to the left.
SLLB	Shifts a register byte to the left.
SLLH	Shifts a 16-bit register to the left.
SRL	Shifts a 22-bit register to the right.
SRLB	Shifts a register byte to the right.
SRLH	Shifts a 16-bit register to the right.

Syntax



Parameters

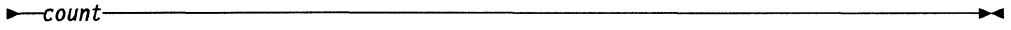


Function Specifies a register to be shifted.

Format Register notation.

Default None.

Remarks Register 0 is not allowed.



Function Specifies the number of positions to shift the register.

Format Absolute notation between 0 and 16.

Default None.

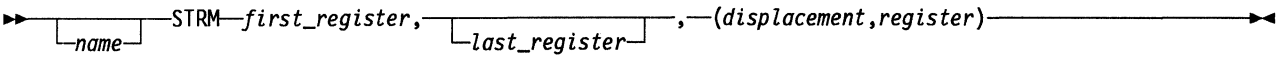
Remarks The contents of register 1(0) are destroyed if a count greater than 3 is specified.

STRM—Store a Series of Registers

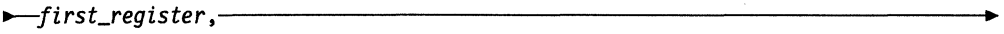
The STRM macro sequentially stores a specified series of registers, starting in a specified area of storage. The macro stores all 24 bits of each register into a 4-byte storage location.

The registers must be coded in ascending order. Register 0 is not allowed.

Syntax



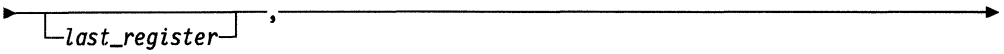
Parameters



Function Specifies the first register in a series to be stored.

Format Absolute register notation.

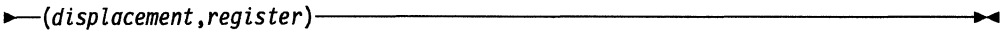
Default None.



Function Specifies the last register in a series to be stored.

Format Absolute register notation.

Default None.



Function Specifies the starting address of the storage location into which the first register in the series is to be stored.

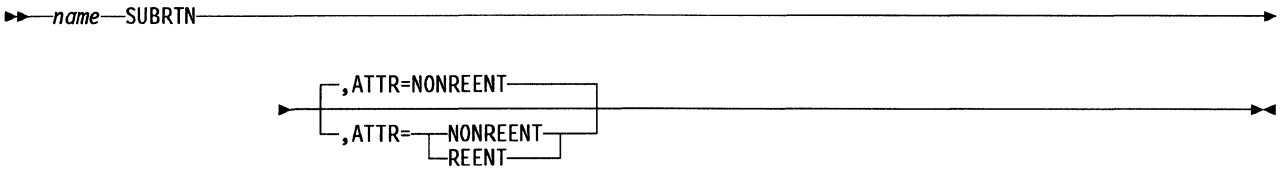
Format Base-displacement notation; the displacement must be a multiple of 4.

Default None.

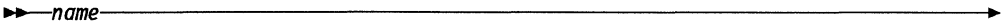
SUBRTN—Define a Subroutine Entry Point

The SUBRTN macro defines a subroutine entry point. This macro is used at the start of any subroutine that is to be invoked by the CALL macro.

Syntax



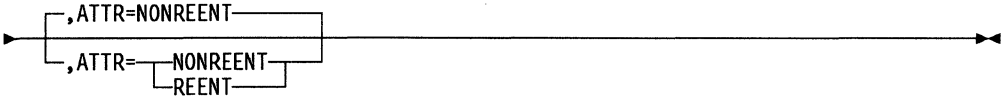
Parameters



Function Specifies the name of the subroutine and is identical to the subroutine name specified on the CALL macro for this routine.

Format Any symbol of 7 or fewer characters.

Default None.



Function Specifies whether the subroutine is reentrant or nonreentrant.

Format REENT or NONREENT.

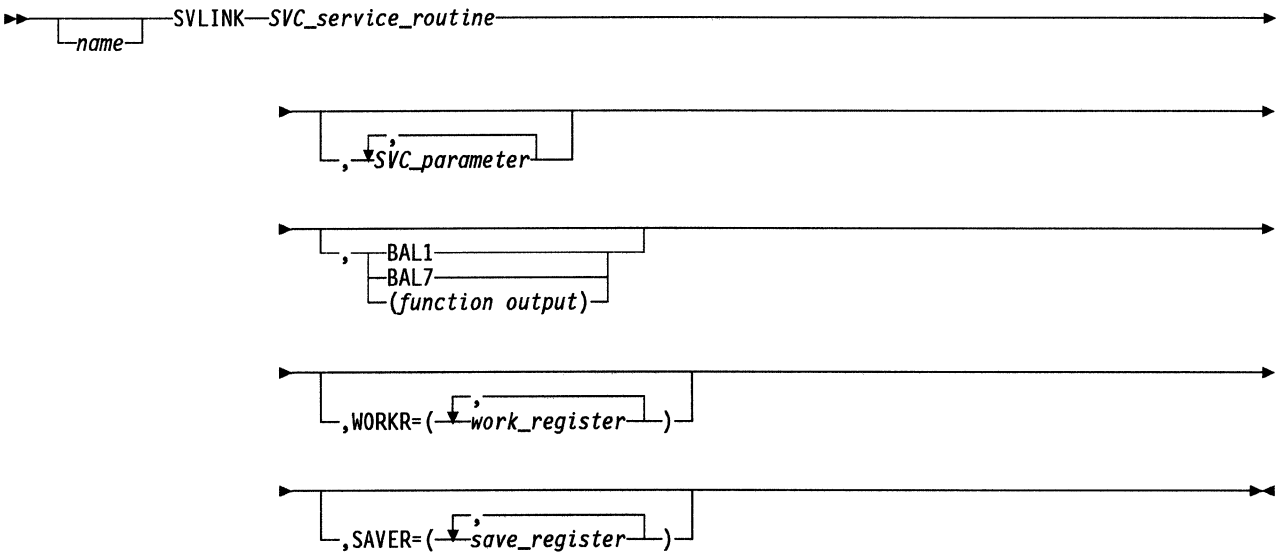
Default NONREENT.

SVLINK—Link to the SVC Service Routine

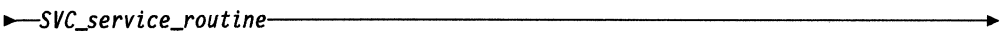
The SVLINK macro is used within a supervisor call (SVC) macro to link directly to the SVC service routine.

Register 0 is not allowed for register parameters.

Syntax



Parameters

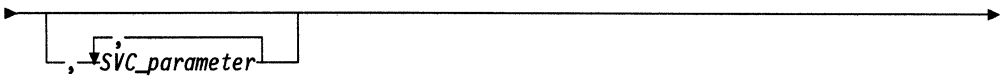


Function Specifies the address of the SVC service routine.

Format Label notation.

Default None.

Remarks On the first occurrence of an external SVC service routine label in a module, an EXTRN statement for that label is generated.

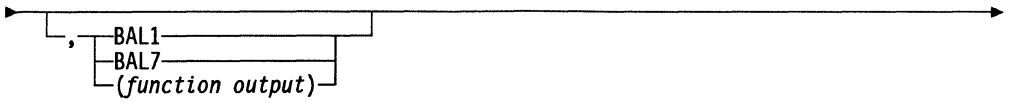


Function Specifies one or more 4-byte parameter values to be passed to the SVC service routine.

Format Any valid assembler expression.

Default No parameter values are passed.

Remarks If BAL1 or BAL7 is specified, the first parameter passed to the SVC service routine is 2 bytes long, rather than 4 bytes.



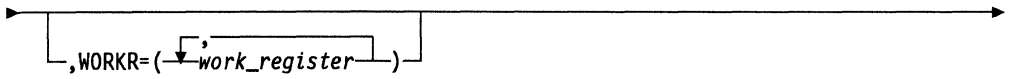
Function Specifies the registers to receive one or more output values from the SVC service routine. BAL1 and BAL7 specify that a more efficient linkage to the SVC service routine is generated, but these keywords preclude receiving any function-output values from the SVC service routine.

Format Register notation.

Default No function-output values are returned.

Remarks If you specify BAL1, the contents of register 1 are not preserved. If you specify BAL7, the contents of register 7 are not preserved.

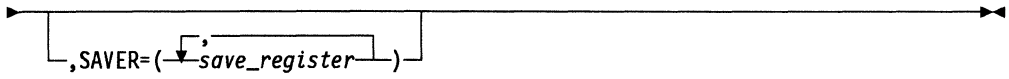
If you specify BAL1 or BAL7, the name field is optional; if you specify *function output*, you must specify the name field.



Function Specifies one or more work registers, the contents of which may be altered during execution of the macro. Specifying such registers makes execution of the SVC service routine more efficient. This keyword is used with the SAVER keyword to determine which registers do not have to be saved in the SVC service routine.

Format You can use any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default No work registers.



Function Specifies registers that are initially saved in the SVC service routine. This keyword is used with WORKR to determine which register saves in the SVC service routine can be skipped to provide better performance.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default All six registers are saved.

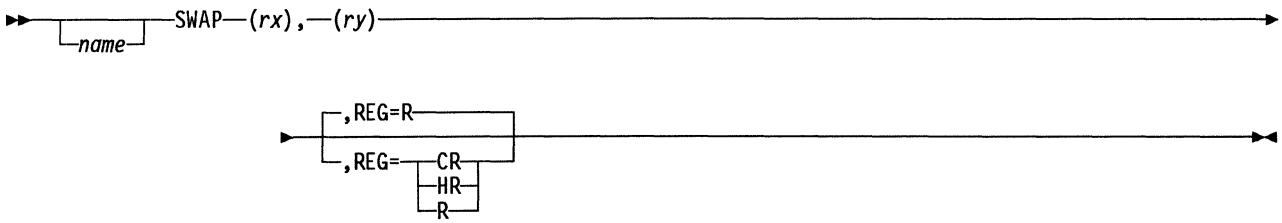
Remarks The registers specified in this keyword must match exactly those registers specified on the SAVE1 keyword of the SVC's SUPVNUC@ macro.

SWAP—Exchange the Contents of Two Registers

The SWAP macro generates a series of exclusive OR instructions that exchange the contents of two registers without using a third register or a storage location.

Register 0 is not allowed for register parameters.

Syntax

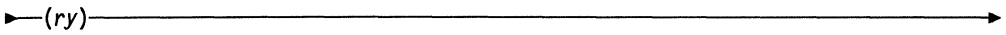


Parameters

Function Specifies the first register of the exchange.

Format Register notation if REG=HR or REG=R; byte register notation if REG=CR.

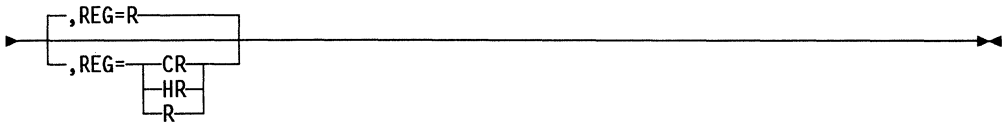
Default None.



Function Specifies the second register, the contents of which is to be exchanged.

Format Register notation if REG=HR or REG=R; byte register notation if REG=CR.

Default None.



Function Specifies whether 8-bit (CR) registers, 16-bit (HR) registers, or 22-bit (R) registers are to be exchanged.

Format CR, HR, or R.

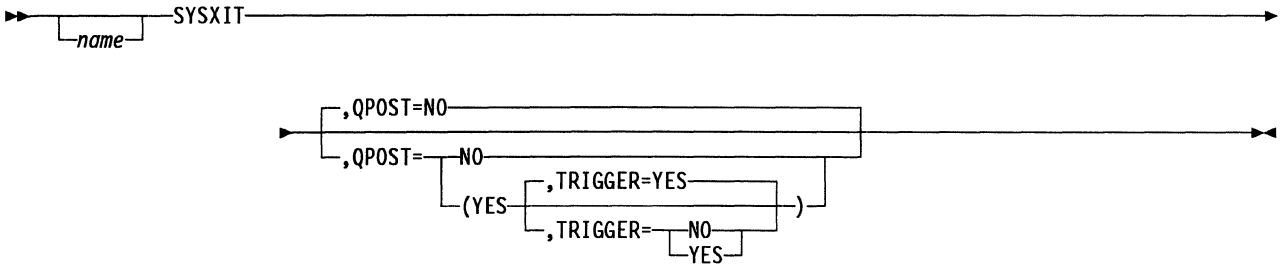
Default R.

SYSXIT—Return Control to Supervisor after an Executed Task

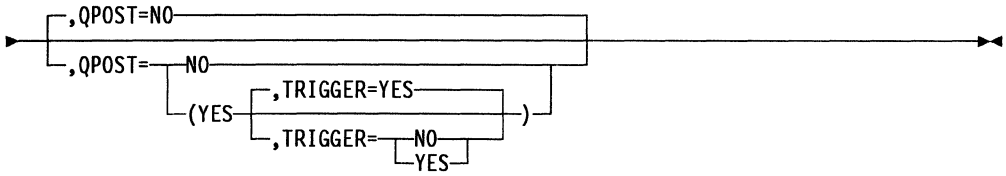
The SYSXIT macro returns control to the supervisor when the task has completed its execution.

For additional information on how the SYSXIT macro affects task states, see the section "Task Management" in *NCP and EP Reference*.

Syntax



Parameters



Function Specifies whether the issuing task's input or pseudo-input queue is to be reactivated. If YES, the TRIGGER keyword indicates whether the queue is to be triggered.

Format YES or NO.

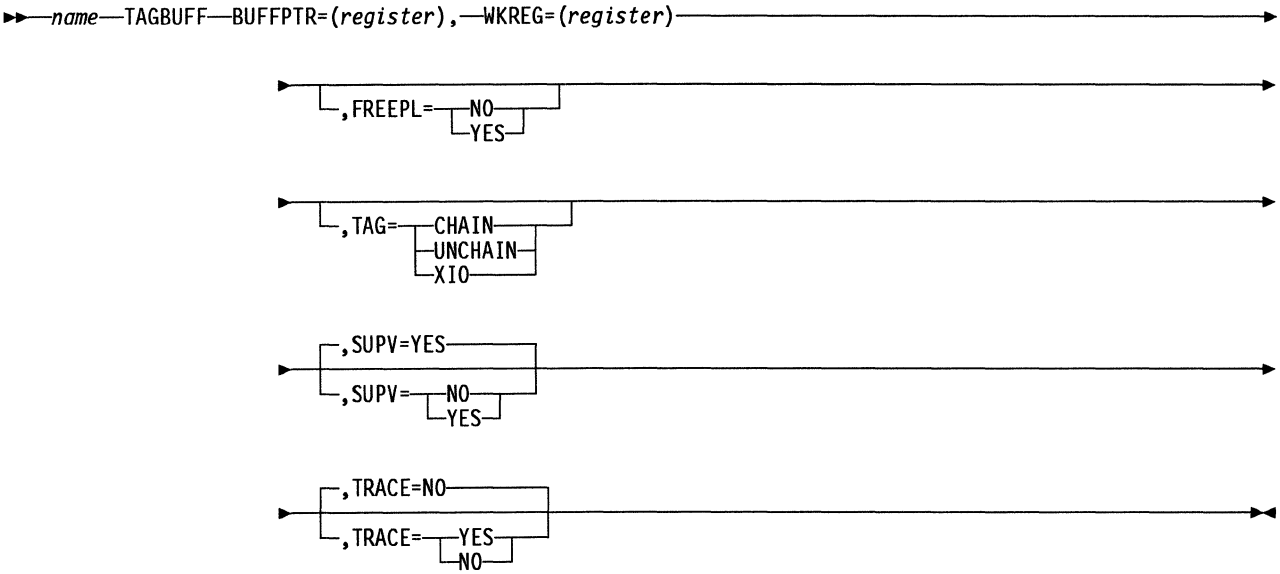
Default NO for QPOST; YES for TRIGGER.

TAGBUFF—Set the Buffer-Tag Field of a BH Control Block

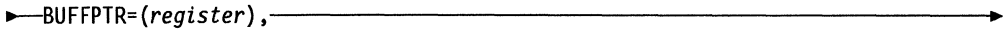
The TAGBUFF macro sets the buffer-tag field of the buffer prefix (BH control block) for a specified buffer with a specified status. The status indicates the buffer's usage. It is referred to when maintenance and debugging procedures require the history of the buffer's activity.

Register 0 is not allowed for register parameters.

Syntax



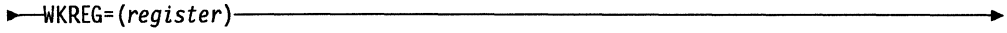
Parameters



Function Specifies an odd-numbered register containing the address of the buffer to be tagged.

Format Register notation.

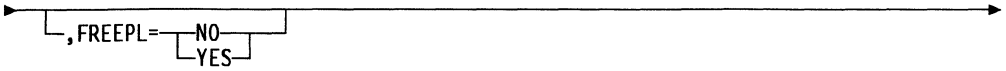
Default None.



Function Specifies a 1-byte odd-numbered work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.



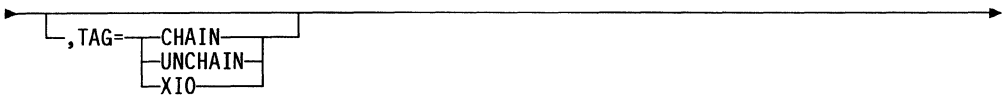
Function Specifies that the buffer is being released. FREEPL=YES specifies that the buffer is being released to the free-buffer pool. FREEPL=NO specifies that the buffer is leased from the free-buffer pool.

Format YES or NO.

Default None.

Remarks When FREEPL=NO, the buffer-is-not-in-free-buffer-pool status indicator is set on, and the remaining buffer-tag status indicators are set off.

You must code either TAG or FREEPL, but not both.

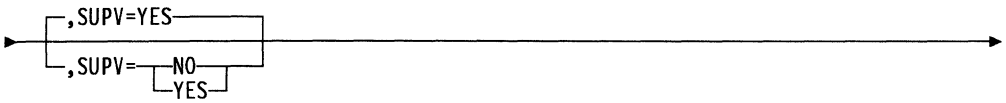


Function Specifies how buffer status is set. TAG=XIO indicates that the buffer-is-initialized-to-the-line-or-link status indicator is set on. TAG=CHAIN indicates that the buffer-is-chained status indicator is set on. TAG=UNCHAIN indicates that the buffer-is-unchained status indicator is set on.

Format XIO, CHAIN, or UNCHAIN.

Default None.

Remarks You must code either TAG or FREEPL, but not both.

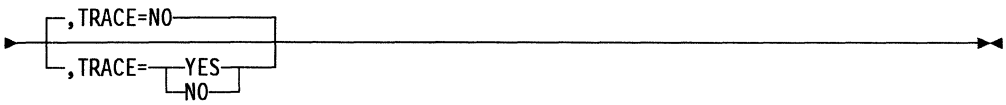


Function Specifies the level in which the issuer is running. SUPV=YES specifies that this macro is used in an interrupt level. SUPV=NO indicates that this macro is used in level 5.

Format YES or NO.

Default YES.

Remarks Inline code is always generated regardless of the value of SUPV.



Function Specifies whether the buffer address and location will be logged in the ABN diagnostic area if an abend occurs.

Format YES or NO.

Default NO .

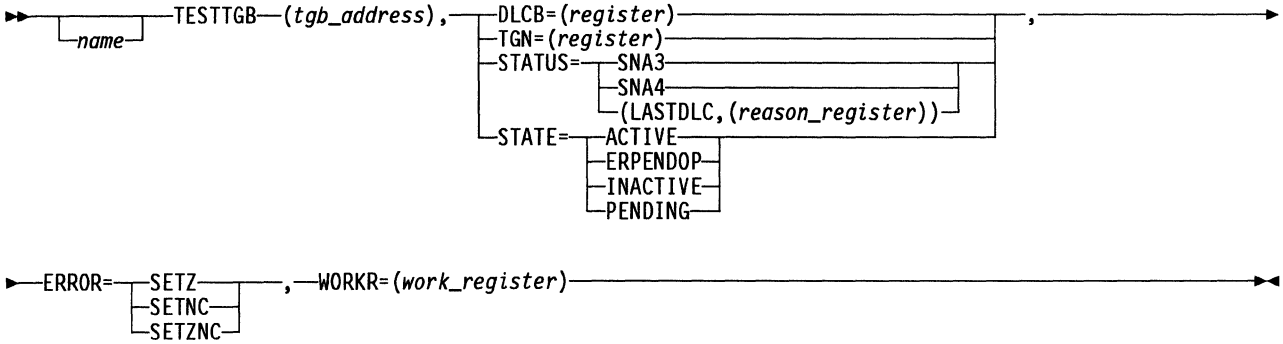
Remarks If YES is specified, dsect XCXTABN must be included in the assembly.

TESTTGB—Test States and Conditions of a TGB

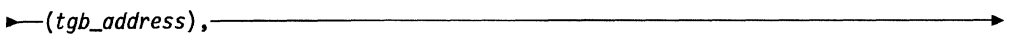
The TESTTGB macro checks specific states and conditions within a transmission group control block (TGB). It can also return the data link control (DLC) control block address currently associated with the TGB and the transmission group number (TGN) of the TGB.

Register 0 is not allowed for register parameters.

Syntax



Parameters



Function Specifies the register that contains the TGB under test.

Format Register notation.

Default None.

Remarks The register cannot be the same as the register specified on the WORKR, DLCB, TGN, or STATUS keywords.

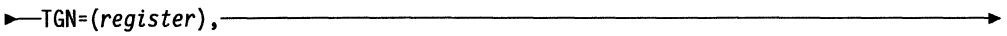
You must provide the TGB DSECT.

Function Specifies the return register for the data link control block address of the specified transmission group block. Returns a value of 0 if no DLC is currently associated with the TGB.

Format Register notation.

Default None.

Remarks The register cannot be the same as the TGB address register. You must specify exactly one of the following keywords: DLCB, TGN, STATUS, or STATE.

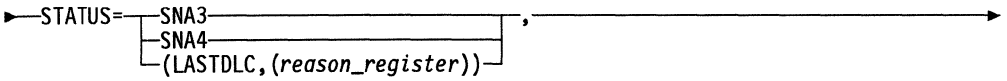


Function Specifies the register into which the TGN is to be returned.

Format The register must be a byte register.

Default None.

Remarks The register you specify cannot be the same as that specified for the TGB address or the DLCB keyword. You must specify exactly one of the following keywords: DLCB, TGN, STATUS, or STATE.

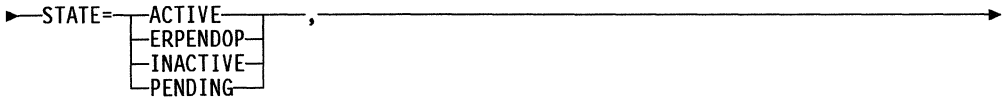


Function Specifies the status to be tested for. SNA3 and SNA4 indicate the level of the adjoining node data link control. An indicator value is returned in the reason register to give the reason the data link control became inoperative the last time.

Format SNA3, SNA4, or (LASTDLC,(*reason register*)). The reason register is a byte register.

Default None.

Remarks The register specified as the reason register cannot be the same as the register that points to the TGB under test. You must specify exactly one of the following keywords: DLCB, TGN, STATUS, or STATE.



Function Specifies the state to be tested for. ACTIVE means that the TGB is operational and able to transfer data between path control and data link control.

ERPENDOP means that the data link control is active and the explicit route manager has been notified, but its processing to become fully active has not been completed.

INACTIVE means that:

- The TGB is currently not associated with a data link control
- The explicit route control function for all explicit routes associated with the TGB is reset.

PENDING means that the TGB is not in the ACTIVE, INACTIVE, or ERPENDOP state.

Format ACTIVE, ERPENDOP, PENDING, or INACTIVE.

Default None.

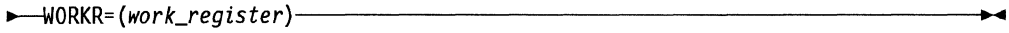
Remarks You must specify exactly one of the following keywords: DLCB, TGN, STATUS, or STATE.

Function Specifies whether the C latch, Z latch, or both are to indicate whether the TGB is in the specified state or level.

Format SETZNC, SETZ, or SETNC.

Default None.

Remarks If the TGB is not in the specified state or level, the Z latch is set to 1, and the C latch is set to 0. If the TGB is in the specified state or level, the Z latch is set to 0, and the C latch is set to 1.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format The register must be a byte register.

Default None.

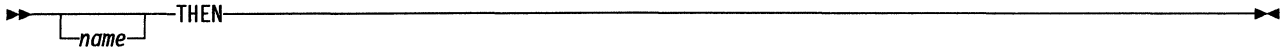
Remarks This register cannot be the same as the register specified for the TGB address or the LASTDLC, DLCB, or TGN keyword.

You do not have to code WORKR when STATUS=(LASTDLC,(*reason register*)), DLCB=(*register*), or TGN=(*register*).

THEN—Begin IF Macro True Condition Instructions

The THEN macro is used with the IF, ANDIF, ORIF, ELSE, and ENDIF macros to form an IF-THEN-ELSE program structure. It prefixes the instructions that are executed if the conditions of the IF macro are true.

Syntax

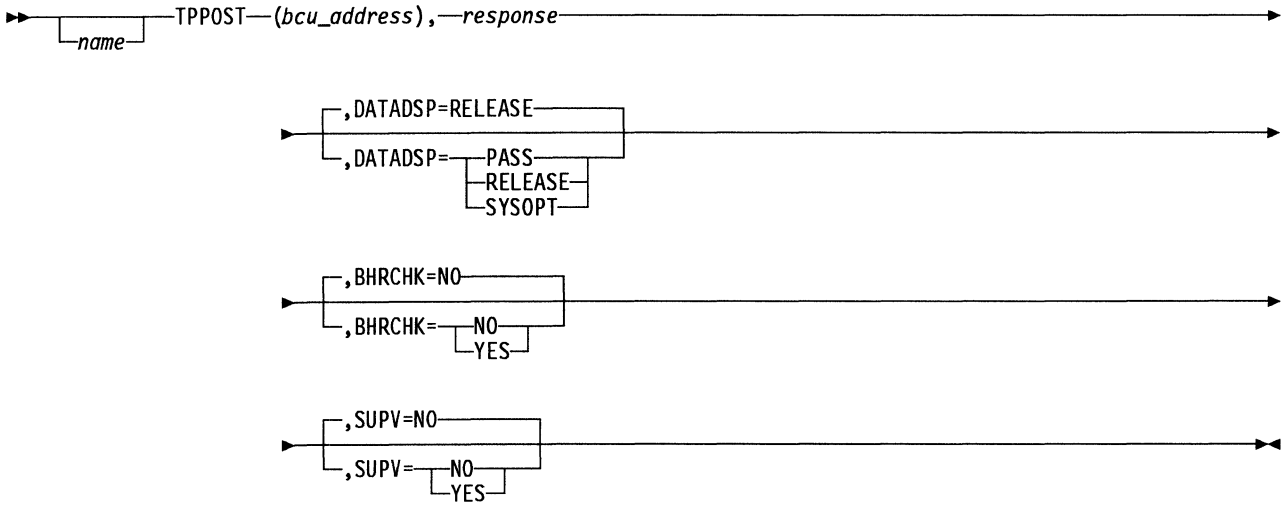


TPPOST—Discard a BCU after Processing

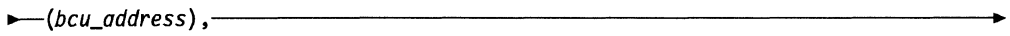
The TPPOST macro is used by any routine to discard a block control unit (BCU) after processing has been completed or after processing has been stopped because of an error condition.

Register 0 is not allowed for register parameters.

Syntax



Parameters



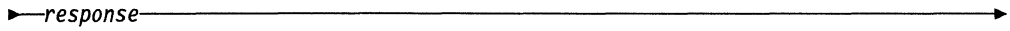
Function Specifies the register containing the address of the BCU that is to be discarded.

Format Register notation.

Default None.

Remarks If SUPV=YES, register 1 or register 6 is not allowed. Register 3 is standard.

The BCU must not reside on a queue and must contain a valid virtual route vector table (VVT) index.

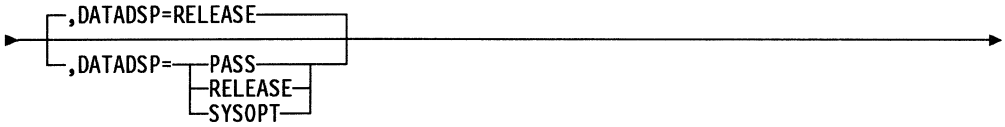


Function Specifies the bit configuration to be stored in the system response byte in the BTU.

Format Byte register notation or absolute notation (0 to 255).

Default None.

Remarks Register 1(0) is standard.



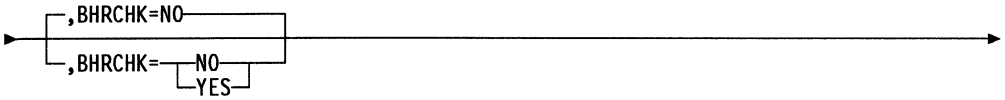
Function Specifies what is to be done with the buffers in the BCU. RELEASE indicates that the buffer is to be released.

PASS indicates that the buffer chain is not to be released.

SYSOPT specifies that the general communication byte (SYSFLG0) is to be examined to determine whether to release data.

Format RELEASE or PASS.

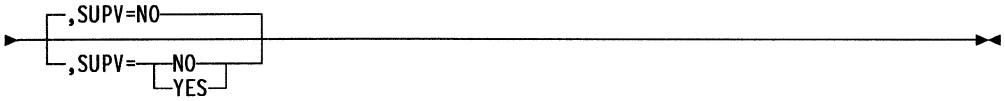
Default RELEASE.



Function Specifies whether point 3 block handler routine access is to be allowed for the BCU.

Format YES or NO.

Default NO.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

TRACEPIU—Trace PIUs

The TRACEPIU macro provides a path information unit (PIU) trace facility in NCP. An activate-line-trace PIU must be received to activate the PIU trace. The records are sent to the resource given in the origin address field (OAF) of this activate-line-trace PIU. The trace records generated by the TRACEPIU macro can be processed in the host by the Advanced Communications Function for the Trace Analysis Program (ACF/TAP).

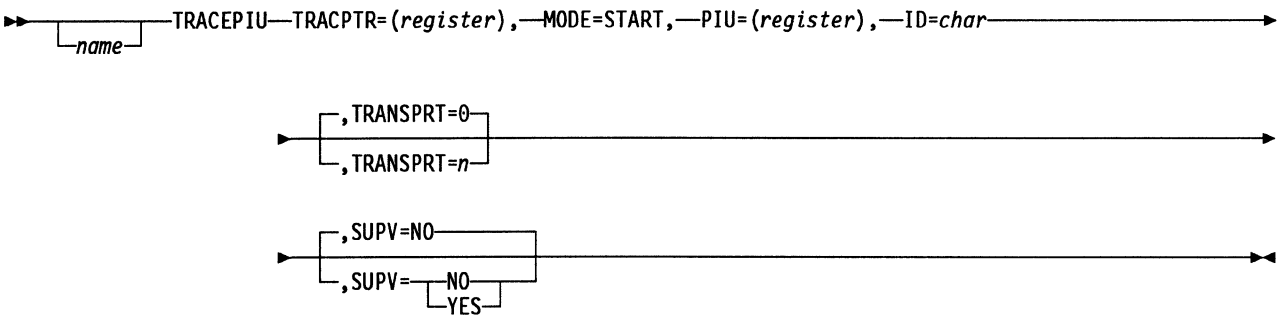
The TRACEPIU macro starts, continues, stops, or aborts a PIU trace. When the TRACEPIU macro is executed, its service routine (CXATRAC#) can do the following:

- Format the record trace data (RECTRD) PIU, including the ID specified
- Store the trace data into the RECTRD PIU
- Insert the proper ending code into the RECTRD PIU status field
- Enqueue the buffer that returns the RECTRD PIU to the SSCP
- Select data to be traced.

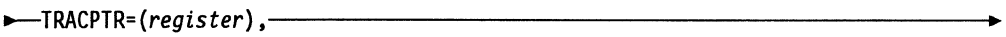
Register 0 is not allowed for register parameters.

Starting a PIU Trace

Syntax



Parameters



Function Specifies the register that points to a one-word storage location used by CXATRAC# to save the address of the buffer for this RECTRD PIU. Later, the data to be traced will be copied into this buffer. This storage word must be initialized to 0 before the execution of the TRACEPIU macro (MODE=START) and should be operated on only by the TRACEPIU macro. You must ensure that this storage word always points to the buffer; if not, data could be randomly stored in storage that contains the network control program.

Format Register notation.

Default None.

```
►─MODE=START,──────────────────────────────────────────────────────────►
```

Function Specifies the start of the PIU trace. Causes the CXATRAC# service routine to lease a buffer and initialize it to a RECTRD PIU format. CXATRAC# uses the pointer from the PIU=(*register*) keyword to address the activate line trace PIU to get information to use in initializing the RECTRD PIU format. The value specified in the ID keyword is used in this RECTRD PIU. CXATRAC# then stores the buffer address in the one-word storage location supplied by the TRACPTR keyword. If an error occurs, CXATRAC# sets an error return code in the condition latches (C=1 and Z=0) for SUPV=NO. For SUPV=YES, return codes set in register 3(1) are:

X'00' No error
X'F0' Sequence error; two consecutive starts
X'FF' Error condition.

Format START.

Default None.

```
►─PIU=(register),──────────────────────────────────────────────────────►
```

Function Specifies the register containing the address of the activate-line-trace PIU.

Format Register notation.

Default None.

Remarks When MODE=START is specified, the register must point to an activate-line-trace PIU request.

```
►─ID=char──────────────────────────────────────────────────────────────►
```

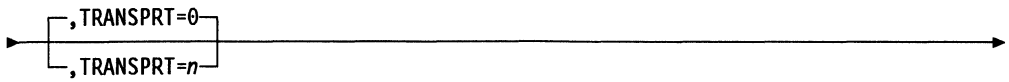
Function Specifies an alphabetical character or number to be used to identify the type of PIU trace to ACF/TAP. The character or number identifies records in the trace data output.

Format One alphabetical character (A to Z) or one digit (0 to 9).

Default None.

Remarks Required with the MODE=START keyword.

You can use the digits 0 to 9. The alphabetical characters are reserved for IBM use.



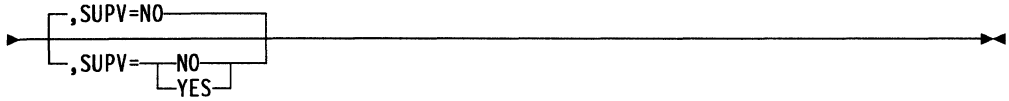
Function Specifies the number of PIUs to be traced and recorded in each RECTRD PIU before it is transmitted to the host.

Format Absolute values 0 to 7.

Default 0.

Remarks When TRNSPRT=0, one buffer is copied up to the length of the initial buffer. When TRNSPRT=1, one PIU is copied and additional buffers are leased if needed.

The count does not include skipped PIUs. For more information, see MODE=CONTINUE under "Continuing a PIU Trace."



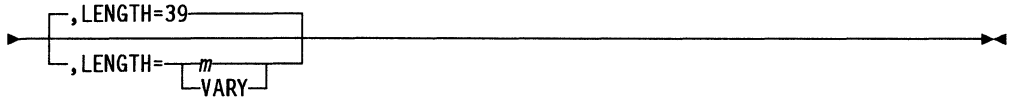
Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

Continuing a PIU Trace

Syntax



Parameters

```
►—TRACPTR=(register),————————————————————————————————►
```

Function Specifies the register pointing to a one-word storage location that is used by CXATRAC# to save the address of the buffer for this RECTRD PIU. The data to be traced is copied into this buffer. This storage word should be referred to only by the TRACEPIU macro. You must ensure that this storage word always points to the buffer; if not, data could be randomly stored in storage that contains the network control program. A unique storage fullword must be supplied for each concurrent trace.

Format Register notation.

Default None.

```
►—MODE=CONTINUE,—————————————————————————————————►
```

Function Specifies the transfer of the data to be traced. Causes the CXATRAC# service routine to transfer the data to be traced by copying the data from the PIU specified by the PIU keyword into the buffer pointed to by TRNSPRT. Copying begins with the first byte of the transmission header, which is stored in the next available byte of the RECTRD PIU. CXATRAC# transfers the number of bytes specified by LENGTH. If the TRNSPRT keyword is 0 when the trace is started, data is truncated if the buffer fills. If TRNSPRT is 1 to 7, additional buffers are leased if needed. If no buffers can be leased, no bytes are copied and the PIU is skipped. If an error occurs, CXATRAC# sets an error return code in the condition latches (C=1, Z=0). For SUPV=YES, return codes set in register 3(1) are:

X'00' No error
X'F0' Sequence error; continue issued before start
X'FF' Error condition.

Format CONTINUE.

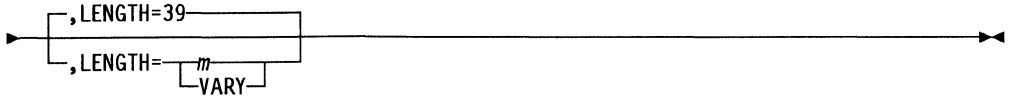
Default None.

```
►—PIU=(register)—————————————————————————————————►
```

Function Specifies the register containing the address of the PIU to be traced.

Format Register notation.

Default None.



Function Specifies the number of bytes to be traced from the PIU specified by the PIU keyword beginning with the first byte of the transmission header.

m may be any absolute value from 1 to 255. For VARY, the number of bytes depends on the type of SNA response header as follows:

(RH) Type	Fields Traced
CFC, NS, SC	TH, RH, and RU
FMD (character coded)	TH and RH
FMD (field format)	TH, RH, and 6 bytes of RU

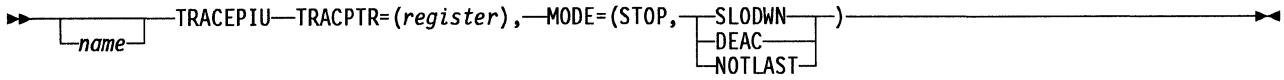
Format *m* (absolute value from 1 to 255). VARY.

Default 39.

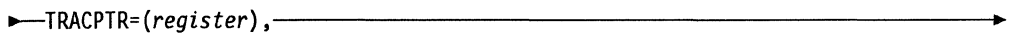
Remarks If TRNSPRT=0 for MODE=START, do not code VARY for MODE=CONTINUE.

Stopping a PIU Trace

Syntax



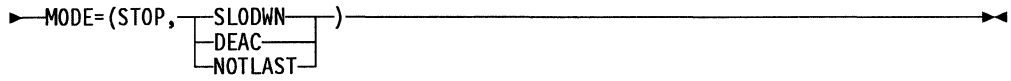
Parameters



Function Specifies the register pointing to a one-word storage location used by CXATRAC# to save the address of the buffer for this RECTRD PIU.

Format Register notation.

Default None.



Function Specifies the end of the PIU trace. Causes the CXATRAC# service routine to terminate the PIU trace.

The second keyword can be:

SLODWN	Last record, no buffers available in buffer pool
DEAC	Last record, deactivate received
NOTLAST	Not last record.

When you specify MODE=STOP, CXATRAC# places the proper code (X'01' for NOTLAST, X'02' for DEAC, and X'03' for SLODWN) into the RECTRD PIU status field and sends the RECTRD PIU to the SSCP. It then sets the address in the storage location pointed to by TRACPTR to 0 and exits. When you specify MODE=STOP, CXATRAC# ends the trace with notification to the SSCP. Use SLODWN or DEAC when termination of the trace is to be signaled to the host. NOTLAST stops the trace, but will not signal termination of the trace to the host.

Note: Use caution when PIU trace and normal line trace are active. Once VTAM* or TCAM receives a termination notice (DEAC or SLODWN), it stops recording trace records for the line. Succeeding record trace PIUs sent to the host for a line will be discarded.

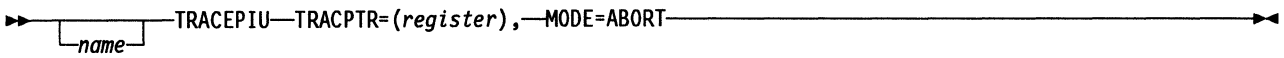
The return code set in register 3(1) is 0 (no error).

Format (STOP,SLODWN);(STOP,DEAC); or (STOP,NOTLAST).

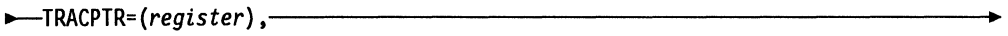
Default None.

Aborting a PIU Trace

Syntax



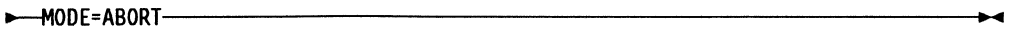
Parameters



Function Specifies the register pointing to a one-word storage location used by CXATRAC# to save the address of the buffer for this RECTRD PIU. The buffer pointed to by this fullword is to be released to the free buffer pool by this operation. Ensure that this storage word always points to the buffer.

Format Register notation.

Default None.



Function Specifies that the PIU trace be terminated. Causes the CXATRAC# service routine to end the trace operation without notifying the SSCP, release the buffer to the buffer pool, set the address in the storage location pointed to by TRACPTR to 0, and then exit. Use MODE=ABORT when the PIU trace can be started and the normal line trace cannot be started. Also use it during automatic network shutdown (ANS). Use this mode to terminate the trace when normal line trace activity will signal the host that line trace has been terminated. The return code set in register 3(1) is 0 (no error) for SUPV=YES.

Format ABORT.

Default None.

TRIGGER—Schedule a Task for Execution

The TRIGGER macro places a specified task in the pending state and schedules the task for execution.

Tasks are scheduled on a first-in, first-out (FIFO) basis with other tasks of the same dispatching priority. The hierarchy of priorities is, from highest to lowest: *appendage, immediate, productive, and nonproductive.*

A task is pending from the time it is triggered until it is dispatched. From the time it is dispatched until a QPOST macro is issued for it, a task is in the active state. The task changes to active and ready after the QPOST and remains this way until SYSXIT.

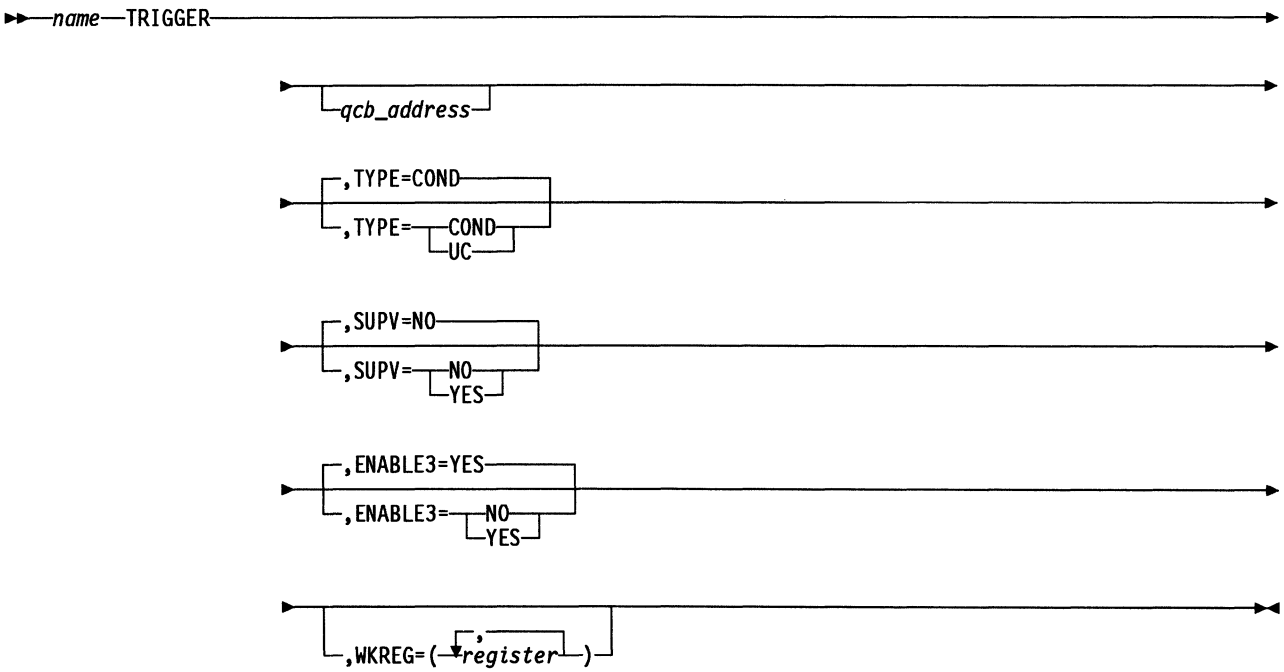
If you specify TYPE=COND, the supervisor ignores the trigger request unless a QPOST macro has been issued for the designated task. If you specify TYPE=UC, the supervisor will trigger the task regardless of whether a QPOST macro has been issued for the task.

A trigger request for a pending task is always ignored, regardless of the TYPE specification, because trigger requests are not remembered by the supervisor.

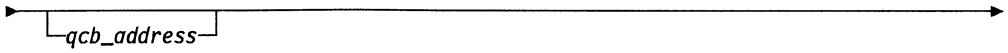
For additional information on specifying task priority and on how the TRIGGER macro affects task states, see "Task Management" in *NCP and EP Reference.*

Register 0 is not allowed for register parameters.

Syntax



Parameters



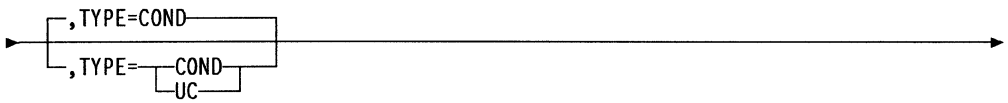
Function Specifies the address of the input or pseudo-input queue control block (QCB) governing the task to be activated.

Format Register or label notation.

Default If SUPV=YES, there is no default. If SUPV=NO, the QCB that activated the issuing task is triggered.

Remarks Register 1 is not allowed.

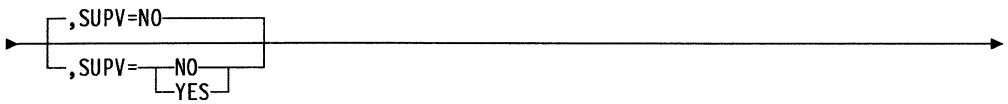
If SUPV=YES, register 2 is standard, and register 6 is not allowed.



Function Specifies whether the execution of this macro is to depend on the specified task being ready, that is, a QPOST macro having been issued for the task. Unit check (UC) causes the supervisor to trigger in any case.

Format UC or COND.

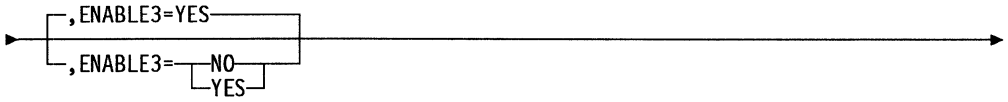
Default COND.



Function Specifies whether the macro is to issue a supervisor request to level 4 or is to link directly to the trigger routine.

Format YES or NO.

Default NO.



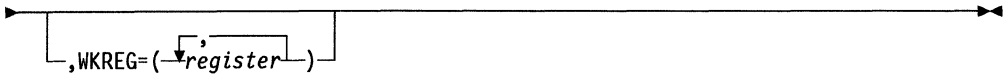
Function Specifies whether level 3 interrupts are to be enabled at the completion of the lease.

ENABLE3=YES allows both level 2 and level 3 interrupts to be enabled at the completion of the lease. ENABLE3=NO allows a level 4 routine to inhibit level 3 interrupts and to issue a lease. Only level 2 interrupts are enabled at the completion of the LEASE when ENABLE3=NO is coded.

Format YES or NO.

Default YES.

Remarks Use this parameter only when SUPV=YES.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

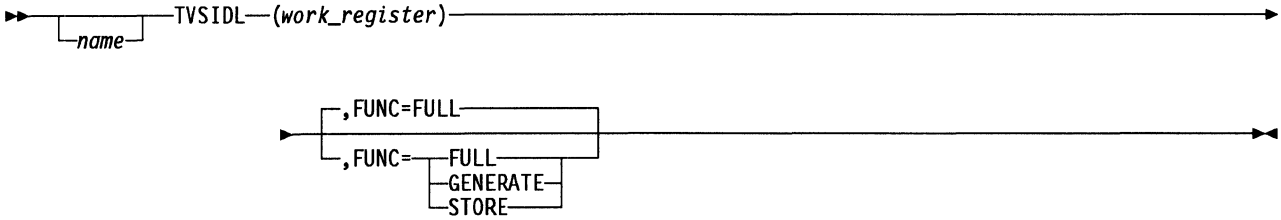
Default No work registers.

Remarks Register 6 is not allowed.

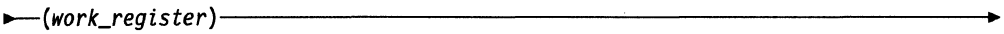
TVSIDL—Start an Idle Time-Out for a Line

The TVSIDL macro starts an idle time-out for a communication line. Character control block (CCB) addressability and the time value select (TVS) DSECT (XXCXTTVS) are required. This macro is intended for use by program level 2 or 3. When TVSIDL is used in program level 4 or 5, program level 2 or 3 may cause interference. Therefore, disable program level 3 if this macro is used in program level 4. Do not use it in program level 5.

Syntax



Parameters

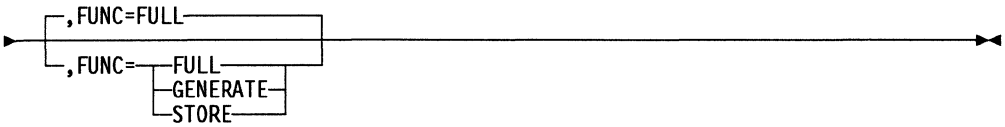


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Byte register notation.

Default None.

Remarks Register 0 is not allowed.



Function Manipulates the timer command. If FUNC=FULL, the timer command is generated and stored in the appropriate CCB field. If FUNC=GENERATE, the timer command is generated and placed in the work register. If FUNC=STORE, the contents of the work register are placed in the appropriate CCB field.

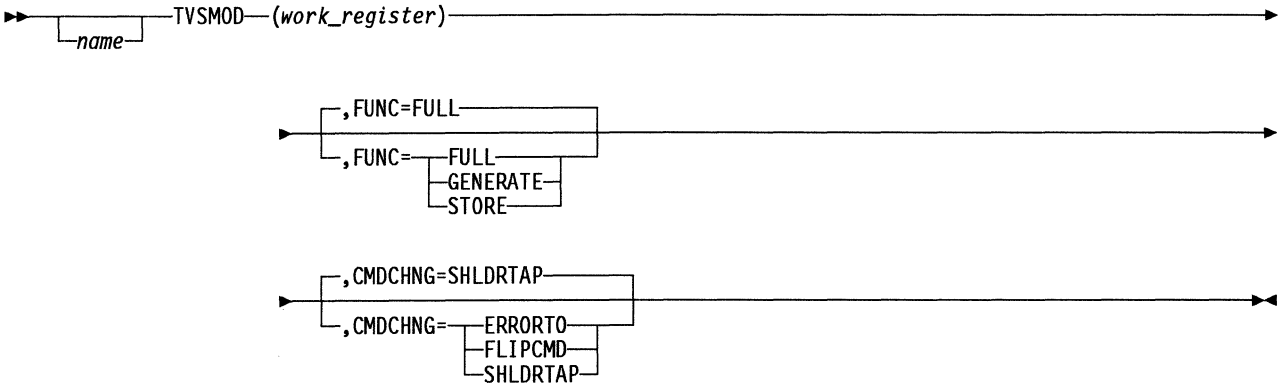
Format FULL, GENERATE, or STORE.

Default FULL.

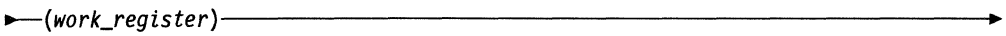
TVSMOD—Change the Current Time-Out Type

The TVSMOD macro changes the type of time-out in effect without changing its value. CCB addressability and the time value select (TVS) DSECT (XXCXTTVS) are required. This macro is intended for use by program level 2 or 3. When TVSMOD is used in program level 4 or 5, program level 2 or 3 may cause interference. Therefore, disable program level 3 if this macro is used in program level 4. Do not use it in program level 5.

Syntax



Parameters

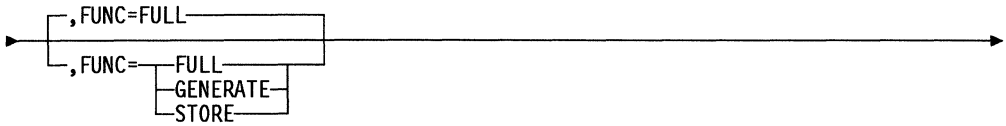


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Byte-register notation.

Default None.

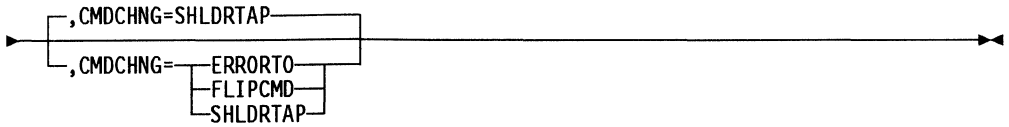
Remarks Register 0 is not allowed.



Function Manipulates the timer command. If FUNC=FULL, the timer command is generated and stored in the appropriate CCB field. If FUNC=GENERATE, the timer command is generated and placed in the work register. If FUNC=STORE, the contents of the work register are placed in the appropriate CCB field.

Format FULL, GENERATE, or STORE.

Default FULL.



Function Alters the type of time-out command in effect:

- CMDCHNG=ERRORTO initiates an error time-out.
- CMDCHNG=FLIPCMD toggles between an error time-out and a shoulder-tap time-out. If the previous time-out was an error time-out, the error time-out is changed to a shoulder-tap time-out. If CMDCHNG=FLIPCMD and the previous time-out was a shoulder-tap time-out, the shoulder-tap time-out is changed to an error time-out.
- CMDCHNG=SHLDRTAP initiates a shoulder-tap time-out.

Format ERRORTO, FLIPCMD, or SHLDRTAP.

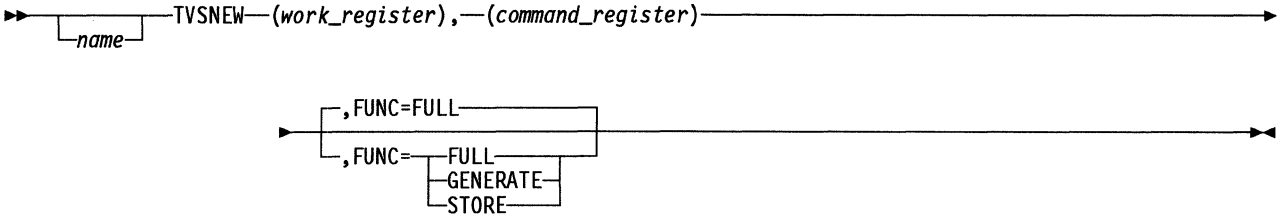
Default SHLDRTAP.

TVSNEW—Start a Time-Out for a Line

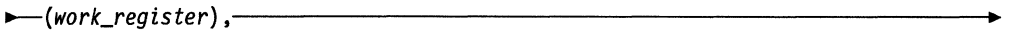
The TVSNEW macro starts a communication line time-out. The time-out command is contained in a register. CCB addressability and the time value select (TVS) DSECT (XXCXTTVS) are required. This macro is intended for use by program level 2 or 3. When TVSNEW is used in program level 4 or 5, program level 2 or 3 may cause interference. Therefore, disable program level 3 if this macro is used in program level 4. Do not use it in program level 5.

Register 0 is not allowed for register parameters.

Syntax



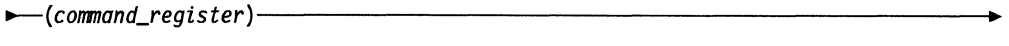
Parameters



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Byte register notation.

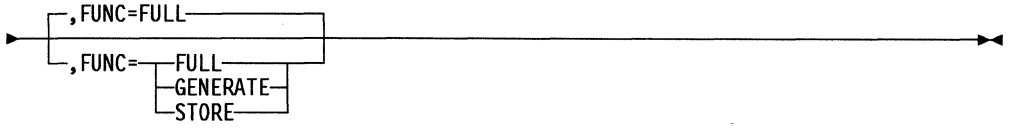
Default None.



Function Specifies the register containing the new time-out command.

Format Byte register notation.

Default None.



Function Manipulates the timer command. If FUNC=FULL, the timer command is generated and stored in the appropriate CCB field. If FUNC=GENERATE, the timer command is generated and placed in the work register. If FUNC=STORE, the contents of the work register are placed in the appropriate CCB field.

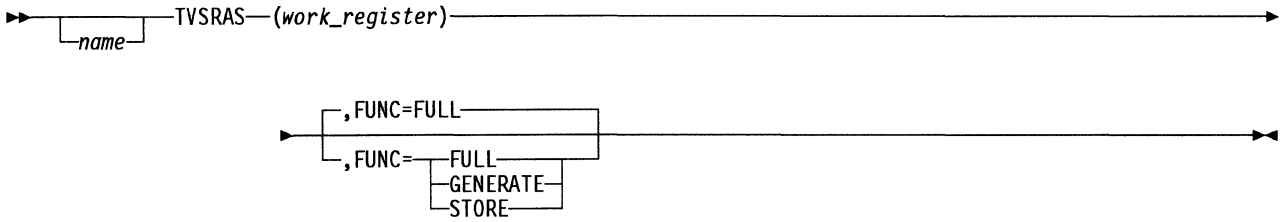
Format FULL, GENERATE, or STORE.

Default FULL.

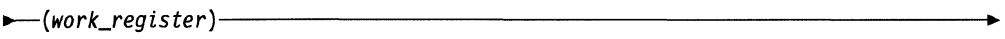
TVSRAS—Start an RAS Time-Out for a Line

The TVSRAS macro starts a reliability, availability, serviceability (RAS) time-out for a communication line. CCB addressability and the time value select (TVS) DSECT (XXCXTTVS) are required. This macro is intended for use by program level 2 or 3. When TVRAS is used in program level 4, program level 2 or 3 may cause interference. Therefore, disable program level 3 if this macro is used in program level 4. Do not use it in program level 5.

Syntax



Parameters

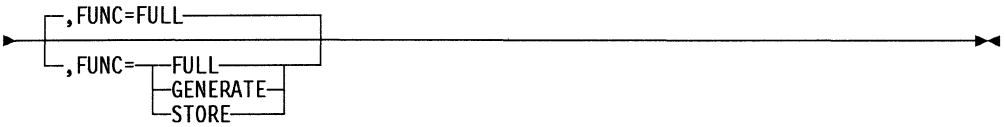


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Byte register notation.

Default None.

Remarks Register 0 is not allowed.



Function Manipulates the timer command. If FUNC=FULL, the timer command is generated and stored in the appropriate CCB field. If FUNC=GENERATE, the timer command is generated and placed in the work register. If FUNC=STORE, the contents of the work register are placed in the appropriate CCB field.

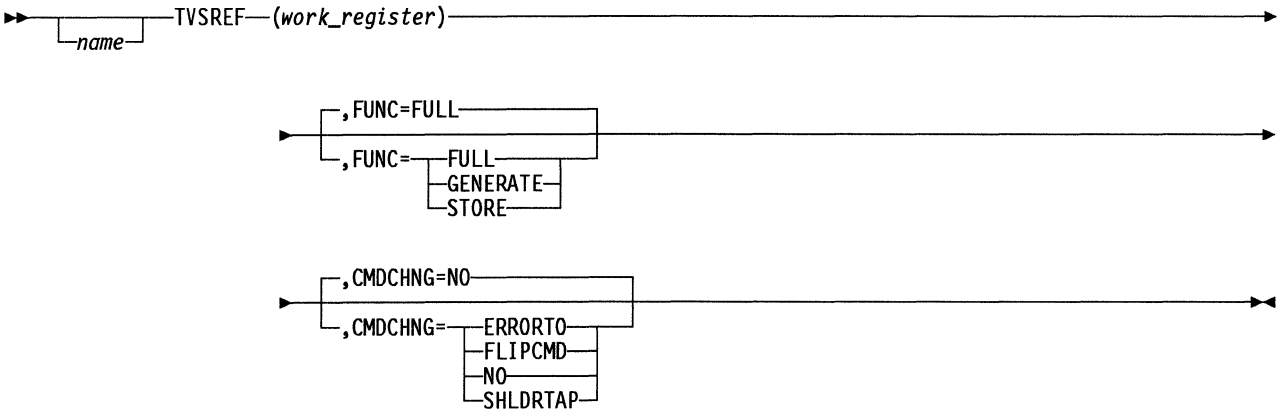
Format FULL, GENERATE, or STORE.

Default FULL.

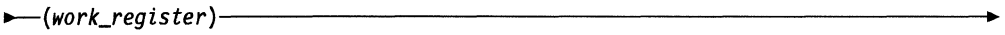
TVSREF—Refresh or Restart a Time-Out for a Line

The TVSREF macro refreshes or restarts a communication line time-out. CCB addressability, the timer value select (TVS) DSECT (XXCXTTVS), and the SYSEQU (XSYSEQU) are required. This macro is intended for use by program levels 2 and 3. When TVSREF is used in program level 4, program level 2 or 3 may cause interference. Therefore, disable program level 3 if this macro is used in program level 4. Do not use it in program level 5.

Syntax



Parameters

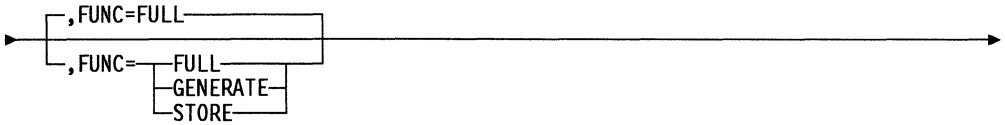


Function Specifies an odd-numbered work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

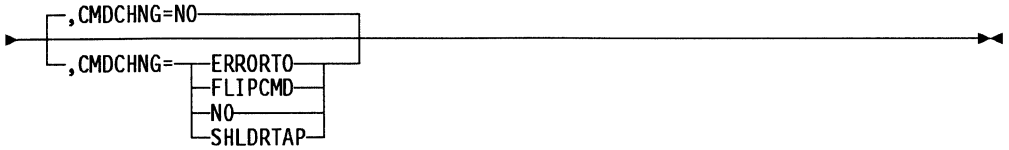
Remarks Register 0 is not allowed.



Function Manipulates the timer command. If FUNC=FULL, the timer command is generated and stored in the appropriate CCB field. If FUNC=GENERATE, the timer command is generated and placed in the specified work register. If FUNC=STORE, the contents of the work registers are placed in the appropriate CCB field.

Format FULL, GENERATE, or STORE.

Default FULL.



Function Alters the type of time-out command in effect:

- CMDCHNG=NO specifies no change.
- CMDCHNG=ERRORTO initiates an error time-out.
- CMDCHNG=FLIPCMD toggles between an error time-out and a shoulder-tap time-out. If the previous time-out was an error time-out, the error time-out is changed to a shoulder-tap time-out. If CMDCHNG=FLIPCMD and the previous time-out was a shoulder-tap time-out, the shoulder-tap time-out is changed to an error time-out.
- CMDCHNG=SHLDRTAP initiates a shoulder-tap time-out.

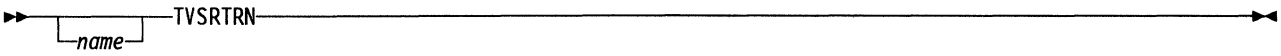
Format NO, ERRORTO, FLIPCMD, or SHLDRTAP.

Default NO.

TVSRTRN—Generate a Return Linkage from a Timer Routine

The TVSRTRN macro generates the proper return linkage to NCP from your timer routines. When control is returned to NCP, register 6 must contain the same address that was passed to your timer routines.

Syntax



TVSTIME—Start a Time-Out for a Line after a Time Interval

The TVSTIME macro starts a line time-out after a specified time interval. It does not use a time-out command. A timer service is available every 100 milliseconds for line adapters that use the high-resolution line timer (SDLC, BSC, and programmed resources). A timer service is available every 500 milliseconds for start-stop line adapters that use the low-resolution line timer. The TICKS keyword specifies the time interval, not as a time value, but as the number of times that the line timer function services the associated line adapter before completion of the time-out.

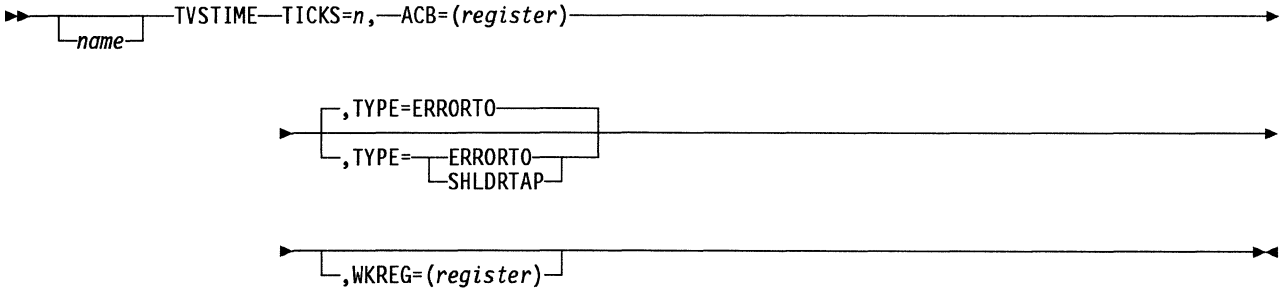
To ensure that a certain time-out interval has elapsed, the number of ticks that must be specified is the number of ticks for that time interval, plus 1. This is how the value used for normal NCP time-out commands is calculated. The reason for the additional tick is that the time until the first timer interrupt (tick) is indeterminate and may be very short; the time between that first tick and the last one is at least as long as the duration of the time-out. Thus, the first tick should not be counted as part of the time-out interval. Because the line timer runs asynchronously with NCP, a TICKS value of 1 results in a line time-out on the next available line-timer service (any time up to 100 milliseconds for a high-resolution line timer, or up to 500 milliseconds for a low-resolution line timer). Likewise, a TICKS value of 3 results in a line time-out on the third available line timer service (any time from 201 milliseconds up to 300 milliseconds for a high-resolution line timer, or from 1001 milliseconds up to 1500 milliseconds for a low-resolution line timer).

The type of time-out can be specified as either an error time-out or a shoulder-tap time-out. When the time-out is completed, the action that the line-timer function takes is the same as that taken when the TVSNEW macro starts an error or shoulder-tap time-out. The time-out is canceled if any of the TVS-type macros are executed for the same line adapter before the time-out is completed. If TVSREF is executed, the time-out expires the next time the line adapter is serviced by the timer function.

The TVSTIME macro must only be executed in a level 3 interrupt or in a level 4 interrupt while level 3 interrupts are disabled. It must not be executed in interrupt level 1 or 2, or in level 5.

Register 0 is not allowed for register parameters.

Syntax



Parameters

Function Specifies the number of times the timer function must service the line adapter before the time-out is completed.

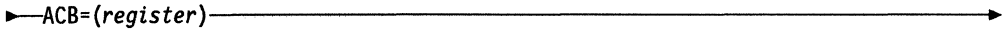
Format Halfword register or label notation.

Default None.

Remarks The maximum number of ticks is 16 383. The value 0 results in the maximum time-out (16 384 ticks). Only the low-order 14 bits in the specified register or in the specified value are used.

If a register is used, it cannot be the same as the register specified on any other keyword.

The WORKR keyword can be omitted for more efficient macro expansion if the TICKS keyword specifies an odd-numbered register and the contents of that register need not be preserved.

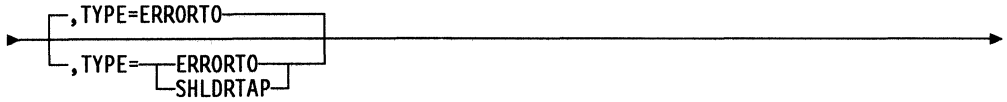


Function Specifies the register pointing to the adapter control block (ACB) for the line adapter on which the line time-out is to be performed.

Format Register notation.

Default None.

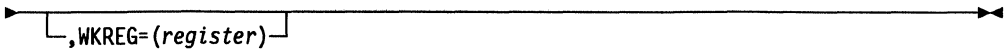
Remarks The register cannot be the same as the register specified on any other keyword.



Function Specifies that the type of time-out command to be performed is either an error time-out or a shoulder-tap time-out.

Format ERRORTO or SHLDRTAP.

Default ERRORTO.



Function Specifies an odd-numbered halfword work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks The register cannot be the same as the register specified on any other keyword.

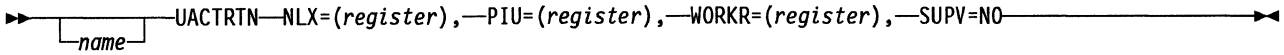
This keyword is not required when the TICKS keyword specifies an odd-numbered register and the contents of that register need not be preserved; otherwise, it is required.

UACTRTN—Pass Control to a User Accounting Exit Routine

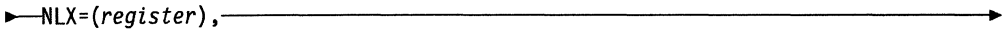
The UACTRTN macro generates a supervisor call (SVC) instruction. It uses as input a programmed resource logical unit extension (NLX) pointer and a path information unit (PIU) pointer. It determines if a user accounting exit routine was specified during NCP generation. If so, the service routine is invoked to pass control to the user accounting exit routine. Code a URETURN macro as the last statement of the routine to return control to level 5. If a user routine does not exist, the service routine is not invoked.

Register 0 is not allowed for register parameters.

Syntax



Parameters



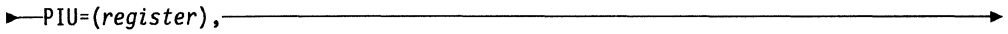
Function Specifies a register pointing to the NLX.

Format Register notation.

Default None.

Remarks The register used must not be the same as any other input register used in this macro.

Neither register 1 nor register 6 is allowed.



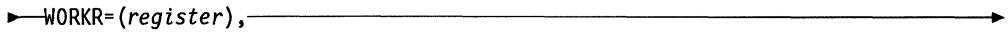
Function Specifies either a register that contains the pointer to the PIU or 0.

Format Absolute register notation.

Default None.

Remarks The register specified must not be the same as any other input register used in this macro. Neither register 1 nor register 6 is allowed.

When the value of the register is 0, the session will be deactivated.



Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Absolute register notation.

Default None.

Remarks The register specified must not be the same as any other input register used in this macro. Neither register 1 nor register 6 is allowed.

Function Specifies that the issuer is running in level 5.

Format NO.

Default None.

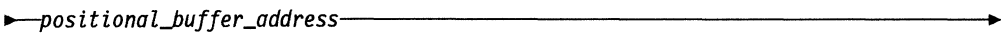
UNCHAIN—Detach a Buffer from a Buffer Chain

The UNCHAIN macro detaches a specified buffer from an existing buffer chain.

Register 0 is not allowed for register parameters.

Syntax

Parameters

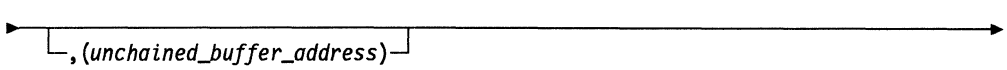


Function Specifies the register containing the buffer before the buffer to be unchained. This buffer can be any member of the chain except the last buffer.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed.



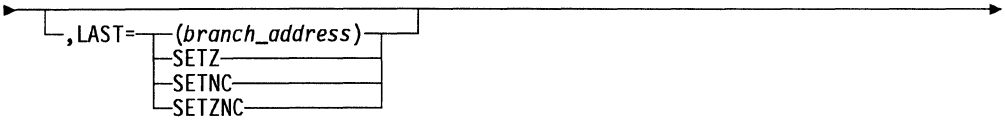
Function Specifies a register into which the supervisor places the address of the unchained buffer.

Format Register notation.

Default If SUPV=NO, the unchained buffer address is returned in the positional buffer address register. If SUPV=YES, there is no default.

Remarks If the positional buffer is last in the chain, the content of this register is set to 0.

Register 4 is standard. Register 1 is not allowed. If SUPV=YES, register 6 is not allowed.



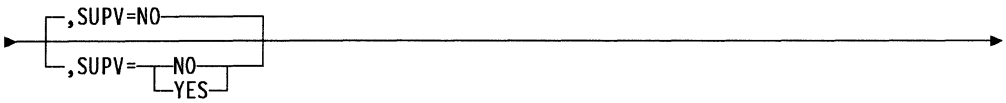
Function Either specifies the address to be given control, or specifies whether the C latch, Z latch, or both are to indicate whether the positional buffer is the last buffer in the element.

Format Register notation, label notation, SETZNC, SETZ, or SETNC.

Default If SUPV=NO, the next instruction after the macro is given control. If SUPV=YES, there is no default value.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed.

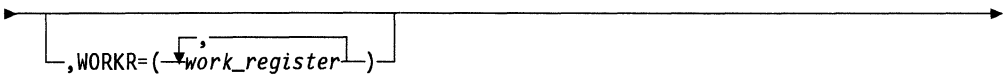
For SETZNC (Z latch and C latch) or SETZ, the Z latch is set to 1 if the buffer is the last buffer in the element or to 0 if is is not. For SETZNC or SETNC, the C latch is set to 0 if it is the last buffer in the element or to 1 if it is not.



Function Specifies whether to generate inline code or to issue a supervisor request.

Format YES or NO.

Default NO.

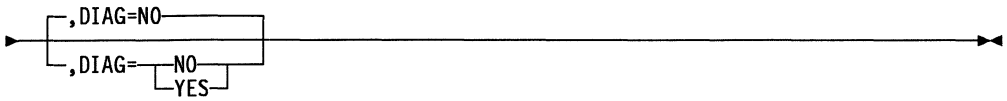


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation. You can use any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default No work registers; the contents of all registers are preserved.

Remarks This keyword is valid only if SUPV=YES.



Function Specifies whether the buffer address and location will be logged in the ABN diagnostic area if an abend occurs.

Format YES or NO

Default NO

Remarks This keyword is valid only if SUPV=YES is coded. If YES is specified, the assembly must include dsect XCXTABN.

UPARMS—Access Fields in the NIX, NLX, NLB, or PIU

The UPARMS macro executes inline. Its function is to provide access to various fields in the network interconnection extension (NIX), programmed resource logical unit block extension (NLX), programmed resource logical unit block (NLB), and path information unit (PIU). It also provides an interface to the GETIME macro for a time stamp. Up to five outputs may be requested during one invocation of the UPARMS macro. Registers 1 and 7 are work registers, and their contents are destroyed.

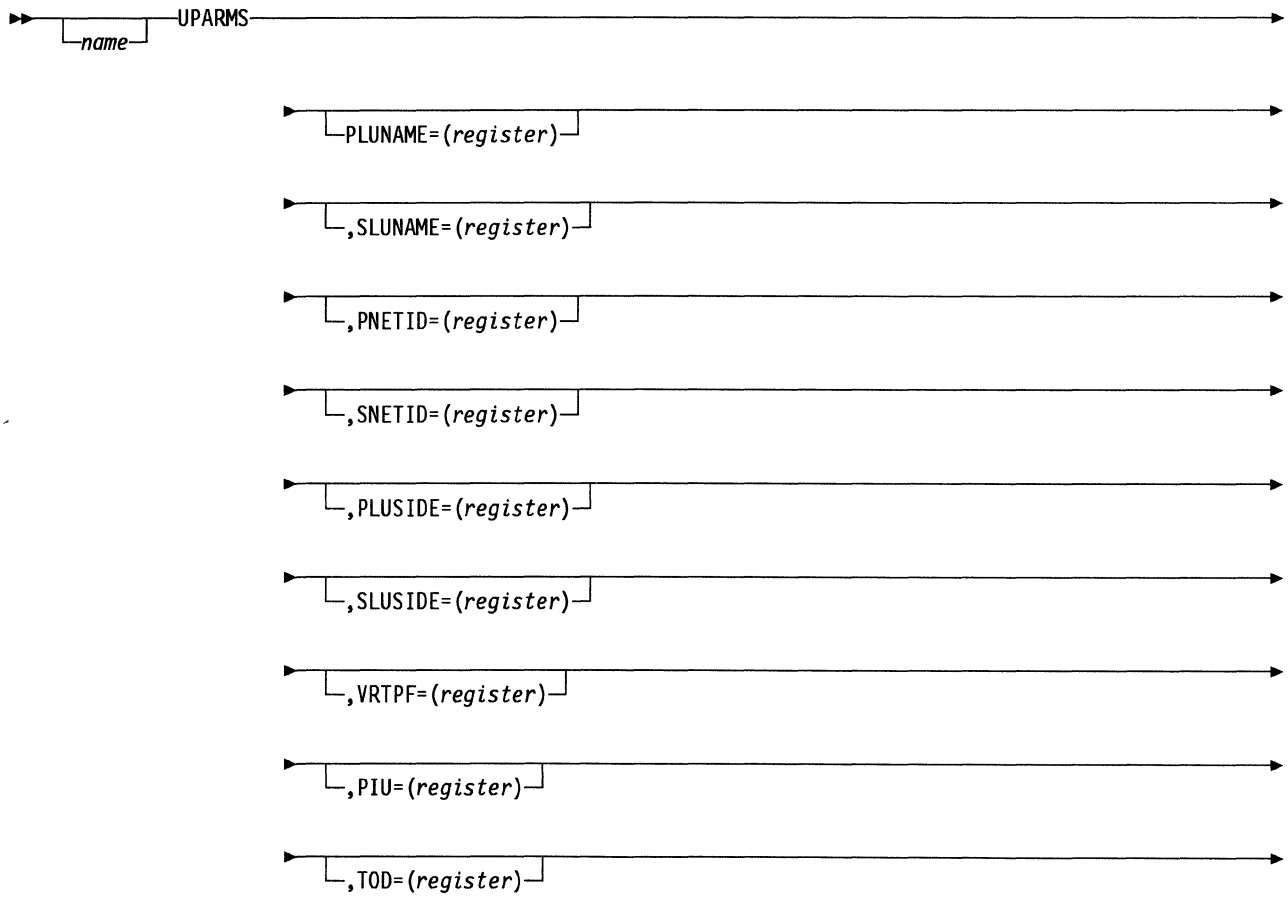
Register 0 is not allowed for register parameters.

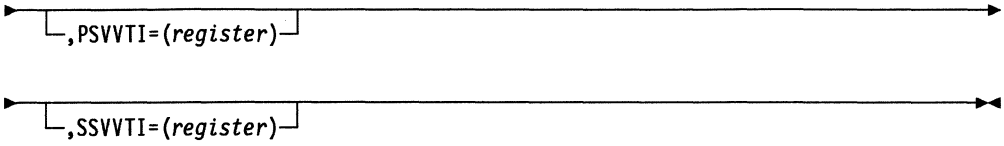
Note: This is an external macro to be used with the gateway accounting exit.

You must include the following DSECTs in any module that calls the UPARMS macro.

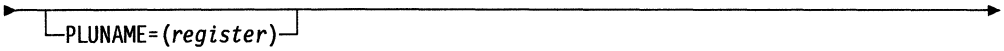
```
XCXTEQU DIGITS=NO
XCXTNIB
XCXTNIX
XCXTNLB
XCXTNLX
XCXTUIC
XSYSABNS
XSYSSEQU
XCXTRCB
XCXTPIU TYPE=4,
  SCRUEQU=(ACTCDRM,REQACT,NOTIFY)
```

Syntax





Parameters

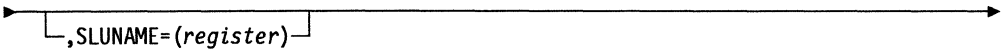


Function Specifies the register pointing to the primary logical unit (PLU) name.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

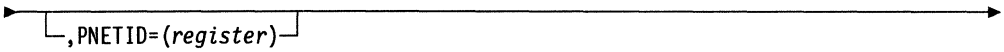


Function Specifies the register pointing to the secondary logical unit (SLU) name.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

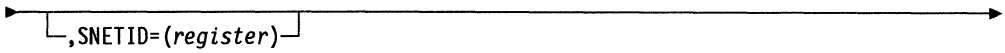


Function Specifies the register pointing to the network ID of the PLU.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

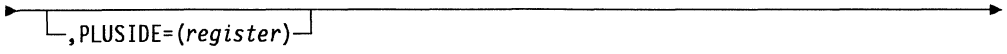


Function Specifies the register pointing to the network ID of the SLU.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.



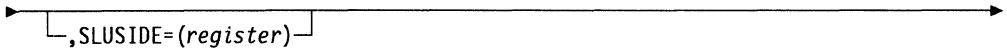
Function Specifies the registers containing the real and alias addresses of the PLU, where *realreg* is the register to contain the real address, and *aliasreg* is the register to contain the alias address.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

The address contained in the alias register is the address of a PLU in another network whose NETID pointer is specified by the PNETID keyword.



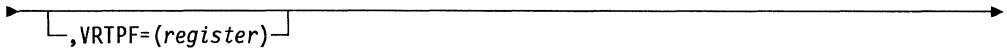
Function Specifies the registers containing the real and alias addresses of the SLU, where *realreg* is the register to contain the real address, and *aliasreg* is the register to contain the alias address.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

The address in the alias register is the address of an SLU in another network whose NETID pointer is specified by the SNETID keyword.

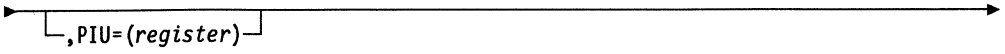


Function Specifies the byte register containing the VR-TPF field from the PIU (byte 3 of the transmission header). The byte register will contain a return value in which bits 0 to 3 are the virtual route number and bits 6 to 7 are the transmission priority field (TPF).

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

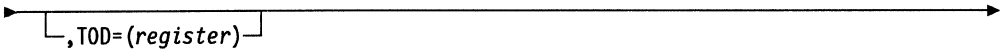


Function Specifies the register containing a pointer to the PIU.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

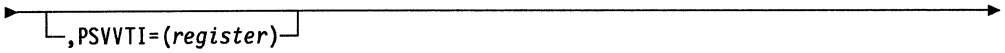


Function Contains a pointer to the place where the time stamp is to be put.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.



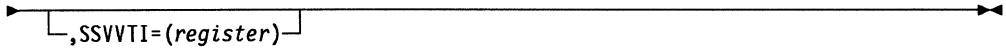
Function Specifies the register returning the virtual route vector table index (VVTI) from the NLX's resource connection block (RCB) on the PLU side of the transform.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

If the PLU side NLX is not attached to a virtual route, the VVT index returned will be not allowed (0).



Function Specifies the register returning the VVTI from the NLX's RCB on the SLU side of the transform.

Format Register notation.

Default None.

Remarks Neither register 1 nor register 7 is allowed. The register must not be the same as that specified for any other keyword.

If the PLU side NLX is not attached to a virtual route, the VVTI returned will be not allowed (0).

URETURN—Return Control from a User Accounting Exit Routine

The URETURN macro generates a supervisor call (SVC) instruction. Use it to pass control back to NCP level 5 from your accounting exit routine.

Note: This is an external macro to be used with the gateway accounting exit.

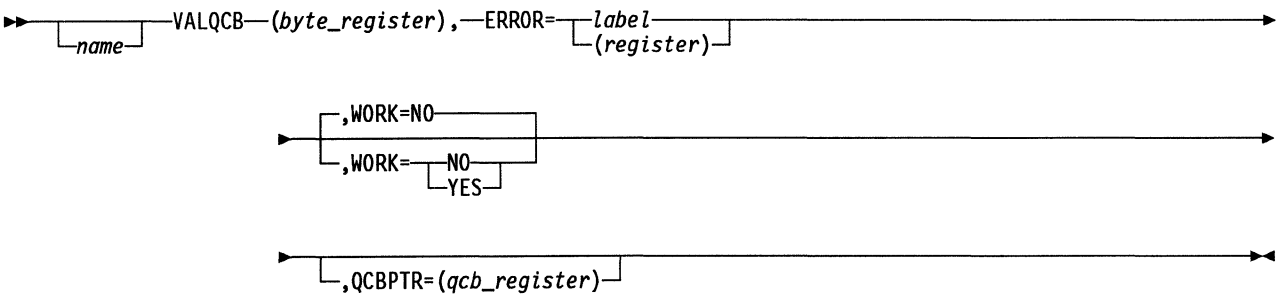
Syntax

VALQCB—Validate a QCB

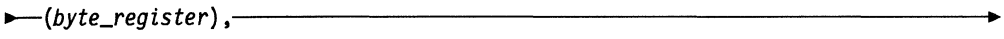
The VALQCB macro generates inline code that examines the identification bits of a specified control block and generates a branch to an error address if the control block is not a queue control block (QCB). If a QCB is not specified, supply the QCB DSECT (XXQCB) and establish addressability to it.

Register 0 is not allowed for register parameters.

Syntax



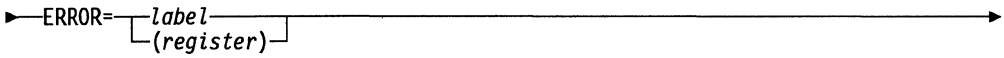
Parameters



Function Specifies a work register byte.

Format Byte register notation.

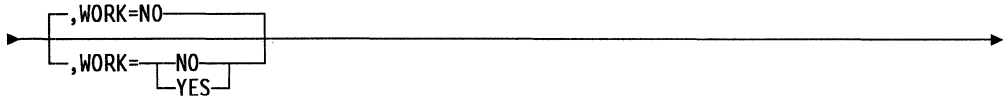
Default The next instruction after the macro expansion is given control.



Function Specifies where to branch if the control block is not a QCB, or if it is a work QCB and WORK=NO is specified.

Format Register or label notation.

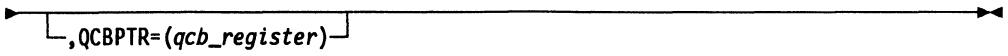
Default None.



Function Specifies whether the system work QCBs are to be accepted as valid.

Format YES or NO.

Default NO.



Function Specifies the register pointing to the QCB.

Format Register notation.

Default None.

Remarks If you do not specify a QCB pointer, you must include the QCB DSECT and have addressability to it.

VRACT—Activate an NCP-External Virtual Route

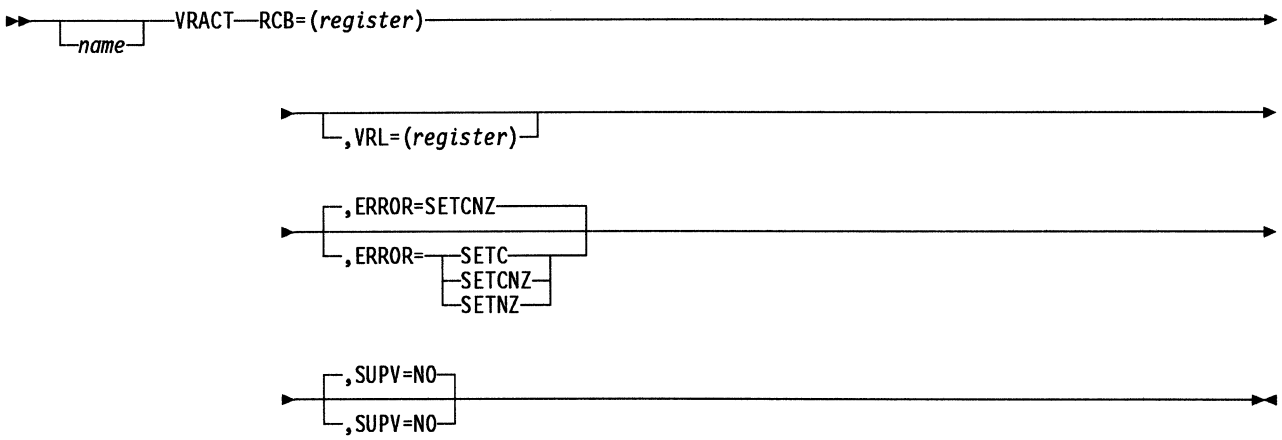
The VRACT macro requests the virtual route manager to start a virtual route external to NCP. The macro is passed either the address of a resource connection block (RCB) and a separate virtual route activation work list or the address of an RCB that already points to a virtual route activation work list. When activation is completed, the task associated with the RCB is triggered, and a return code is put in the virtual route work list. The virtual route vector table (VVT) index of the virtual route that was started is placed in the RCB. See the NVRID and VRACTCK macro descriptions for more information.

You do not need to use an ATTACHVR macro if the VRACT macro operation is successful. An ATTACHVR macro only attaches a virtual route that has been previously activated. The VRACT macro activates a virtual route and automatically attaches the virtual route. If the VRACT macro operation is unsuccessful, your code must either retry the operation or go on to the next procedure.

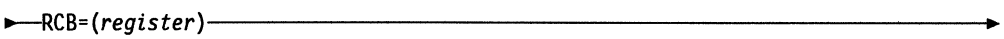
The VRACT macro generates a supervisor call (SVC) instruction.

Register 0 is not allowed for register parameters.

Syntax



Parameters

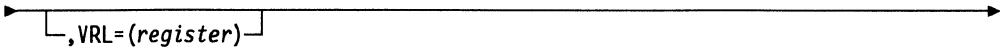


Function Specifies the register pointing to your RCB.

Format Absolute register notation.

Default None.

Remarks Neither register 1 nor register 6 is allowed.

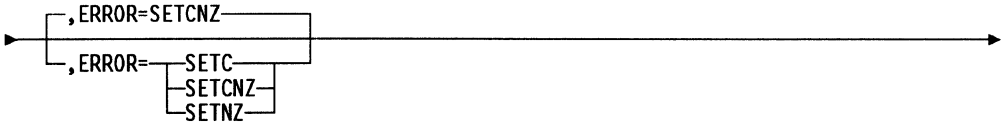


Function Specifies the register pointing to the separate virtual route activation work list.

Format Register notation.

Default None.

Remarks If this keyword is coded, the VRL pointer will be stored in the RCB.

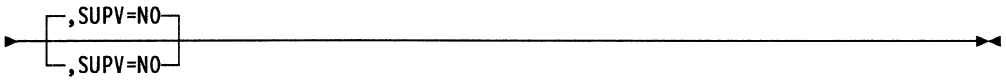


Function Specifies whether the C latch, Z latch, or both are to indicate whether the virtual route activation was initiated.

Format SETC, SETNZ, or SETCNZ.

Default SETCNZ.

Remarks If the virtual route to be activated is already active, the C latch is set on, and the Z latch is set off. If the virtual route to be activated is not active, the C latch is set off, and the Z latch is set on.



Function Specifies that the issuer is running in level 5.

Format NO.

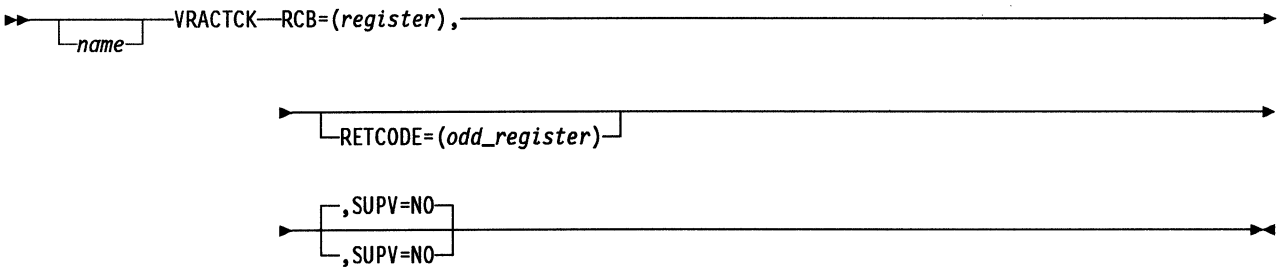
Default NO.

VRACTCK—Check the Activation of a Virtual Route

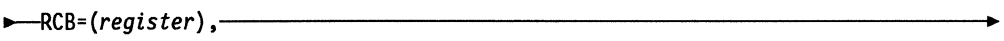
The VRACTCK macro indicates whether activation of a virtual route is successful. If activation is successful, the return code register returns the value X'00'. If activation is not successful, the register returns X'FF'. Any other value returned is a virtual route activation failure code.

Register 0 is not allowed for register parameters.

Syntax



Parameters

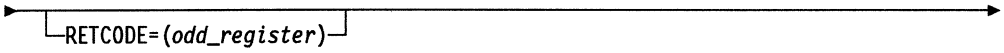


Function Specifies a register pointing to resource connection block (RCB).

Format Absolute register notation.

Default None.

Remarks The RCB register is not preserved.



Function Specifies an odd-numbered register, the low byte of which will contain the return code. The high byte will be used for work.

Format Absolute register notation.

Default None.

Remarks The register specified for RETCODE must not be the same as that specified for RCB. The possible return codes are:

X'00'	Virtual route activation was successful.
X'01'	No vector mapping was specified.
X'03'	No virtual route resource is available.
X'04'	No explicit route is operative.
X'05'	No explicit route can be activated.
X'06'	No explicit route can be activated.
X'FF'	Virtual route activation is not complete.

Function Specifies that the issuer is running in level 5.

Format NO.

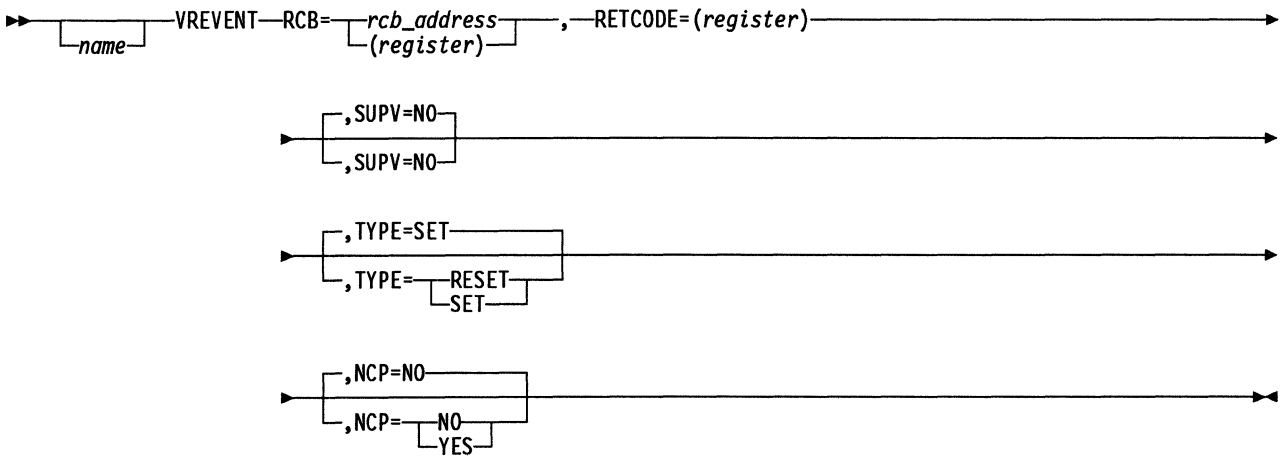
Default NO.

VREVENT—Monitor a Virtual Route Exiting the Held State

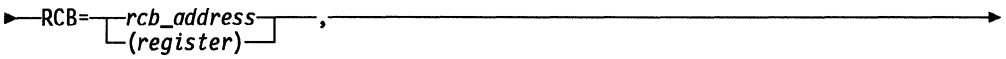
The VREVENT macro allows monitoring of a virtual route exiting the *held* state. The queue control block (QCB) pointer contained in the specified event control block triggers an awaiting task when the virtual route exits. This macro cannot be used if the event control block in the resource connection block (RCB) is in use. Register 6 must point to a save area that can be overwritten.

Register 0 is not allowed for register parameters.

Syntax



Parameters

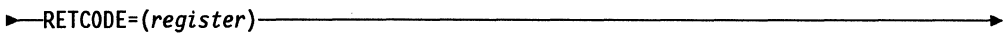


Function Specifies the RCB associated with an NCP resource and the virtual route for the event.

Format Register or label notation.

Default None.

Remarks Use VREVENT only with the virtual route associated with this RCB. The register specified must not be the same as that specified for RETCODE. Register 6 is not allowed.



Function Specifies the register in which a return code is returned.

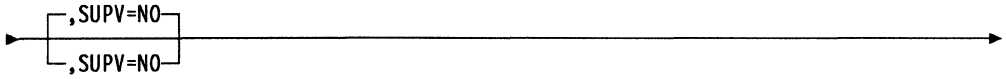
Format Register notation (odd-numbered register only)

Default None.

Remarks The return code values are:

X'0000' The register is 0 if VREVENT is successful.
X'0002' NO was previously set, or the task is already triggered.
X'0004' The RCB is not currently associated with a virtual route.
X'0008' The RCB is attached to a VRB that is inactive.

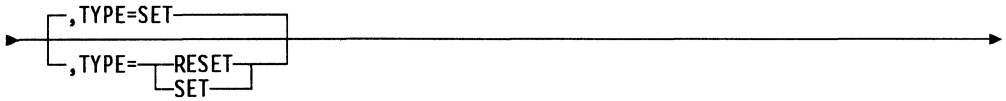
Register 6 is not allowed. The register specified cannot be the same as that specified for RCB.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format NO or YES.

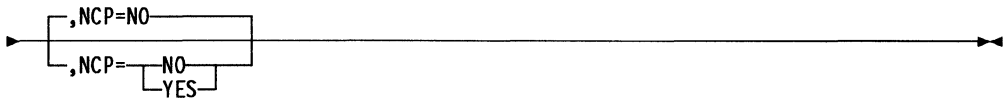
Default NO.



Function Specifies whether a task should be triggered when the virtual route exits the *held* state. TYPE=SET specifies that a task should be triggered. TYPE=RESET specifies that the task should not be triggered.

Format SET or RESET.

Default SET.



Function Specifies the type of code generated based on the issuer's storage protection key. NCP=YES specifies that the issuer is running in NCP storage protection key 0; inline code is generated. NCP=NO specifies that the issuer is running in NCP storage protection key 1; SVC code is generated.

Format YES or NO.

Default NO.

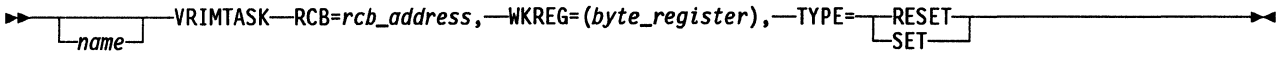
Remarks Specify this parameter only when SUPV=NO.

VRIMTASK—Initiate a Task When a Virtual Route Is Held

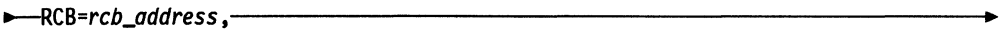
The VRIMTASK macro either sets or resets the immediate bit in the resource connection block (RCB). If the virtual route goes into the *held* state and the immediate bit is set, control is given to the immediate routine whose address is in the RCB. Include the DSECTs for XCXTRCB and XCXTEQU in routines issuing this macro.

Register 0 is not allowed for register parameters.

Syntax



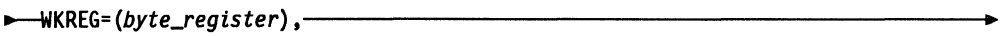
Parameters



Function Specifies the address of the RCB in which the immediate bit is to be either set or reset.

Format Register notation.

Default None.

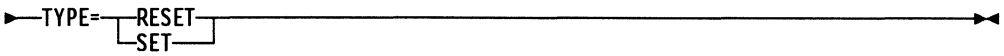


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Byte register notation.

Default None.

Remarks The register specified must not be the same as the register specified for the RCB keyword.



Function Specifies whether the immediate bit in the RCB should be turned on or off. If TYPE=SET, the immediate bit in the RCB should be turned on. If TYPE=RESET, the immediate bit should be turned off.

Format SET or RESET.

Default None.

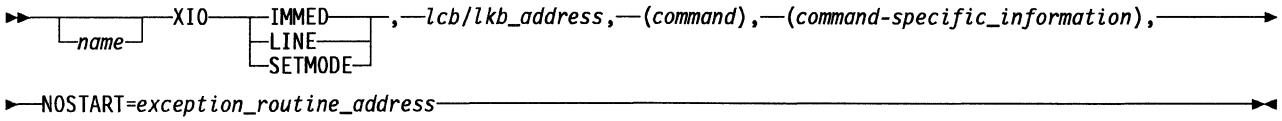
XIO—Start an Input/Output Operation

Line Operations

This XIO macro enqueues and begins an input/output (I/O) operation on any of the BSC, start-stop, or SDLC communication lines and on peripheral channel links. This macro is used to interface with program level 2 and 3 code. It can be issued only in level 5.

Register 0 is not allowed for register parameters.

Syntax



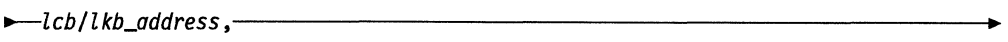
Parameters

Function Specifies whether an I/O operation will be initiated. If you specify LINE for a BSC or start-stop line, an I/O operation will be initiated. If you specify LINE for a SDLC link, the line will be enabled, disabled, or dialed, or the SDLC link scheduler will be activated. If you specify SETMODE, a set mode operation will be initiated. If you specify IMMED, an immediate communication function will be initiated.

Format LINE, SETMODE, or IMMED.

Default None.

Remarks There are several different SETMODEs, and the parameters for each one are coded differently. The best way to code a SETMODE function is to use an existing function as an example. If you code SETMODE or IMMED, you must code (*command*) and NOSTART.



Function Specifies the address of the line control block (BSC/SS) (LCB) or line control block (SDLC) (LKB) governing the line on which I/O will be initiated.

Format Label or register notation.

Default None. The supervisor assumes that the program issuing the XIO macro has been dispatched from the LCB that will be used.

Remarks Register 1 is not allowed.

Function Specifies a register containing the command, which is found in the immediate control flags in the adapter control block (ACB).

Format Byte register notation.

Default None.

Remarks This keyword is not allowed for the LINE function and is required for SETMODE and IMMED. Register 1(0) is standard.

When the SETMODE command is X'26', X'28', X'2A', X'2C', X'2E', X'30', X'32', or X'3A', the command must be in register 1(0), and a bit selection must be in register 1(1) to indicate the bit to be modified or tested.

Table 6. SETMODE Command Functions

Command Value	Function
X'26'	Test SCB bits
X'28'	Set SCB bits
X'2A'	Reset SCB bits
X'2C'	Set CCB bits
X'2E'	Reset CCB bits
X'30'	Test CCB bits
X'32'	Query/add/delete in service order table (SOT)
X'34'	NetView Performance Monitor (NPM) notify request
X'35'	Remove the XID on the LOBQ from the queue (idle line only)
X'36'	Set line control for a peripheral switched line (idle or busy line)
X'37'	Set line control for a peripheral switched line (idle line only)
X'38'	Test for active logical lines
X'3A'	Set subarea dial activity timer
X'42'	Test if the associated physical resource is active

The bit selection values for X'26', X'28', and X'2A' are:

X'01' Set/reset/test the control field operation mode bit.
X'02' Set/reset/test the set normal response mode bit.
X'03' Set/reset/test both the control field operation mode bit and the set normal response mode bit.
X'04' Test for XID on link outbound queue (only X'26').
X'05' Set the ready not received bit to force data link control (DLC) to send a poll instead of an I frame (only X'28').
X'06' Set waiting on good response (WOGR) bit (only X'28').

The bit selection values for X'2C', X'2E', and X'30' are:

X'01' Set/reset/test the incoming call bit.
X'02' Set/reset/test the link problem determination aid (LPDA) supported on the line bit.
X'03' Set/reset/test the LPDA supported on the line bit and the LPDA 2 supported on the line bit.
X'04' Set/reset/test the scanner down bit in the LNVT entry.
X'05' Set/reset/test echo-supported bit.
X'06' Reset CCBHPTS bit in CCBSETYP.
X'07' Set the secondary link activity time-out (LATO) timer (only X'2C').
X'08' Reset CCBRBLUC so no disconnect mode U-frames are sent in the reset state (only X'2E').

When the SETMODE command is X'32', the command must be in register 1(0), and a modifying command is required to be in register 1(1) to indicate an operation that will be performed on the service order table (SOT). The modifying commands for X'32' are:

X'01' Determine the number of available entries in the SOT.
X'02' Add an SCB/CUB entry in the SOT.
X'03' Delete all occurrences of an SCB/CUB entry from the SOT.

When the SETMODE command is X'34', the command must be in register 1(0), and a command modifier is in register 1(1). The command modifiers for X'34' are:

X'01' Request capability
X'02' START request
X'03' FORWARD request
X'04' STOP request
X'05' Collect data.

When the SETMODE command is X'35', the command must be in register 1(0), and a modifying command is required to be in register 1(1) with a pointer in register 5 to the station control block. There are no command modifiers for X'35'.

When the SETMODE command is X'36' or X'37', the command must be in register 1(0), and a modifying command is required to be in register 1(1) with a pointer in register 5 to the station control block. The modifying commands for X'36' and X'37' are:

X'01' Set the DLC role to primary.
X'03' Set the DLC role to secondary.
X'05' Set the DLC role to negotiable.

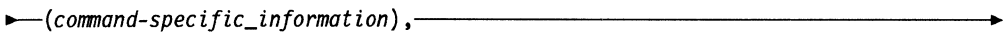
When the SETMODE command is X'38', the command must be in register 1(0). There are no command modifiers for X'38'.

When the SETMODE command is X'3A', the command must be in register 1(0), and a modifying command is required to be in register 1(1) to indicate an operation that will be performed on the subarea dial timer. The modifying commands for X'3A' are:

X'01' Start the subarea dial timer.
X'02' Stop the subarea dial timer.

When the SETMODE command is X'42', the command must be in register 1(0) and the command modifier must be in register 1(1). Register 5 must contain a pointer to the PIU buffers. The command modifiers for X'42' are:

X'01' ACTLINK request
X'02' CONNOUT request.



Function Specifies a register containing the command-specific field.

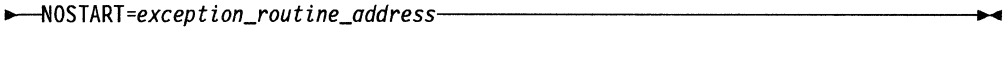
Format Register notation.

Default None.

Remarks This keyword is not allowed for the LINE and IMMED functions and is required for some SETMODEs. Register 5 is required.

When the SETMODE command is X'26', X'28', X'2A', X'2C', X'2E', X'30', X'3A', or X'32', the SCB address must be passed in this parameter.

When the SETMODE command is X'34', the address of the user resource control block requiring NetView Performance Monitor (NPM) activity must be passed in this parameter.



Function Specifies the address of the routine to receive control if an IMMED or SETMODE operation cannot be initiated. This parameter is required.

Format Register or label notation.

Default None.

Remarks Register 1 is not allowed.

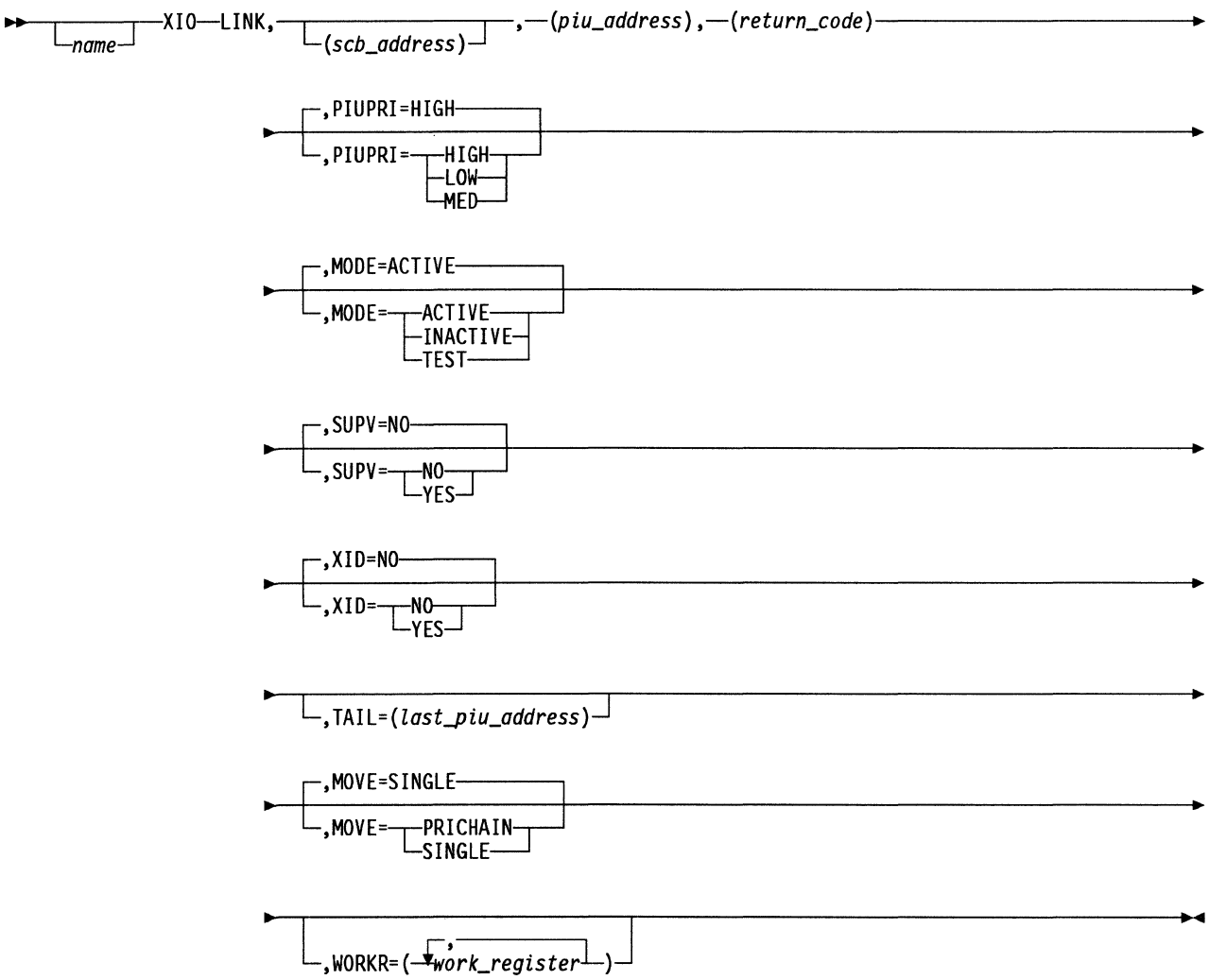
This keyword is not allowed for the LINE function.

SDLC Link Operations

This XIO macro transfers a path information unit (PIU) or chain of PIUs to a specified link outbound queue and begins the I/O routine that transfers the PIUs across the communication link.

Register 0 is not allowed for register parameters.

Syntax

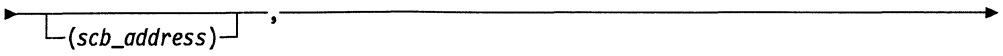


Parameters

Function Specifies that this XIO starts I/O on an SDLC link.

Format LINK.

Default None.

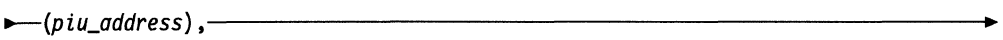


Function Specifies the register pointing to the station control block (SCB) or common physical unit block (CUB) on which to enqueue the PIU. The PIU is placed on the link outbound queue of the control block.

Format Register notation.

Default The SCB or CUB that activated the task.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed. Register 2 is standard.

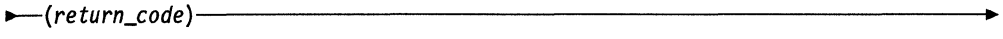


Function Specifies the register pointing to the PIU buffer chain to be transmitted. The PIU can be either a request or a response PIU.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed. Register 3 is standard.

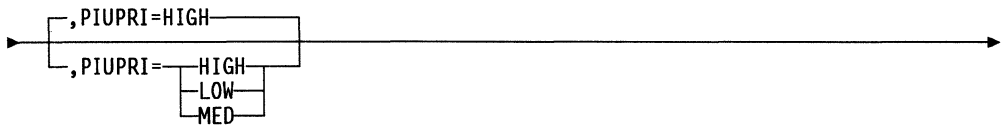


Function Specifies the register to receive the return code indicating the success or failure of the XIO LINK. A return code of X'00' indicates that the XIO LINK started properly. A nonzero code indicates that the station or link is in failure mode.

Format Register notation.

Default None.

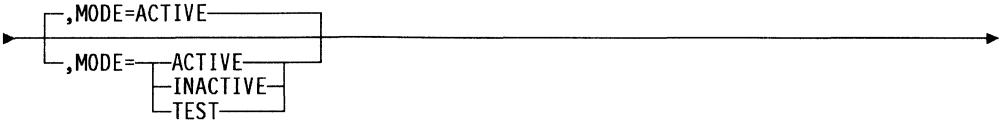
Remarks Register 1 is standard. The 8-bit return code is returned in the low-order byte of the specified register.



Function Specifies the PIU priority based on the value of the virtual route over which the BIND traverses for a multiple-entry queue.

Format HIGH, MEDIUM, or LOW.

Default HIGH.

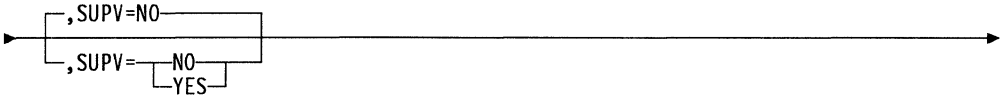


Function Checks that the link is in same state as specified.

When MODE=ACTIVE, normal data and XIDs can be transmitted on the link. When MODE=INACTIVE, only load and dump data can be transmitted. When MODE=TEST, only test data can be transmitted. If the link is not active, the return-code register indicates a path error and the XIO LINK function is not performed. If XID exchange with a station is in progress, a request to transmit another XID is rejected.

Format ACTIVE, INACTIVE, or TEST.

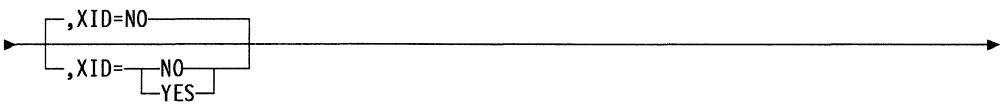
Default ACTIVE.



Function Specifies the level in which the issuer is running.
SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

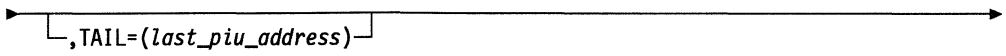


Function Specifies whether data is transferred to a specified link outbound queue. If XID=YES, data is transferred to a specified link outbound queue but is not transferred across the link. The PIU keyword contains the address of the XID data.

If XID=NO, data is transferred to a specific link outbound queue and is transferred across the link. The PIU keyword contains the address of the PIUs to be sent.

Format YES or NO.

Default NO.



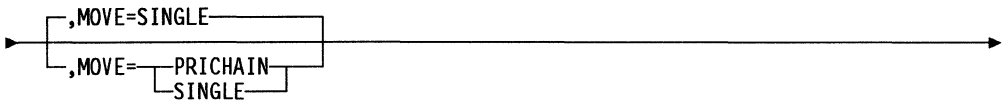
Function Specifies the register pointing to the last PIU of a chain of PIUs to be sent.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed. Register 4 is standard.

If MOVE=SINGLE, this keyword is ignored.

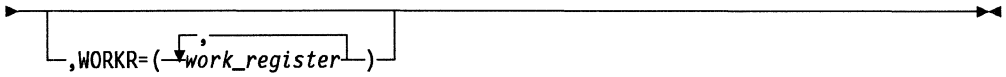


Function Specifies the number of PIUs sent to the link outbound queue. If MOVE=SINGLE, only one PIU is sent to the link outbound queue. If MOVE=PRICHAIN, one or more PIUs are sent to the tail of the link outbound queue.

Format PRICHAIN or SINGLE.

Default SINGLE.

Remarks If MOVE=PRICHAIN, you must specify the TAIL keyword.



Function Specifies a work register, the contents of which may be altered during execution of the macro. Specifying such registers makes the macro more efficient.

Format You can use any or all of the following registers: 1, 2, 3, 4, 5, and 7. Do not use equated values.

Default No work registers.

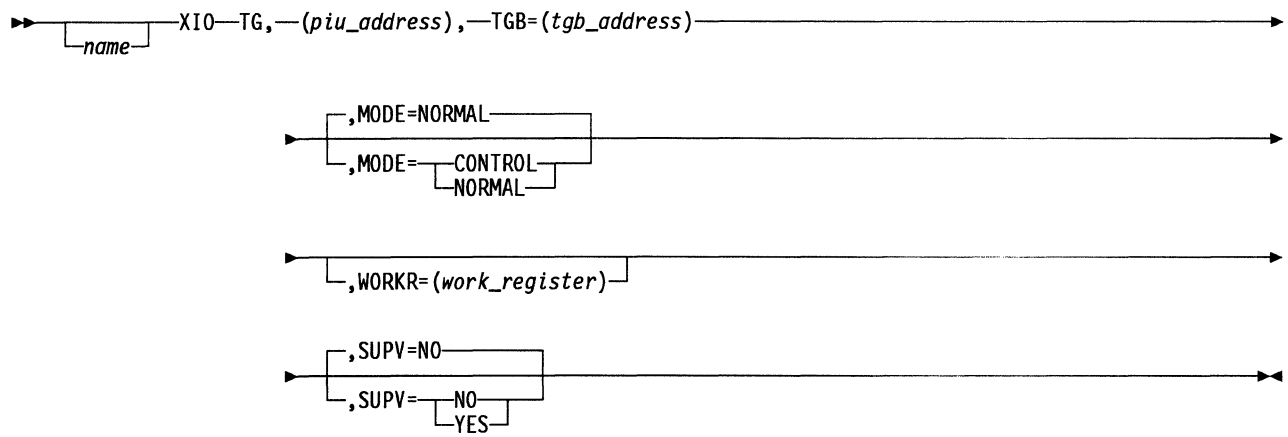
Remarks This keyword is valid only when SUPV=YES.

Transmission Groups

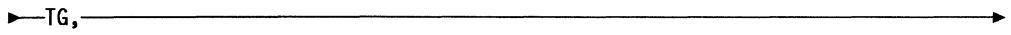
This XIO macro passes a PIU from within the path control function (explicit routing) to the specified transmission group control block (TGB) for routing to the current data link control (DLC).

Register 0 is not allowed for register parameters.

Syntax



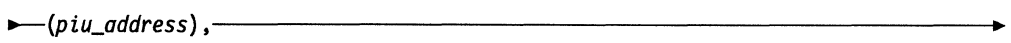
Parameters



Function Specifies that this XIO passes a PIU to the specified TGB for routing to DLC.

Format TG.

Default None.



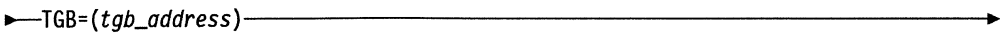
Function Specifies the buffer address of the PIU to be transferred.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. If SUPV=YES, registers 2 and 6 are not allowed. The register you specify must not be the same as the register specified for the TGB address. Register 3 is standard.

The PIU is discarded if it is rejected by DLC.



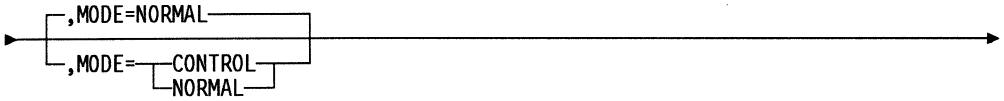
Function Specifies the address of the TGB for the routed PIU.

Format Register notation.

Default None.

Remarks Register 1 is not allowed. If SUPV=YES, register 6 is not allowed. Register 2 is standard.

The register you specify must not be the same as the register specified for the PIU address.

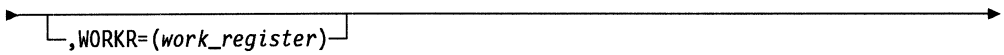


Function Specifies whether to pass this PIU to DLC depending on the state of the transmission group. If MODE=NORMAL, do not pass this PIU to DLC if the transmission group is not operative. If MODE=CONTROL, send the PIU to DLC regardless of the transmission group state.

Format NORMAL or CONTROL.

Default NORMAL.

Remarks The PIU is discarded if the DLC rejects the PIU.



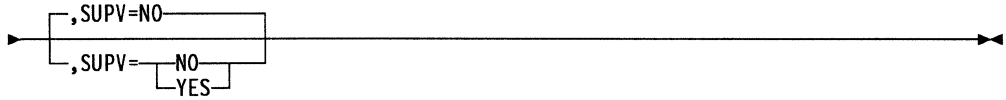
Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default None.

Remarks Register 6 is not allowed. If SUPV=NO, this keyword is not allowed.

The register must not be the same as the register specified for the TGB address or PIU address.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES indicates that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

XIOFL—Set or Test an SCB for a Multilink Transmission Group

The XIOFL macro is issued by data-link-control-level programs for a specified station control block (SCB) to correspond with the multilink transmission group function. Depending on the mode specified when XIOFL is invoked, NCP:

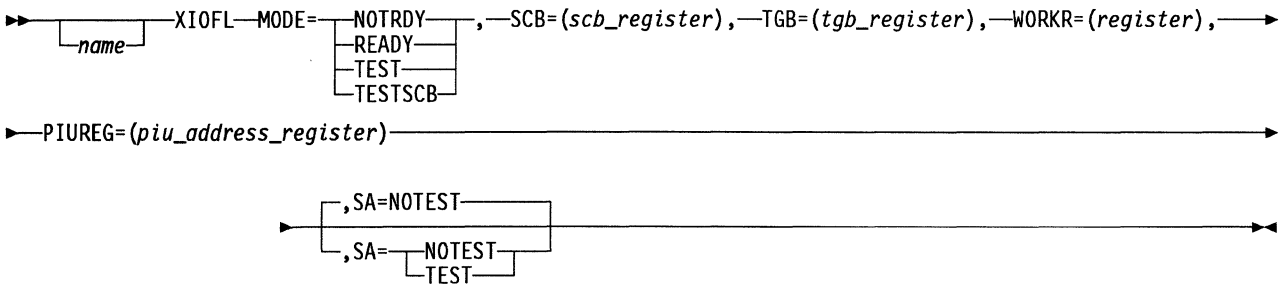
- Sets the SCB status to ready and initiates PIU transmission (MODE=READY)
- Sets the SCB status to not ready (MODE=NOTRDY)
- Tests the status of the SCB and its associated TGB (MODE=TEST)
- Tests the SCB to determine its characteristics (MODE=TESTSCB).

Register 6 must point to a save area that can be overwritten. You must assemble the DSECTs for the transmission group control block (TGB) (XCXTTGB), SCB (XXCXTSCB), and multilink transmission group control block (FLB) (XCXTFLB) in your program. If you use the MODE=READY keyword, you must include an EXTRN for CXAFLXS. This macro is expanded to execute inline with your code.

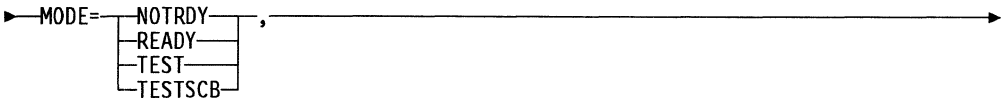
Register 0 is not allowed for register parameters.

Note: Do not invoke XIOFL from a compound IF structure.

Syntax



Parameters



Function Specifies that XIOFL is to access the specified control block, check the status, then set it, as coded.

When MODE=READY, XIOFL accesses the specified SCB and checks its current status. If the SCB is not ready, XIOFL sets it to ready status. If no other SCBs associated with this multilink transmission group are currently ready, the macro checks for other conditions and, if they are satisfactory, it dequeues a PIU and returns the address in the specified PIUREG register. If the SCB is ready, or if other SCBs in the multilink transmission group are ready, no action is taken.

When MODE=NOTRDY, XIOFL accesses the specified SCB and checks its current status. If the SCB is ready, XIOFL sets it to not-ready status. If the SCB is not ready, no action is taken.

When MODE=TEST, XIOFL accesses the specified TGB and checks its data link control (DLC) type. If the SCB is attached to a multilink transmission group, XIOFL sets the Z latch on and sets the C latch off. If the SCB is not attached to a multilink transmission group, XIOFL sets the Z latch off and sets the C latch on.

When MODE=TESTSCB and SA=TEST, XIOFL accesses the specified SCB and determines if the station is a subarea node, the station is associated with the specified TGB, and if the TGB is a multilink transmission group. If the specified SCB is for a subarea station attached to a multilink transmission group, XIOFL sets the Z-latch on and sets the C-latch off. If the SCB is not attached to a multilink transmission group, XIOFL sets the Z-latch off and sets the C-latch on.

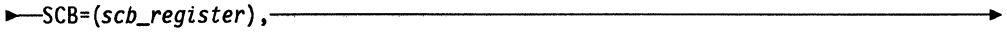
When MODE=TESTSCB and SA=NOTEST, XIOFL accesses the specified SCB and determines whether the station is associated with the specified TGB and the TGB is a multilink transmission group. If the SCB is attached to a multilink transmission group, XIOFL sets the Z-latch on and sets the C-latch off. If the SCB is not attached to a multilink transmission group, XIOFL sets the Z-latch off and sets the C-latch on.

Format READY, NOTRDY, TEST, or TESTSCB.

Default None.

Remarks Specify MODE=READY and MODE=NOTRDY only when running in level 3 or, if level 3 is disabled, in level 4.

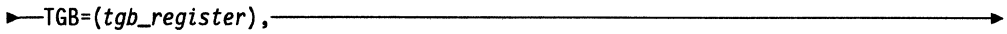
Warning: MODE=READY destroys register 1, even if it was specified by WORKR.



Function Specifies the register pointing to the SCB to be processed.

Format Register notation.

Default None.



Function Specifies the register containing the address of the TGB to be checked.

Format Register notation.

Default None.

Remarks This keyword is valid only when MODE=TEST.

```
►──WORKR=(register),──────────────────────────────────────────────►
```

Function Specifies a work register, the contents of which may be altered during execution of the macro. The macro uses this register to determine how to set the C and Z latches.

Format Register notation.

Default None.

Remarks You must specify an odd-numbered register. If MODE=TEST, this keyword is required; it is optional when MODE=READY or MODE=NOTRDY.

```
►──PIUREG=(piu_address_register)──────────────────────────────────►
```

Function Specifies the register used to return the PIU address if the PIU can be sent.

Format Register notation.

Default None.

Remarks Register 3 is standard. Do not use register 2. If MODE=READY, this keyword is required. When MODE=NOTRDY and MODE=TEST, this keyword is not valid.

```
                ┌─,SA=NOTEST─┐
►───────────────┤            ├──────────────────────────────────►◄
                └─,SA=─┬─NOTEST─┬─┘
                       └─TEST───┘
```

Function SA=NOTEST means that the input SCB is for a subarea station. SA=TEST means that the input SCB should be checked to determine whether it is for a subarea station.

Format TEST or NOTEST.

Default NOTEST.

Remarks This keyword is required for MODE=TESTSCB.

XPC—Pass a PIU between Data Link Control and Path Control

The XPC macro passes a path information unit (PIU) between the data link control (DLC) function and the specified transmission group within the path control function. Data flow direction is specified with the positional keywords OUT and IN.

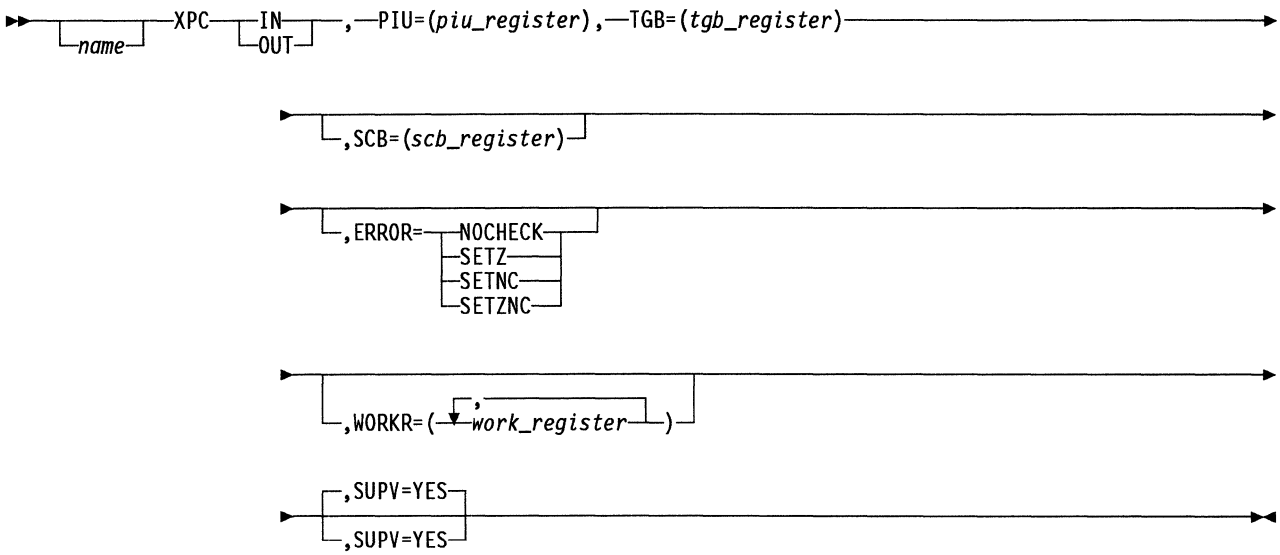
OUT transfers the PIU to path control, which performs the routing to the appropriate destination. IN indicates that the DLC is going to begin transmission of the PIU over a link. IN causes virtual route pacing fields to be set in the PIU if necessary.

This macro is executed by interrupt level code (SUPV=YES) only.

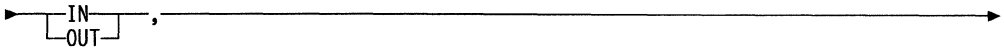
Register 0 is not allowed for register parameters.

Note: The PIU total text count (UIBTCNT) and last buffer address (UIBLBBA) must be valid before executing.

Syntax



Parameters

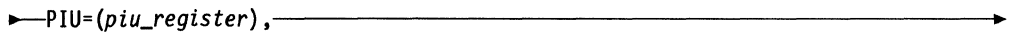


Function Specifies the direction of data flow and the function to be performed.

OUT specifies that data flow is from the host. The PIU is transferred to the transmission group function for routing to the appropriate destination. IN specifies that data flow is to the host and that the data link control is going to begin transmission of the PIU over a link. IN causes virtual route pacing fields to be set in the PIU if necessary. The PIU is not transferred to the transmission group, but parameters in the PIU are used by the transmission group.

Format IN or OUT.

Default None.



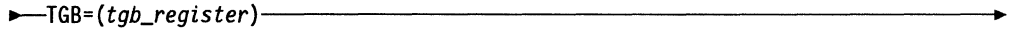
Function Specifies the buffer chain to be routed.

Format Register notation.

Default None.

Remarks If you code OUT, the content of this register is set to 0 if an error occurs and to nonzero if no error occurs.

Register 6 is not allowed. The register must not be the same as the TGB or WORKR register. Register 3 is standard.

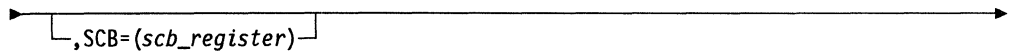


Function Specifies the transmission group control block (TGB) through which the PIU is routed.

Format Register notation.

Default None.

Remarks Register 6 is not allowed. Register 2 is standard. The register must not be the same as the PIU or WORKR register.



Function Specifies the station control block (SCB) from which the PIU came.

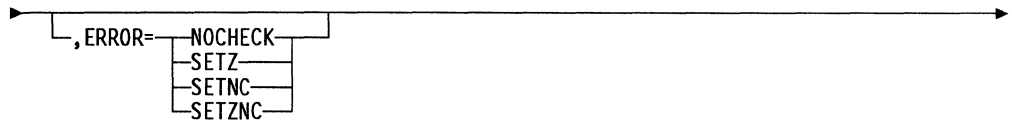
Format Register notation.

Default None.

Remarks This keyword is necessary if you code OUT and performance collection by transmission priority is supported for the station.

This keyword is ignored when you code IN.

Register 6 is not allowed. The register must not be the same as the PIU, TGB, or WORKR register. Register 5 is standard.



Function Specifies whether to perform error checking. If ERROR=NOCHECK, no checking is done. If ERROR=SETZNC, ERROR=SETZ, or ERROR=SETNC, an error indication is returned if the transmission group function cannot transfer the PIU because of any of the following:

- The FID type is not valid or is not FID0, FID1, or FID4.
- The PIU length is too short.

- The PIU's TH length field does not match the actual PIU length.
- The subarea address value in the PIU's TH is 0.

Format SETZNC, SETZ, SETNC, or NOCHECK.

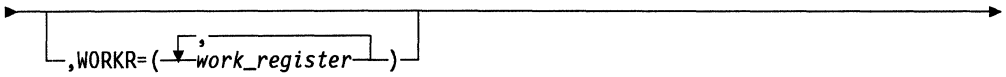
Default None.

Remarks If you code IN, this keyword is not allowed. If you code OUT, you must code this keyword.

If you code ERROR=SETZNC, ERROR=SETZ, or ERROR=SETNC and no error occurs, the Z latch is set to 0. If an error occurs, the Z latch is set to 1 and the C latch is set to 0.

If any error is detected, the PIU's buffers are released, and the content of the PIU register is set to 0 on return.

NOCHECK is valid only if the transmission group is not in migration mode. Specify NOCHECK only if the same checks have been performed on this PIU (FID4).

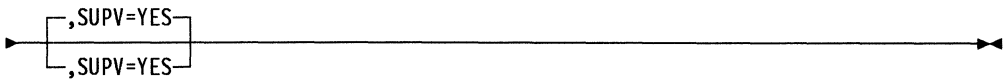


Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format Register notation.

Default No work registers.

Remarks Register 6 is not allowed. The register must not be the same as the PIU or TGB register.



Function Specifies that the issuer is running in an interrupt level.

Format YES.

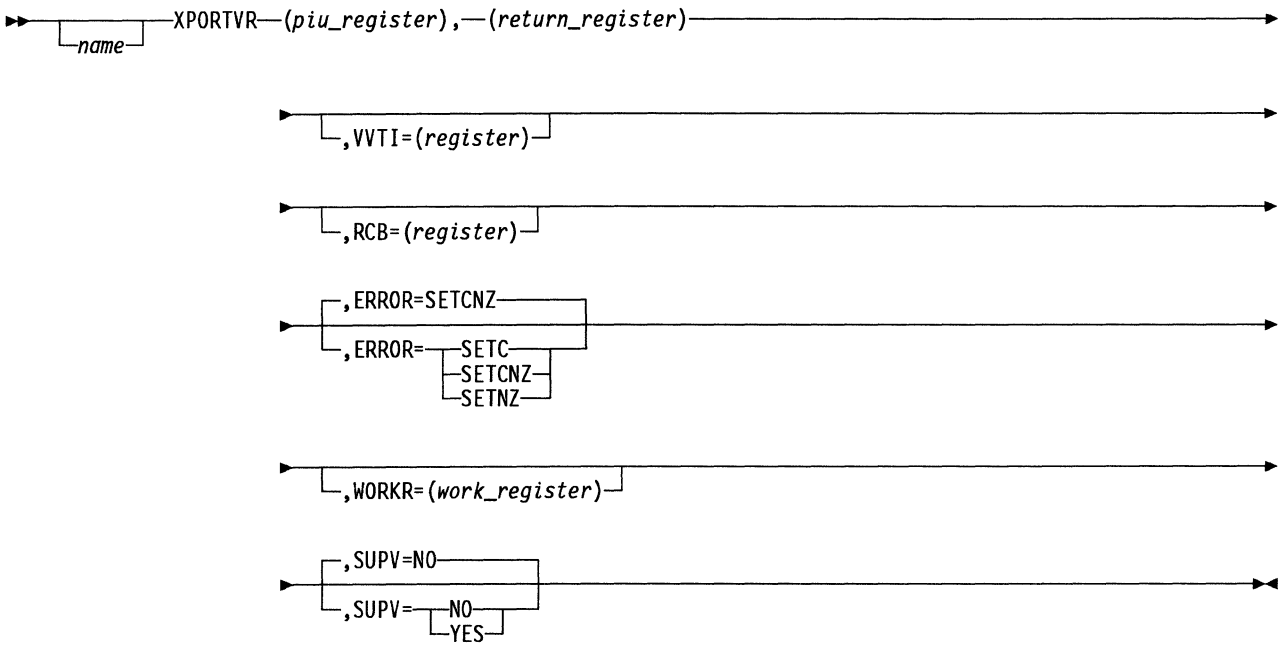
Default YES.

XPORTVR—Deliver a PIU to a Virtual Route

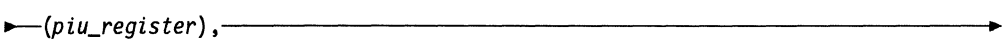
The XPORTVR macro delivers a path information unit (PIU) to virtual route routing from the boundary function or physical unit services. The PIU is enqueued to the virtual route transmit queue associated with the virtual route identified by the virtual route vector table index (VVTI). The VVTI is passed either in the FID4 transmission header or as a separate parameter. If the virtual route is inoperative, an error code is returned.

Register 0 is not allowed for register parameters.

Syntax



Parameters

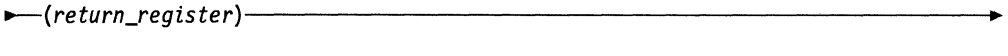


Function Specifies the PIU to be routed.

Format Register notation.

Default None.

Remarks Register 3 is standard. If SUPV=NO, register 1 is not allowed. If SUPV=YES, register 6 is not allowed.

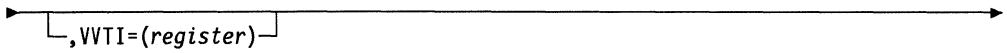


Function Specifies the register to receive the return code. Return codes are X'00' for a PIU successfully enqueued to the virtual route transmit queue and X'01' for an inoperative virtual route.

Format Register notation.

Default None.

Remarks The return code is in the low-order byte of the register. If SUPV=YES, register 6 is not allowed. If SUPV=YES, you must specify a return register. If SUPV=NO, a return register is not allowed. Register 1 is standard.



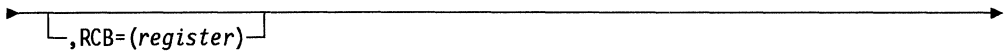
Function Specifies the register that contains the VVTI.

Format Register notation.

Default If you do not code this parameter, the VVTI is assumed to be in the FID4 transmission header of the PIU.

Remarks If you code this parameter, the value of the VVTI specified takes precedence over any value in the FID4 transmission header.

If SUPV=NO, register 1 is not allowed. If SUPV=YES, register 6 is not allowed. The VVTI register must not be the same as the register for PIU. Register 2 is standard.



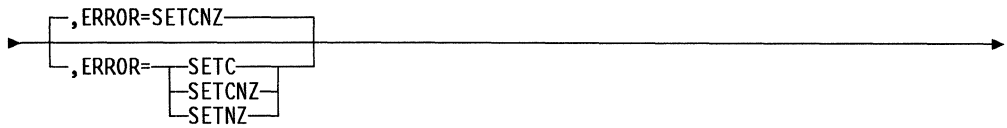
Function Specifies the register that contains the address of the resource's RCB.

Format Register notation.

Default None.

Remarks If SUPV=YES, register 6 is not allowed. If SUPV=NO, register 1 is not allowed. The RCB register you specify must not be the same as the registers specified for PIU, VVTI, and WORKR. Register 4 is standard.

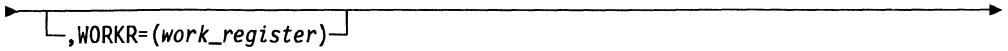
Note: This keyword is required for BSB resources. If you do not specify RCB, unpredictable results can occur if the virtual route is held.



Function Specifies whether the C latch, the Z latch, or both are to indicate whether the PIU was routed. If ERROR=SETCNZ or ERROR=SETC, the C latch is set to 1 if the PIU was not routed or to 0 if the PIU was routed. If ERROR=SETCNZ or ERROR=SETNZ, the Z latch is set to 0 if the PIU was not routed or to 1 if the PIU was routed.

Format SETNZ, SETC, or SETCNZ.

Default SETCNZ.



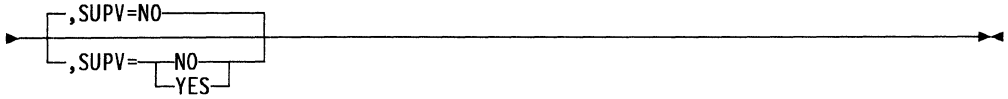
Function Specifies a work register, the contents of which may be altered during execution of the macro.

Format You can specify any or all of the following registers: 1, 2, 3, 4, 5, and 7. Use register notation.

Default No work registers.

Remarks This keyword is valid only when SUPV=YES.

The registers specified must not be the same as the PIU, RETURN, or VVTI register.



Function Specifies the level in which the issuer is running. SUPV=NO specifies that the issuer is running in level 5. SUPV=YES specifies that the issuer is running in an interrupt level.

Format YES or NO.

Default NO.

Chapter 3. Entrances and Exits for User-Written Line Control

The following sections describe the attachments for user-written line control code in NCP code. The table in each section provides:

- The function to be performed
- The name of the module that gives control to user-written code
- The label in the group control block (GCB) where the address of the user-written routine is stored
- The level the module runs in
- The register values at the entry point to the user-written routine
- The register values expected by NCP on return from the user-written routine
- The process used to return to NCP, which can be either by the label of an immediate NCP return address or an NCP-provided method of returning to NCP (EXIT, CALL and SUBROUTINE linkage, BRANCH, and so on).

The immediate NCP return address is the point at which control should be given back to NCP if NCP is to handle the execution of the function. This method can be used if user-written code gets control and does not do the function in question. Along with the immediate return address, an alternate method of return can be indicated. The alternate return is usually used if user-written code executes the function and no other NCP processing is required for that function. If an immediate return address is not specified, the indicated return method should always be used.

Note: Some of these attachments may not receive control when certain diagnostic procedures or traces are active (wrap test, for example).

XIO Interrupts

Table 7 (Page 1 of 2). Attachment Details for XIO Interrupts

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
XIO Line	CXDCG0B (CXEXIOCD)	GCBXIOLN	3	R1 ▲ GCB R2 ▲ UACB R3 Disable mask R4 ▲ LGT R5 Work register R6 ▲ PSA / CCBBAR R7 Work register	R2 ▲ UACB R3 Disable mask R4 ▲ LGT R5 Work register R6 ▲ PSA / CCBBAR R7 Work register	Immediate at CXENEO1 or use EXIT (if no NCP processing required). Also, increment level 5 IAR by 2 bytes if no errors.

Table 7 (Page 2 of 2). Attachment Details for XIO Interrupts

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
XIO SETMODE	CXDCG0B (CXESETMD)	GCBXIOSM	3	**R1** ▲ GCB **R2** ▲ UACB **R3** Work register **R4** ▲ LGT **R5** Work register **R6** ▲ PSA / CCBBAR **R7** CCBCTL CCBTYPE (See note 1)	**R1** Work register **R2** ▲ UACB **R3** Work register **R4** ▲ LGT **R5** Work register **R6** ▲ PSA / CCBBAR **R7** CCBCTL CCBTYPE (See note 2)	Immediate at CXENEO2 or use EXIT (if no NCP processing required). Also increment level 5 IAR by 2 bytes if no errors.
XIO Immediate	CXDCG0B (CXEIMEDC)	GCBXIOIM	3	**R1** ▲ GCB **R2** ▲ UACB **R3** Work register **R4** ▲ LGT **R5** Work register **R6** ▲ PSA / CCBBAR **R7** CCBCTL CCBTYPE	**R1** Work register **R2** ▲ UACB **R3** Work register **R4** ▲ LGT **R5** Work register **R6** ▲ PSA / CCBBAR **R7** CCBCTL CCBTYPE	Immediate at CXENEO3 or use EXIT (if no NCP processing required). Also, increment level 5 IAR by 2 bytes if no errors.
XIO Link	CXDCG02	GCBXIOLK	4	**R1(0)** SVC code **R1(1)** comm. flags **R2** ▲ UACB **R3** ▲ PIU **R4** Return address (See note 3) **R6** ▲ Save area **R7** Entry point address	**R1(0)** SVC code **R1(1)** comm. flags **R3** ▲ PIU **R4** Return address or tail pointer if adding chain of PIUs **R6** ▲ Save area	Immediate at CXANEO1 or branch R4 (if no NCP processing required).

Notes:

1. Level 5 R1(0) contains index into setmode command decode vector table.

   Level 5 R1(1) contains setmode command modifier.

   For NPM Notify setmode, level 5 R5 contains the LKB or SCB address.

2. For line trace setmode R7 is used as an exception condition indicator. The following values generate sense codes. (Otherwise R7 still contains CCBCTL CCBTYPE.)

   X'80'—0801,0000
   X'40'—080C,0007
   X'20'—0801,0000
   X'0C'—080C,0007
   X'04'—0815,0000
   X'02'—0815,0000 (on multipoint subarea node)
   X'02'—084B,0000
   X'01'—0801,0000

3. R4 in save area has tail pointer if adding chain of PIUs.

Box Event Record

Table 8. Attachment Details for Box Event Record

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
Box Event Record	CXABDCRP	GCBBERP	1, 2, 3, and 4	**R1** ▲ GCB **R2** ▲ BER **R4** Entry point address **R6** ▲ Save area **R7** Return address	**R2** ▲ BER **R5(1)** Return code X'00'=BER built, X'FF'=no BER **R6** ▲ Save area	Immediate. Use branch R7 (if no NCP processing required).

Timer Interrupts

Table 9. Attachment Details for Timer Interrupts

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
Line Error Time-out	CXDCG07 (CXCCLINT)	GCBERR	3	**R2** Return address **R5** ▲ UACB **R6** ▲ Save area **R7** Entry point address	**R5** ▲ UACB **R6** ▲ Save area	Immediate at CXCNEO4 (see note 1) or use TVSRTRN (if no NCP processing required).
Shoulder-Tap Time-Out	CXDCG07 (CXCCLINT)	GCBSTAP	3	**R2** Return address **R5** ▲ UACB **R6** ▲ Save area **R7** Entry point address	**R5** ▲ UACB **R6** ▲ Save area	Immediate at CXCNEO3 (see note 1) or use TVSRTRN (if no NCP processing required).
Lagging Shoulder-Tap Time-Out	CXDCG07 (CXCCLINT)	GCBLAGST	3	**R2** Return address **R5** ▲ UACB **R6** ▲ Save area **R7** Entry point address	**R5** ▲ UACB **R6** ▲ Save area	Immediate at CXCNEO2 (see note 1) or use TVSRTRN (if no NCP processing required).

Note:

1. If the immediate return is used, R6 (save area) is not used.

Timer Tick

Table 10. Attachment Details for Timer Tick

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
Display Control-Update Refresh (Timer Tick)	CXDCG07 (CXCCPDRS)	Not used	3	**R3** Entry point address **R4** CXCCPDSP address **R6** ▲ Save area	**R4** CXCCPDSP address **R6** ▲ Save area	Use TVSRTRN

Level 2 and Level 3 Interrupts

Table 11. Attachment Details for Level 2 and Level 3 Interrupts

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point		Register Values at Exit		Return to NCP
Level 2 Interrupt	CXDCG00 (CXBTRP2C)	GCBL2	2	R2 R3 R4 R6	▲ GCB SCF/PDF from PSA ▲ LNVT ▲ PSA	R2 R3 R4 R6	Must point to UACB using FINDUACB with LNVT (R4) SCF/PDF from PSA ▲ LNVT ▲ PSA	Immediate at CXBNEO1 (error re-entry point only) or use EXIT (if no NCP processing required).
Level 3 Interrupt	CXDCG0D (RNENDR)	GCBL3	3	R1 R2 R3	▲ GCB ▲ UACB SYSL2DEM equate (X'20')	R2 R3	▲ UACB SYSL2DEM equate (X'20')	Immediate at CXENEO4 or use EXIT (if no NCP processing required).

Level 3 Router I/S and D/S Interrupt Handler

Table 12. Attachment Details for Level 3 Router I/S and D/S Interrupt Handler

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point		Register Values at Exit	Return to NCP
Level 3 CA Data Status Interrupt	CXCCNEO	GCBDS	3	R1 R3(1) R5 R6	▲ CAVT entry for CA with interrupt High byte of TA data for CA with interrupt ▲ GCB ▲ Save area		Use EXIT
Level 3 CA Initial Select Interrupt	CXCCNEO	GCBIS	3	R1 R3(1) R5 R6	▲ CAVT entry for CA with interrupt High byte of TA data for CA with interrupt ▲ GCB ▲ Save area		Use EXIT

Channel Errors

Table 13 (Page 1 of 3). Attachment Details for Channel Errors

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point		Register Values at Exit		Return to NCP
Set up all ESCs owned by user code on this CA to be cleared of traffic so CA can be disabled.	CXCAERP	GCBL1ERP	1	R1 CERSTATE ERP seq. flags R2 ▲CER R3(1) NEORCTL (X'24') R4 ▲CAVT entry R5 User entry point R6 ▲Save area R7 Return address		R1 CERSTATE Erp seq. flags R2 ▲CER R4 ▲CAVT entry R6 ▲Save area		Return address passed in R7
Handle I/S interrupt for ESCs owned by user code, reset it, then exit.	CXCAERP	GCBL1ERP	1	R1 IOH IN X'00' R2 ▲CER R3(1) NEOL3IS (X'2C') R4 ▲CAVT entry R5 User entry point R6 ▲Save area				Use EXIT
System reset or selective reset. Attempt to disable CA.	CXCAERP	GCBL1ERP	1	R1 IOH IN X'00' R2 ▲CER R3(1) NEOSYSR (X'30') R4 ▲CAVT entry R5 User entry point R6 ▲Save area R7 Return address		R1 IOH IN X'00' R2 ▲CER R4 ▲CAVT entry R6 ▲Save area		Return address passed in R7
Handle D/S interrupt for ESC owned by user code, reset it, then exit.	CXCAERP	GCBL1ERP	1	R1 IOH IN X'00' R2 ▲CER R3(1) NEOL3DS (X'28') R4 ▲CAVT entry R5 User entry point R6 ▲Save area				Use EXIT

Table 13 (Page 2 of 3). Attachment Details for Channel Errors

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
Clean up all logical and physical paths to this CA so no more traffic is routed through it.	CXBCAPBF	GCBL1ERP	1	**R2** ▲CAVT **R3(1)** NEODSAB (X'20') **R4** ▲L1B **R5** User entry point **R6** ▲Save area **R7** Return address	**R2** ▲CAVT **R4** ▲L1B **R6** ▲Save area	Return address passed in R7
Ground fault error processing, I/S flag is on.	CXBL1PM(372X) (CXBADAP)	GCBL1ERP	1	**R1** CA error register **R2** ▲AST or AIT **R3(1)** NEOCBIS (X'0C') **R4** ▲L1B **R5** User entry point **R6** ▲Save area **R7** Return address	**R1** CA error register **R2** ▲AST or AIT **R4** ▲L1B **R6** ▲Save area	Return address passed in R7
Ground fault error processing, D/S flag is on.	CXBL1PM(372X) (CXBADAP)	GCBL1ERP	1	**R1** CA error register **R2** ▲AST or AIT **R3(1)** NEOCBDS (X'10') **R4** ▲L1B **R5** User entry point **R6** ▲Save area **R7** Return address	**R1** CA error register **R2** ▲AST or AIT **R4** ▲L1B **R6** ▲Save area	Return address passed in R7
Internal CA error, I/S flag is on.	CXBL1PM(372X) (CXBADAP)	GCBL1ERP	1	**R1** CA error register **R2** ▲AST or AIT **R3(1)** NEOCIIS (X'14') **R4** ▲L1B **R5** User entry point **R6** ▲Save area **R7** Return address	**R1** CA error register **R2** ▲AST or AIT **R4** ▲L1B **R6** ▲Save area	Return address passed in R7
Handle interrupt CA error processor	CXBL1PM(372X) (CXBADAP)	GCBL1ERP	1	**R1** CA error register **R2** ▲CER **R3(1)** NEOCIDS (X'18') **R4** ▲L1B **R5** User entry point **R6** ▲Save area **R7** Return address	**R1** CA error register **R2** ▲CER **R4** ▲L1B **R6** ▲Save area	Return address passed in R7

Table 13 (Page 3 of 3). Attachment Details for Channel Errors

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point		Register Values at Exit		Return to NCP
IOH input error processor	CXBADAP CXBADAPA	GCBL1ERP	1	R1 CA error register R2 ▲ AST or AIT R3(1) NEOCIL3 (X'1C') R4 ▲ L1B R5 User entry point R6 ▲ Save area R7 Return address		R1 CA error register R2 ▲ AST or AIT R4 ▲ L1B R6 ▲ Save area		Return address passed in R7
IOC-detected AIO error processing	CXBL1PM(372X) (CXBAIO)	GCBL1ERP	1	R1 Work register R2 ▲ CAB R3(1) NEOAIO (X'04') R4 ▲ L1B R5 User entry point R6 ▲ Save area R7 Return address		R1 Work register R2 ▲ CAB R4 ▲ L1B R6 ▲ Save area		Return address passed in R7
CA PIO error processor	CXBL1PM(372X) (CXBPIO)	GCBL1ERP	1	R1 CA error register R2 ▲ AST or AIT R3(1) NEOPIO (X'08') R4 ▲ L1B R5 User entry point R6 ▲ Save area R7 Return address		R1 CA error register R2 ▲ AST or AIT R4 ▲ L1B R6 ▲ Save area		Return address passed in R7
CA AIO error processor	CXBL1PM	GCBL1ERP	1	R1 Interrupt requests R2 Work register R3(1) NEODSAB (X'20') R4 ▲ L1B R5 User entry point R6 ▲ Save area R7 Return address		R1 Interrupt requests R2 Work register R4 ▲ L1B R6 ▲ Save area		Return address passed in R7

Note: R3(1) contains a code indicating the reason the routine was called.

General User-Code Functions

Table 14. Attachment Details for General User-Code Functions

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to NCP
User system reset routine	CXDCG06 (CXCAIOS3)	GCBRESET	3	**R1** ▲ GCB **R2** ▲ CAE **R4** ▲ CAB **R6** ▲ Save area	**R1** ▲ GCB **R2** ▲ CAE **R4** ▲ CAB **R6** ▲ Save area **R5** Set for IOH OUT 7 to CA Control Register **R6** ▲ Save area	At CYPNRSET
User queue scan routine	CXDCG06 (CXCAIOS3)	GCBSCAN	1	**R1** ▲ GCB **R6** ▲ Save area	**R1** ▲ GCB **R6** ▲ Save area	At CYPNSTRT
User queue scan routine	CXCAERP	GCBSCAN	3	**R1** ▲ GCB **R6** ▲ Save area	**R1** ▲ GCB **R6** ▲ Save area	At CYPNSTRT
User RECMS/NMVT build routine	CXDKINP	GCBRNBLD	5	**R2** ▲ LKB		Use CALL and SUB-ROUTINE linkage
User global interface routine	CXANEOR	Not used	1, 2, 3, and 4	**R1** ▲ NEOG parameter status entry **R6** ▲ Save area **R7** Return address	**R6** ▲ Save area	Return address in R7 (PERFORM-ROUTINE linkage is recommended).
User initialization routine	CXFINITC	Not used	1	**R3** User entry point **R6** Return address		Branch to address in R6
User accounting notify routine	CXJQNNOT	Not used	5	**R1** ▲ Reason X'0001'—Enabled X'0002'—Solicit X'0003'—Disabled **R6** ▲ Save area	**R6** ▲ Save area	NEOAXT
User accounting notify routine	CXJQNNAT	Not used	5	**R1** ▲ Reason X'0004'—Request user accounting parameter CV **R2** ▲ Buffer for CV **R6** ▲ Save area	**R2** ▲ Buffer with user accounting parameter CV **R6** ▲ Save area	NEOAXT
User accounting notify routine	CXJQNLSN	Not used	5	**R1** ▲ Reason X'0005'—Change user accounting parameter CV **R2** ▲ Buffer for CV **R6** ▲ Save area	**R2** ▲ Buffer from input **R6** ▲ Save area	NEOAXT

NCP Code Functions

Table 15 (Page 1 of 2). Attachment Details for Using NCP Code Functions

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point	Register Values at Exit	Return to User
Entry point CXERETRY (Level 3 X21 Call Request Retry)	User routine	Not used	3	**R2** ▲ ACB (Xmit if FDX) **R5(1)** Completion codes field (CCBCMPCD) **R6** ▲ PSA **R7(1)** Line type (CCBTYPE)	No return to user	No return to user from this routine
Entry point CXERCVNT (FDX Receive Leg Initialization). (See note 1)	User routine	Not used	3	**R1(1)** (See note 2) **R2** ▲ ACB (Xmit if FDX) **R3** None (See note 3) **R4** Return address **R5** None (See note 3) **R6** ▲ Save area **R7(0)** Control flags (CCBCTL) **R7(1)** Line type (CCBTYPE)	**R2** ▲ ACB (Rcv if FDX) **R3** None (See note 3) **R5** None (See note 3) **R6** ▲ Save area **R7(0)** Control flags (CCBCTL) **R7(1)** Line type (CCBTYPE)	Returns to address in R4
Entry point CXEXMTNT (Initiate a Transmission)	User routine	Not used	3	**R1** Return address **R2** ▲ ACB **R3** None (See note 4) **R4** Address to store in CCBL3 (FDX XMT Leg Only) **R6** ▲ Save area **R7(0)** Control flags (CCBCTL) **R7(1)** Line type (CCBTYPE) (See note 5)	**R2** ▲ ACB (XMT Leg if FDX) **R3** None (See note 4) **R6** ▲ Save area **R7(0)** Control flags (CCBCTL) **R7(1)** Line type (CCBTYPE)	Returns to address in R1

Table 15 (Page 2 of 2). Attachment Details for Using NCP Code Functions

Function	Called from	Field Name in GCB	Level	Register Values at Entry Point		Register Values at Exit		Return to User
Entry point CXBDQACB (Queue UACB to Level 3 CCPQ; used only when you code ROUTINE=YES on the POSTUACB macro).	User routine	Not used	2	R2 R4	▲ UACB Return address	R1 R2	Work register ▲ UACB	Returns to address in R4

Notes:

1. Routine CXERCVNT performs no processing unless CCBL2=CCBDLIDL.
2. For FDX receive leg initialization, R1(1) contains the following parameters:

1...	An RR with poll is to be transmitted (set for an HPTSS line only)
.1..	Reserved
..1.	Reserved
...1	Reserved
.... 1...	Reserved
.... .1..	Set "answer requested" in PSA Mod
.... ..1.	Reserved
....1	Reserved

3. R3 and R5 are saved on entry, restored on exit, and used as work registers.
4. R3 is saved on entry, restored on exit, and used as a work register.
5. 24(R6) contains a pointer to buffers to be transmitted for XID, TEST, FRMR, and I-FRAMES.

Part 2. SSP Customization

Chapter 4. NDF Utility Directory

This chapter contains General-Use Programming Interface and Associated Guidance Information.

This chapter describes the SSP NDF internal utilities for user-written generation applications. These utilities provide functions such as manipulating strings, passing link-edit statements, and creating error messages. These utilities are called using the transfer vector table (XVT), which is described in detail in *NCP and SSP Customization Guide*.

In this chapter, the descriptions of the utilities are arranged alphabetically. The description of each utility includes the function of the utility, a description of its parameters, and one or more examples.

NDF Internal Utilities Grouped by Function

The NDF internal utilities provide the following general functions:

- Validate character and number strings
- Print error messages in the generation report
- Pass link-edit statements to NDF
- Generate table assembly source
- Trace utilities and their parameters
- Manipulate strings
- Convert number strings to numbers
- Process a generation definition
- Search the SYSLIB data set for a member.

The NDF internal utilities are grouped by function in the following list.

Validate character and number strings:

ICNCVLAB	Validate statement symbols.
ICNCVRNG	Validate a value range.
ICNCVTOK	Validate a string.

Print error messages to the generation report:

ICNERPST	Post an error message.

Pass link-edit statements to NDF:

ICNLEPTN	Post a link-edit statement.
ICNLEPTS	Post link-edit statements to create a separate load module from table 1.
ICNLEPT2	Post link-edit statements to create a separate load module from table 2.

Generate table assembly source:

ICNOBPUN	Punch table 1 assembly source.
ICNOBPU2	Punch table 2 assembly source.

Trace utilities and their parameters:

ICNRPFMT	Print a formatted string.
ICNRPINF	Print a default or inheritance message.
INCRPPBT	Trace a Boolean flag.
ICNRPPCH	Trace a standard string.
ICNRPPCX	Trace a nonstandard string.
ICNRPPFX	Trace a decimal number.
INCRPPHX	Trace a hexadecimal number.
ICNRPPNM	Trace the value of a number.
ICNRPTRC	Trace procedure entry or exit.

Manipulate strings:

ICNSMAFT	Return the substring following the target string.
ICNSMALN	Return the actual length of a string.
ICNSMBEF	Return the substring before the target string.
ICNSMCAT	Concatenate strings.
ICNSMCMP	Compare strings.
ICNSMFMT	Format a value into a string.
ICNSMFND	Find a string.
ICNSMLEN	Determine a string's logical length.
ICNSMPLS	Convert a standard string to a nonstandard string.
ICNSMSTD	Convert a nonstandard string to string standard representation.
ICNSMSUB	Return a substring of a specified length.
ICNSMTRM	Trim trailing blanks from a string.

Convert number strings to numbers:

ICNTCBTA	Convert a bit to an arithmetic number.
ICNTCBTC	Convert a bit to a decimal string.
ICNTCDEC	Convert a decimal string to a number.
ICNTCDTC	Convert a number to a decimal character string.
ICNTCDTH	Convert a number to a hexadecimal string.
ICNTCHEX	Convert a hexadecimal string to a number.

Process a generation definition:

ICNIPGKI	Get user-coded keyword value.
ICNUSDTG	Get date and time of generation.
ICNUSGKI	Get keyword values.
ICNUSKAD	Add a keyword to the generation definition.
ICNUSKRP	Replace a keyword in the generation definition.
ICNUSNEW	Get the NEWNAME value.
ICNUSRNA	Get network addresses.
ICNUSSTA	Activate a group of statements.
ICNUSSTC	Add a comment to a generated statement group.
ICNUSSTI	Set an insertion point for a statement group.
ICNUSSTS	Save a definition statement.
ICNUSTAD	Add an entry to the table storage facility.
ICNUSTRT	Get data from the table storage facility.
ICNUSTUP	Update the table storage facility.

Search the SYSLIB data set for a member:

ICNUSLIB	Search the SYSLIB data set for a member.

NDF Internal Utility Descriptions

The descriptions of the NDF internal utilities have the following general form:

XVT Location	Parameters
X'7C'	*in_number* *out_decstring*

In this description, **XVT Location** is the offset in the transfer vector table (XVT) where the utility's start address is located. **Parameters** lists the parameters passed to the utility. All parameters are required unless enclosed in square brackets [ ]. Any utility that has `in_argx` as a parameter uses a variable-length parameter list. A description of the parameters follows the table.

The description of each utility includes a brief example of how to call the utility. These examples use a symbolic format to demonstrate the order of parameters and their values. Actual utility calls are written in OS/VS, DOS/VSE, or VM/370 Assembler Language or another equivalent language.

For more information on variable-length parameter lists, the linkage conventions for utility calls, and the NDF string handling system, refer to Chapter 4, "Creating and Using NDF Generation Applications," in *NCP and SSP Customization Guide*.

ICNCVLAB—Validate Statement Symbols

The ICNCVLAB utility validates a statement symbol and returns a status code. It tests for proper length, ensures that the characters are valid, and checks that the statement symbol has not already been defined in the generation definition. If all these results are positive, ICNCVLAB sets *out_status* to 0; otherwise, *out_status* contains a nonzero return code, which is explained below. If the symbol is not valid and *in_ncp_print* is set to 1, NDF prints an error message.

The utility saves all valid symbols to make sure that future symbols do not duplicate a value. It also records the symbol and the line number of the generation definition where the statement begins in a data structure for later use in creating the cross-reference.

Use ICNCVLAB to validate statement symbols for user-defined statements only. Do not use ICNCVLAB to validate statement symbols for NCP statements; NDF also uses ICNCVLAB, and errors for duplicate statement symbols would result.

Syntax

XVT Location	Parameters
X'04'	*in_symbol* *in_ncp_print* *out_status*

Parameters

in_symbol

Function Specifies the statement symbol to be validated.

Format Sixteen-character string in string standard representation.

in_ncp_print

Function Specifies whether ICNCVLAB should generate an NCP error message if the symbol is not valid.

Format One-byte flag. Bit 0 is checked.

Remarks If *in_ncp_print* is set to 1, NDF prints an error message if the symbol is not valid.

out_status

Function Specifies a return code.

Format Fullword integer.

Remarks The status values are as follows:

Status Value	Meaning
0	Symbol passed all validity checks.
1	Symbol has a length of 0.
2	Symbol is longer than 8 characters.
4	First character is not a letter, @, or #.
6	Symbol is too long and the first character is not valid.
8	First character is $.
10	Symbol is too long and the first character is $.
16	A character other than the first character is not a letter, digit, @, #, or $.
18	Symbol is too long, and a character other than the first is not a valid character.
20	First character and some other character are both not valid characters.
22	Symbol is too long and the first and some other character are not valid characters.
24	First character is $ and some character other than the first is not a valid character.
26	Symbol is too long, the first character is $, and another character is not a valid character.
32	Symbol is a duplicate.

Example

```
CALL ICNCVLAB('SYMBO %','1'B,OUTSTAT)
```

Because the symbol is valid, OUTSTAT is set to 0 by ICNCVLAB.

ICNCVRNG—Validate a Numeric String

The ICNCVRNG utility validates a number string and returns a value and a status code of 0 if the value meets a conversion check and falls within the specified minimum and maximum values. If the value is not valid and the print-error flag is set to 1, NDF generates an error message.

ICNCVRNG validates both decimal and hexadecimal number strings.

Syntax

XVT Location	Parameters
X'08'	*in_keyname* *in_value* *in_min* *in_max* *in_type* *in_ncp_print* *out_value* *out_status*

Parameters

in_keyname

Function Specifies the current keyword name. It is used in any error messages generated by ICNCVRNG.

Format Sixteen-character string in string standard representation.

in_value

Function Specifies the number string to be checked.

Format A string in string standard representation. The maximum length is 256 characters.

in_min

Function Specifies the minimum allowable value of *in_value* after it is converted.

Format Fullword integer.

in_max

Function Specifies the maximum allowable value of *in_value* after it is converted.

Format Fullword integer.

in_type

Function Specifies the type of value to verify.

Format Fullword integer.

Remarks An *in_type* value of 4 indicates a hexadecimal number; 5 indicates a decimal number.

in_ncp_print

Function Sets the flag to indicate whether ICNCVRNG should generate an error message if the value is not valid.

Format One-byte flag. Bit 0 is checked.

Remarks If *in_ncp_print* is 0, NDF will not print an error message if a conversion or range-check error occurs.

out_value

Function Returns the converted and validated value.

Format Fullword integer.

out_status

Function Returns the status code.

Format Fullword integer.

Remarks The *out_status* parameter returns 0 if the value passes the range and validity checks; otherwise, it returns a status value specifying the error, as follows:

Status Value	Meaning
0	Value passes all validity checks.
1	*in_type* is not valid.
9	*in_value* is not within the required range.
21	The conversion of *in_value* failed.
56	*in_value* is a null string.

Example

```
CALL ICNCVRNG('KEY1%','12%',10,15,5,'1'B,OUT_VALUE,OUT_STATUS)
```

Because *in_value* is within the range and is a valid number, *out_value* is set to 12 and *out_status* is set to 0.

ICNCVTOK—Validate a Character String

The ICNCVTOK utility validates a character string. It returns a 0 status code if the string meets the length and character tests. If the string is not valid and the *in_ncp_print* flag is set to one, ICNCVTOK generates an NCP error message.

Syntax

XVT Location	Parameters
X'0C'	*in_keyname* *in_string* *in_max* *in_type* *in_ncp_print* *out_status*

Parameters

in_keyname

Function Specifies the current keyword name. It is used in any error messages ICNCVTOK generates.

Format Character string in string standard representation. The maximum length is 16 characters.

in_string

Function Specifies the string to be checked.

Format Character string in string standard representation. The maximum length is 256 characters.

in_max

Function Specifies the maximum number of characters the string may have.

Format Fullword integer.

in_type

Function Specifies the type of string to verify.

Format Fullword integer.

Remarks *in_type* can have the following values:

Value	Meaning
1	Assembler string check. String cannot start with $, or the digits 0 to 9.
2	File name string check. String cannot start with the digits 0 to 9.
3	Any type. Strings can start with any character.
4	Hexadecimal string check. String can consist only of the characters A to F and 0 to 9. This check is used to validate literal data sequences for CUID and any other polling operations that require hexadecimal strings longer than 8 characters.

in_ncp_print

Function Contains the flag to indicate whether NDF should generate error messages if the string is not valid.

Format One-byte flag. Bit 0 is checked.

Remarks If *in_ncp_print* is set to 1, NDF issues an error message if the string is not valid.

out_status

Function Returns the status code from ICNCVTOK.

Format Fullword integer.

Remarks *Out_status* returns 0 if the string is valid, or it returns a nonzero code indicating the error if the string is not valid. The status values are as follows:

Status Value	Meaning
0	String passes all validity checks.
1	String has a length of 0.
2	String is longer than 8 characters.
4	String character is not a letter, @, or #.
6	String is too long, and the first character is not valid.
8	String character is $.
10	String is too long and the first character is $.

Status Value	Meaning
16	A character other than the first character is not a letter, digit, @, #, or $.
18	String is too long, and a character other than the first is not a valid character.
20	First character and some other character are both not valid characters.
22	String is too long and the first and some other character are both not valid characters.
24	The first character is $, and some character other than the first is not a valid character.
26	String is too long, the first character is $, and another character is not a valid character.
32	*In_type* was not 1, 2, 3, or 4.

Example

```
CALL ICNCVTOK('KEY1%','STRING%',8,1,'1'B,OUTSTAT)
```

The string meets the length requirement, and is a proper assembler string (*in_type* is 1, indicating an assembler string), so *out_status* is set to 0.

ICNERPST—Post an Error Message

Syntax

ICNERPST prints an error message in the NDF listing. It also does error summary processing, such as incrementing error counts and formatting error messages.

XVT Location	Parameters
X'10'	*in_severity* *in_msg* *in_argx*

Parameters

in_severity

Function Contains the error severity code.

Format Fullword integer.

Remarks Table 16 shows the numeric value of the severity codes and their meanings. The table also shows the severity message associated with each severity value. Note that a severity of 4 or greater causes NDF to validate the generation only, and NDF does not perform any further processing.

Table 16. Error Message Severity Codes

Severity Message	Severity Value	Meaning
Info	0	Informational message only. No programmer action is required.
Warning	4	An error has occurred for which NDF has taken corrective action by assuming a default keyword value or by ignoring the value supplied. The generation process is terminated after validation of the generation definition.
Error	8	A user error has occurred for which NDF cannot assume a value or ignore the value supplied. The generation process is terminated after validation of the generation definition.
Ten	10	A fatal user error has been detected. The generation process is terminated.
Severe	12	A system error has occurred. NDF produces a procedure traceback. The generation process is terminated after validation of the generation definition.
Fatal	16	A fatal system error has occurred. A procedure traceback is printed and the generation process is terminated.

in_msg

Function Contains the error message.

Format Character string in string standard representation. The maximum length is 256 bytes.

Remarks The error message has the following format:

```
ICN234I Body of message
```

The body of the message can contain control codes to control the insertion of text from the variable parameter list *in_argx*. For information about control codes, see "NDF String Handling" in *NCP and SSP Customization Guide*.

in_argx

Function Provides a variable number of filler parameters that are either inserted into the error message or used to control the formatting of the error message.

Format Variable-length parameter list.

Example

```
CALL ICNERPST(8,'ICN999E /S = /F IS NOT VALID, /S IS ASSUMED%',
              'USRSTMT1%',3745,'UNIT%')
```

The control codes insert the arguments into the message text to produce the following message:

```
*ERROR*  ICN999E   8   USERSTMT1 = 3745 IS NOT VALID, UNIT IS ASSUMED
```

The *ERROR* and severity value 8 are generated by ICNERPST from the value of *in_severity*. The message is generated as follows:

1. Control code /S gets the first subargument of *in_argx* and inserts it after the message number.
2. The first part of the text in the call is printed (=).
3. Control code /F gets the fullword integer 3745 and inserts it.
4. Control code /S gets the last subargument of *in_argx* ('UNIT%').
5. The rest of the text in the call is printed.

ICNIPGKI—Get a User-Coded Keyword Value

ICNIPGKI gets the user-coded value for any keyword on the current statement.

Syntax

XVT Location	Parameters
X'C0'	*key_name* *key_val* *key_len* *parsed* *key_cnt*

Parameters

key_name

Function Contains the name of the keyword whose value you are retrieving.

Format Character string in string standard representation.

key_val

Function Returns the value entered for the specified keyword.

Format Character string in string standard representation.

Remarks If you do not code the keyword, *key_val* contains a null string ('%').

key_len

Function Returns the length of *key_val*.

Format Fullword integer.

Remarks If you do not code the keyword, *key_len* is 0.

parsed

Function Contains parsed representation of *key_val*; used for multiple suboperands.

Format Array with the following structure:

```
DCL 1 parsed(100)
```

Remarks Characters such as parentheses are not included in *parsed*.

key_cnt

Function Returns the number of suboperands contained in *parsed*.

Format Fullword integer.

Remarks If you do not code the keyword, *key_cnt* is 0.

Examples

```
CALL ICNIPGKI('LNCTL%',KEY_VAL,KEY_LEN,PARSED,KEY_CNT)
```

If you code LNCTL=SDLC, ICNIPGKI returns the following values:

```
key_val=SDLC%
key_len=4
parsed val_len(1)=4
         val_ptr(1) points to SDLC
key_cnt=1
```

```
CALL ICNIPGKI('USERID%',KEY_VAL,KEY_LEN,PARSED,KEY_CNT)
```

If you code USERID=5668854,ECLRBDT,NORECMS,U,ECLNMVX, ICNIPGKI returns the following values:

```
key_val='(5668854,ECLRBDT,NORECMS,U,ECLNMVX)%'
key_len=35
parsed val_len(1)=7
         val_ptr(1) points to 5668854
parsed val_len(2)=7
         val_ptr(2) points to ECLRBDT
parsed val_len(3)=7
         val_ptr(3) points to NORECMS
parsed val_len(4)=1
         val_ptr(4) points to U
parsed val_len(5)=7
         val_ptr(5) points to ECLNMVX
key_cnt=5
```

ICNLEPTN—Post a Link-Edit Statement

With the ICNLEPTN utility, you can post link-edit statements. The statements determine the construction of the controller load modules.

ICNLEPTN posts the statements to a specific list, called a cluster. There is one cluster for each block of storage allocated for user-written controller code. Each cluster corresponds to GENEND keywords, as shown:

Cluster ID	GENEND Definition Statement Keywords
1	SRCLO
2	Should not be used. If used, the object modules are placed in the lower 64KB of controller storage.
3	INCL2LO and ORDL2LO keywords
4	INCL2HI and ORDL2HI keywords
5	INCLO and ORDLO keywords
6	SRCHI, INCHI, and ORDHI keywords
7	INCINIT and ORDINIT keywords
8	Reserved
9	KEY0INC and KEY0ORD keywords
10	REPLACE keywords for each control block assembled in the table 1 assembly

You must post the link-edit statements in the order defined for the CSECTs in the load module. Your generation application must post its link-edit statements before or during GENEND statement processing.

Note: If your generation application posts link-edit statements to cluster 6, it must not post them until after NDF has completed processing the BUILD statement.

Syntax

XVT Location	Parameters
X'14'	*in_cluster_id* *in_mvs* *in_le_command*

Parameters

in_cluster_id

Function Identifies the cluster to which the link-edit command is posted.

Format Fullword integer.

in_mvs

Function Specifies the operating system this statement is used for.

The flag *in_mvs* indicates that the link-edit is for either MVS or VM, not VSE. ICNLEPTN posts statements only if the link-edit type indicated by the flag matches the type specified on the TYPSYS keyword on the BUILD definition statement.

Format Fullword integer.

Remarks A value of 0 indicates MVS or VM; a value of 1 indicates VSE.

in_le_command

Function Specifies the link-edit statement to be posted.

Format Character string in string standard representation. It must be a valid link-edit statement.

Example

```
CALL ICNLEPTN(1,0,'INCLUDE MYPDS(MEM)%')
```

This call posts the control statement `INCLUDE MYPDS(MEM)` to the first cluster if the system type is MVS or VM.

ICNLEPTS—Post Link-Edit Statements to Produce a Separate Load Module from Table 1

Syntax

The ICNLEPTS utility posts link-edit statements that control the generation of a load module from table 1 that is separate from the NCP controller load module. It can generate NCP and UWGA REPLACE cards and all other link-edit statements (INCLUDE, ORDER, ENTRY, NAME, and so on).

For different operating systems, your utility must meet the following requirements:

- For MVS and VM, your utility must post the following link-edit statement as its first link-edit command in order to include code added to the table assembly input:

  ```
  INCLUDE SYSPUNCH(ICNTABL1)
  ```

  This is for code added to table 1.
- For MVS and VM, you must post REPLACE statements for all CSECTs punched into the table assemblies.
- For VSE, you must post as the first link-edit statement `PHASE modname,+0` where `modname` is the name of your load module.

Note: This utility does not replace the functions of ICNLEPTN, which posts link-edit statements for code that resides in NCP.

XVT Location	Parameters
X'B4'	*in_uwga_name* *in_typsys* *in_replace* *in_le_card*

Parameters

in_uwga_name

Function Specifies the name of a load module as coded on the USERGEN keyword of the OPTIONS definition statement. It is used for diagnostic purposes by NDF.

Format Sixteen-character string in string standard representation.

in_typsys

Function Identifies the operating system for which this link-edit statement is applicable.

Format Fullword integer.

Remarks A value of 0 indicates MVS or VM; a value of 1 indicates VSE.

in_replace

Function Identifies whether this is a REPLACE statement.

Format Fullword integer.

Remarks A value of 0 indicates that this is not a REPLACE statement; a value of one indicates that it is.

in_le_card

Function Specifies the card image of the link-edit statement to be posted.

Format String standard representation. The maximum length is 256 characters.

Remarks The card image is expected to be in valid link-edit statement format.

Example

```
CALL INCLEPTS('UWGA1%',0,0,' INCLUDE MYPDS(MEMBER)%')
```

The call posts the link-edit statement for MVS or VM for a separate load module.

ICNLEPT2—Post Link-Edit Statements to Produce a Separate Load Module from Table 2

The ICNLEPT2 utility posts link-edit statements that control the generation of a load module from table 2 that is separate from the NCP controller load module. It can generate NCP and UWGA REPLACE cards and all other link-edit statements (INCLUDE, ORDER, ENTRY, NAME, and so on).

This utility may not be used by VSE users, since NDF running under VSE punches all its table assembly code to table 1. For MVS and VM, your utility must meet the following requirements:

- Post the following link-edit statement as its first link-edit command in order to include code added to the table assemble input:

  ```
  INCLUDE SYSPUNCH(ICNTABL2)
  ```

  This is for code added to table 2.
- Post REPLACE statements for all CSECTs punched into the table assemblies.

Note: This utility does not replace the functions of ICNLEPTN, which posts link-edit statements for code that resides in NCP.

Syntax

XVT Location	Parameters
X'CC'	*in_uwga_name* *in_typsys* *in_replace* *in_le_card*

Parameters

in_uwga_name

Function Specifies the name of a load module as coded on the USERGEN keyword of the OPTIONS definition statement. It is used for diagnostic purposes by NDF.

Format Sixteen-character string in string standard representation.

in_typsys

Function Identifies the operating system for which this link-edit statement is applicable.

Format Fullword integer.

Remarks A value of 0 indicates MVS or VM; a value of 1 indicates VSE.

in_replace

Function Specifies whether this is a REPLACE statement.

Format Fullword integer.

Remarks A value of 0 indicates that this is not a REPLACE statement; a value of 1 indicates that it is.

in_le_card

Function Specifies the card image of the link-edit statement to be posted.

Format String standard representation. The maximum length is 256 characters.

Remarks The card image is expected to be in valid link-edit statement format.

Example

```
CALL INCLEPT2('UWGA1%',0,0,' INCLUDE MYPDS(MEMBER)%')
```

The call posts the link-edit statement for MVS or VM for a separate load module.

ICNOBPUN—Punch Table 1 Assembly Source

The ICNOBPUN utility produces an assembly record to go into the table 1 assembly source. The record is generated from the input string and from the parameters in the argument string.

Note: You must create your own CSECT for your user control blocks. You should not use $SRCLO or $SRCHI for control blocks generated using ICNOBPUN.

Syntax

XVT Location	Parameters
X'18'	*in_control_str* *in_argx*

Parameters

in_control_string

Function Contains a string of string-standard-representation control codes that control the construction of the assembly record from the values in the parameter list *in_argx*.

Format Character string in string standard representation. The maximum string length is 256 characters.

Remarks For information about control codes, see "NDF String Handling" in *NCP and SSP Customization Guide*.

in_argx

Function Contains the parameter list of elements that are used to construct the assembly record.

Format Variable-length parameter list.

Example

```
CALL ICNOBPUN('USER% /T5 /S','CSECT%')
```

This call produces the following assembly record:

```
USER     CSECT
```

ICNOBPU2—Punch Table 2 Assembly Source

The ICNOBPU2 utility produces an assembly record to go into the table 2 assembly source. The record is generated from the input string and from the parameters in the argument string.

Note: You must create your own CSECT for your user control blocks. You should not use $SRCLO or $SRCHI for control blocks generated using ICNOBPU2.

Syntax

XVT Location	Parameters
X'C8'	*in_control_str* *in_argx*

Parameters

in_control_string

Function Contains a string of string-standard-representation control codes that control the construction of the assembly record from the values in the parameter list *in_argx*.

Format Character string in string standard representation. The maximum string length is 256 characters.

Remarks For information about control codes, see "NDF String Handling" in *NCP and SSP Customization Guide*.

in_argx

Function Contains the parameter list of elements that are used to construct the assembly record.

Format Variable-length parameter list.

Example

```
CALL ICNOBPU2('USER% /T5 /S','CSECT%')
```

This call produces the following assembly record:

```
USER     CSECT
```

ICNRPFMT—Format and Print a Character String

The ICNRPFMT utility formats a character string using the values in the argument list *in_argx* and the control codes in *in_control_str*. It puts this string in the NDF listing. If the string is too long to fit on one line in the listing, ICNRPFMT breaks it at a blank, if possible, and indents the continuation lines.

Syntax

XVT Location	Parameters
X'1C'	*in_control_str* *in_argx*

Parameters

in_control_str

Function Contains control codes that control the construction of the output line from the elements in the parameter list *in_argx*.

Format Character string in string standard representation.

Remarks For information about control codes, see "NDF String Handling" in *NCP and SSP Customization Guide*.

in_argx

Function Contains the parameter list of elements that are used to construct the listing string.

Format Variable-length parameter list.

Example

```
CALL ICNRPFMT('/S/5T/S/72T/F%','*%','This is a comment line%',0001)
```

The control string uses the elements of *in_argx* to make the following output line:

```
*   This is a comment line                                          1
```

ICNRPINF—Format and Print a Default or Inheritance Message

The ICNRPINF utility inserts tab characters in a default or inheritance message.

Syntax

XVT Location	Parameters
X'20'	*in_string* *in_msg* *in_argx*

Parameters

in_string

Function Contains the character code indicating the type of default string.

Format Character string in string standard representation. The maximum length is 4 characters.

Remarks The character code for the default types is as follows:

C Keyword inherited from CLUSTER
D Keyword value is a default
G Keyword inherited from GROUP
L Keyword inherited from LINE
P Keyword inherited from PU
T Keyword inherited from TERMINAL

in_msg

Function Contains the control string for the default message.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks The final string must be in the form *name=value* where *name* is the name of a keyword and *value* is the default value. Either *name* or *value* can be constructed by using control characters and the values in *in_argx*.

For information about control codes, see "NDF String Handling" in *NCP and SSP Customization Guide*.

in_argx

Function Contains the elements to be used in constructing the default message.

Format Variable-length parameter list.

Example

```
CALL ICNRPINF('D%','/S=/F%','NTO.CWALL%',8)
```

This call produces the following default message:

```
D               NTO.CWALL=8
```

ICNRPPBT—Print a Trace Message for a Boolean Flag

The ICNRPPBT utility prints a trace message in the NDF listing. The message consists of an identifying message string followed by the character representation of a Boolean flag being traced.

Syntax

XVT Location	Parameters
X'24'	*in_msg* *in_boolean*

Parameters

in_msg

Function Contains the message identifying the value being traced.

Format Character string in string standard representation. The maximum length is 32 characters.

in_boolean

Function Contains the Boolean flag that is being traced.

Format One-byte flag.

Example

```
FLAG1='1'B

CALL ICNRPPBT('FLAG1%',FLAG1)
```

This call produces the following listing message:

```
    FLAG1='1'B
```

ICNRPPCH—Print a Trace Message for a Standard String

The ICNRPPCH utility prints a trace message in the NDF listing. The message consists of an identifying message string followed by the the string being traced. The traced string is printed on as many lines as necessary.

Syntax

XVT Parameters	
X'28'	*in_msg* *in_str*

Parameters

in_msg

Function Specifies the string identifying the value being printed.

Format Character string in string standard representation. The maximum length is 32 characters.

in_str

Function Specifies the string being traced.

Format Character string in string standard representation. The maximum length is 256 characters.

Example

```
IN_STR='VALUE OF STRING'

CALL ICNRPPCH('STRING1%',IN_STR)
```

This call produces the following listing message:

```
STRING1=VALUE OF STRING
```

ICNRPPCX—Print a Trace Message for a Nonstandard String

The ICNRPPCH utility prints a trace message in the NDF listing. The message consists of an identifying message string followed by the the string being traced. The traced string is printed on as many lines as necessary.

Syntax

XVT Location	Parameters
X'2C'	*in_msg* *in_str* *in_length*

Parameters

in_msg

Function Contains the string identifying the value being printed.

Format Character string in string standard representation. The maximum length is 32 characters.

in_str

Function Specifies the nonstandard string being traced.

Format Character string *not* in string standard representation. The maximum length is 256 characters.

in_length

Function Specifies the length of the string.

Format Fullword integer.

Example

```
CALL ICNRPPCX('STRING1%','VALUE OF STRING',15)
```

This call produces the following listing message:

```
STRING1=VALUE OF STRING
```

ICNRPPFX—Print a Trace Message for a Decimal Number

The ICNRPPFX utility prints a trace message in the NDF listing. The message consists of a descriptive string followed by the character representation of the decimal number.

Syntax

XVT Location	Parameters
X'30'	*in_msg* *in_num*

Parameters

in_msg

Function Specifies the string identifying the value being printed.

Format Character string in string standard representation. The maximum length is 32 characters.

in_num

Function Specifies the integer being traced.

Format Fullword integer.

Example

```
NUMBER1=256

CALL ICNRPPFX('NUMBER1%',NUMBER1)
```

This call produces the following message:

```
NUMBER1=256
```

ICNRPPHX—Print a Trace Message for a Hexadecimal Number

The INCRPPHX utility prints in the NDF listing a message string followed by the string representation of a 32-bit hexadecimal number.

Syntax

XVT Location	Parameters
X'34'	*in_msg* *in_num*

Parameters

in_msg

Function Specifies the string identifying the value being printed.

Format Character string in string standard representation. The maximum length is 32 characters.

in_num

Function Specifies the hexadecimal number being traced.

Format Fullword integer.

Example

```
IN_NUM=X'00FE'

CALL ICNRPPHX('HEXNUM%',IN_NUM)
```

This call produces the following message:

```
HEXNUM='00FE0000'X
```

ICNRPPNM—Print a Trace Message for the Value of a Number

The ICNRPPNM utility converts a fullword integer to a specified base and then generates its string representation. The string is padded on the left as specified, and then printed with an identifying string. You must specify the total length of the output string and the pad character. You must also specify whether the number is signed. If the field length specified is smaller than the string length, the string is not padded. The string is not truncated if it is longer than the field length.

Syntax

XVT Location	Parameters
X'38'	*in_msg* *in_num* *in_len* *in_sign* *in_pad* *in_base* *in_field*

Parameters

in_msg

Function Specifies the string identifying the number being traced.

Format Character string in string standard representation. The maximum length is 32 characters.

in_num

Function Contains the number being traced.

Format Fullword integer.

Remarks The number of bits in this number must be declared using *in_len*.

in_len

Function Specifies the number of bits in *in_num*.

Format Fullword integer.

Remarks If *in_num* is a signed number, then *in_len* is the number of bits in *in_num* plus 1.

in_sign

Function Specifies whether the number is signed.

Format One-byte flag. Bit 0 is checked.

Remarks When bit 0 is set to 1, it indicates a signed number.

in_pad

Function Specifies the padding character.

Format Single character; *not* in string standard representation.

in_base

Function Specifies the base in which the number is to be represented.

Format Fullword integer.

Remarks *in_base* must be in the range 2 to 32.

in_field

Function Specifies the length of the output string.

Format Fullword integer.

Remarks If *in_field* is smaller than the length of the output string, the string is not padded. ICNRPPNM does not truncate the output string if *in_field* is too small.

Example

```
CALL ICNRPPNM('NUMBER1%',-54,32,'1'B,' ',10,4)
```

This call produces the following message:

```
NUMBER1= -54
```

ICNRPTRC—Print a Trace Message for the Procedure Entry or Exit

The ICNRPTRC utility prints a message in the NDF listing saying that the specified procedure is being entered or exited.

Syntax

XVT Location	Parameters
X'3C'	*in_procedure_name* *in_entry_or_exit*

Parameters

in_procedure_name

Function Specifies the name of the procedure being traced.

Format Character string in string standard representation. The maximum length is 16 characters.

in_entry_or_exit

Function Specifies whether this call is a procedure entry or exit.

Format One-byte flag. Bit 0 is checked.

Remarks When bit 0 is set to 0, it indicates procedure entry; when bit 0 is set to 1, it indicates procedure exit.

Example

```
CALL ICNRPTRC('USROUTIN%','1'B)
```

This call produces the following message:

```
++EXITING USROUTIN
```

ICNSMAFT—Return the Substring Following a Target String

The ICNSMAFT utility returns a substring of the input string following the first occurrence of the search string. The output string is set to null if the search string is not found or if the search string is '%'.

Syntax

XVT Location	Parameters
X'40'	*in_str* *in_search_str* *out_str*

Parameters

in_str

Function Specifies the input string from which the substring is taken.

Format Character string in string standard representation. The maximum length is 256 characters.

in_search_str

Function Specifies the search string.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks If *in_search_str* is '%', *out_str* is set to null.

out_str

Function Returns the rest of *in_str* following the search string.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks If the search string is not found or if the search string is '%', *out_str* is set to null.

Example

```
CALL ICNSMAFT('ABCDEF%','DE%',OUT_STR)
```

This call sets *out_str* to F%.

ICNSMALN—Determine the Actual Length of a String

The ICNSMALN utility returns the actual length of the input string. The actual length is the number of characters needed to express the string in string standard representation. The actual length includes any control or escape characters.

Syntax

XVT Location	Parameters
X'44'	*in_str* *out_len*

Parameters

in_str

Function Specifies the string to be sized.

Format Character string in string standard representation. The maximum length is 256 characters.

out_len

Function Returns the actual length of the string.

Format Fullword integer.

Remarks The value of *out_len* cannot be 0 since the minimum string in string standard representation is an end delimiter (%).

Example

```
CALL ICNSMALN('AB/QC%',OUT_LEN)
```

This call sets *out_len* to 6.

ICNSMBEF—Return the Substring Preceding a Target String

The ICNSMBEF utility returns that portion of the input string before the target string. If the target string is not found or the target string is '%', the output string is set to the input string.

Syntax

XVT Location	Parameters
X'48'	*in_str* *in_target_str* *out_str*

Parameters

in_str

Function Specifies the input string from which the substring is taken.

Format Character string in string standard representation. The maximum length is 256 characters.

in_target_str

Function Specifies the target string.

Format Character string in string standard representation. The maximum length is 256 characters.

out_str

Function Returns the portion of the input string located before the target string.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks If the target string is not found or if the target string is '%', *out_str* is set to *in_str*.

Example

```
CALL ICNSMBEF('ABCDEF%','DE%',OUT_STR)
```

This call sets *out_str* to 'ABC%'.

ICNSMCAT—Concatenate Strings

The ICNSMCAT utility concatenates the strings in the input string parameter list and puts the resulting string in the output string.

Syntax

XVT Location	Parameters
X'4C'	*out_str* *in_str_list*

Parameters

out_str

Function Returns the concatenated string.

Format Character string in string standard representation. The maximum length is 256 characters.

in_str_list

Function Variable-length strings to be concatenated.

Format Variable-length parameter list of character strings in string standard representation.

Remarks If only one input string is specified, that string is copied to the output string.

Example

```
CALL ICNSMCAT(OUT_STR,'AB%','CD%','EF%')
```

This call sets *out_str* to 'ABCDEF%'.

ICNSMCMP—Compare Strings

The ICNSMCMP utility searches a list of input strings for the first match to the target string. If it finds a match, it returns the position in the list of that string. If it does not find a match, *out_match_num* is set to 0.

Syntax

XVT Location	Parameters
X'50'	*out_match_num* *in_target_str* *in_str_list*

Parameters

out_match_num

Function Returns the position in the list of the first matching string.

Format Fullword integer.

Remarks If no match is found, 0 is returned.

in_target_str

Function Specifies the target string.

Format Character string in string standard representation. The maximum length is 256 characters.

in_str_list

Function Specifies a list of strings against which the target string is compared.

Format Variable-length parameter list of character strings in string standard representation.

Example

```
CALL ICNSMCMP(OUT_MATCH_NUM,'ABC%','DEF%','A%','ABC%')
```

This call sets *out_match_num* to 3.

ICNSMFMT—Format a Value into a Character String

The ICNSMFMT utility formats a character string from a fixed point number, character string, or Boolean value. A control string uses control codes in string standard representation to generate the output string. For information about control codes, see "NDF String Handling" in *NCP and SSP Customization Guide*.

Syntax

XVT Location	Parameters
X'54'	*out_str* *in_control_str* *in_arg_list*

Parameters

out_str

Function Returns the formatted string.

Format Character string in string standard representation. The maximum length is 256 characters.

in_control_str

Function Controls the generation of the output string.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks This string can be a string, control codes, or a combination of the two.

in_arg_list

Function Specifies the list of elements used to construct the output string.

Format Variable-length parameter list.

Example

```
CALL ICNSMFMT(OUT_STR,'/S/T5/F/T5/S%',
              'LINE NUMBER %',6,'MUST BE FIXED%')
```

This call sets *out_str* to LINE NUMBER 6 MUST BE FIXED%.

ICNSMFND—Find a String

The ICNSMFND utility finds the first occurrence of the target string in the input string.

Syntax

XVT Location	Parameters
X'58'	*in_str* *in_target_str* *out_pos*

Parameters

in_str

Function Specifies the string to be searched.

Format Character string in string standard representation. The maximum length is 256 characters.

in_target_str

Function Specifies the string that is searched for in *in_str*.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks If *in_target_str* is '%', *out_pos* is set to 0.

out_pos

Function Returns the position of the first occurrence of the target string in the input string.

Format Fullword integer.

Remarks If the target string is not found or is '%', *out_pos* is set to 0.

Example

```
CALL ICNSMFND('ABCDEF%','C%',OUT_POS)
```

This call sets *out_pos* to 3.

ICNSMLEN—Determine the Logical Length of a String

The ICNSMLEN utility returns the logical length of the string. The logical length is the length of the string after the effects of string standard control codes and special characters have been processed.

Syntax

XVT Location	Parameters
X'5C'	*in_str* *out_len*

Parameters

in_str

Function Specifies the string to be processed.

Format Character string in string standard format. The maximum length is 256 characters.

out_len

Function Returns the logical length of the input string.

Format Fullword integer.

Example

```
CALL ICNSMLEN('ABCD¬¬EF%',OUT_LEN)
```

This call sets *out_len* to 7. The normal string version of *in_str* is 'ABCD¬EF'. The escape character ¬ and the end-of-string character % are not counted.

ICNSMPLS—Convert a Standard String to a Nonstandard String

The ICNSMPLS utility converts a string in string standard representation to its equivalent normal string. It also returns the length of the resulting normal string.

Syntax

XVT Location	Parameters
X'60'	*in_str* *out_norm_str* *out_len*

Parameters

in_str

Function Specifies the string to be converted to non-string standard representation.

Format Character string in string standard representation. The maximum length is 256 characters.

out_norm_str

Function Returns a normal string.

Format Character string in non-string standard representation. The maximum length is 256 characters.

Remarks This string has a maximum length of 256 characters.

out_len

Function Returns the number of characters in the resulting normal string.

Format Fullword integer.

Example

```
CALL ICNSMPLS('ABC¬%DEF%',OUT_NORM_STR,OUT_LEN)
```

This call sets *out_norm_str* to 'ABC%DEF' and *out_len* to 7.

ICNSMSTD—Convert a Nonstandard String to String Standard Representation

The ICNSMSTD utility converts a string in non-string standard representation to a string in string standard representation. The routine inserts an escape character before each special character in the string and adds an end-of-string character at the end of the string.

Syntax

XVT Location	Parameters
X'64'	*in_norm_str* *in_len* *out_str*

Parameters

in_norm_str

Function Specifies the string to be converted to string standard representation.

Format Character string in non-string standard representation. The maximum length is 256 characters.

in_len

Function Specifies the number of characters in the input string that are to be converted. Normally this is the length of the input string.

Format Fullword integer.

out_str

Function Returns the string standard equivalent of the input string.

Format Character string in string standard representation. The maximum length is 256 characters.

Remarks ICNSMSTD returns only the first *in_len* characters of the input string.

Example

```
CALL ICNSMSTD('ABC%DEF',7,OUT_STR)
```

This call sets *out_str* to 'ABC¬%DEF%'.

ICNSMSUB—Return a Substring of a String

The ICNSMSUB utility returns a substring of the input string starting at *in_start_pos*, with a length of *in_len*. Note that an escape character (¬) paired with the character immediately following is considered one logical character when calculating the length of the substring.

Syntax

XVT Location	Parameters
X'68'	*in_str* *in_start_pos* *in_len* *out_str*

Parameters

in_str

Function Specifies the string from which the substring is to be taken.

Format Character string in string standard representation. The maximum length is 256 characters.

in_start_pos

Function Specifies the position in the input string where the substring begins.

Format Fullword integer.

Remarks The first character of a string is character 1, not character 0.

in_len

Function Specifies the length of the substring.

Format Fullword integer.

out_str

Function Returns the substring.

Format Character string in string standard representation. The maximum length is 256 characters.

Example

```
CALL ICNSMSUB('ABC¬%DEF',3,4,OUT_STR)
```

This call sets *out_str* to 'C¬%DE'. Note that the escape character with the percent sign ('¬%') counts as 1 character.

ICNSMTRM—Trim Trailing Blanks from a String

The ICNSMTRM utility removes any trailing blanks from the input string and returns the result in the output string.

Syntax

XVT Location	Parameters
X'6C'	*in_str* *out_str*

Parameters

in_str

Function Specifies the string from which trailing blanks are to be stripped.

Format Character string in string standard representation. The maximum length is 256 characters.

out_str

Function Returns the resulting string.

Format Character string in string standard representation. The maximum length is 256 characters.

Example

```
CALL ICNSMTRM('ABCDEF  %',OUT_STR)
```

This call sets *out_str* to `'ABCDEF%'`.

ICNTCBTA—Convert a Bit to an Arithmetic Number

The ICNTCBTA utility converts a bit to a fullword integer.

Syntax

XVT Location	Parameters
X'70'	*in_bit* *out_value*

Parameters

in_bit

Function Specifies the bit to be converted to an integer.

Format Single binary digit.

out_value

Function Specifies the integer representation of the bit.

Format Fullword integer.

Example

```
CALL ICNTCBTA(B'1',OUT_VALUE)
```

This call sets *out_value* to the fullword integer 1.

ICNTCBTC—Convert a Bit to a Decimal String

The ICNTCBTC utility converts a bit to a character string of its decimal value.

Syntax

XVT Location	Parameters
X'74'	*in_bit* *out_string*

Parameters

in_bit

Function Specifies the bit to be converted to a decimal string.

Format Single binary digit.

out_string

Function Specifies the string representation of the decimal value of the bit.

Format Character string with a length of 16 characters.

Example

```
CALL ICNTCBTC(B'1',OUT_STRING)
```

This call sets *out_string* to the string '1%'.

ICNTCDEC—Convert a Decimal String to a Number

The ICNTCDEC utility determines the numeric value represented by the decimal digits of the input string. If the input string contains characters other than the digits 0 to 9, or if the resulting value is outside the range $-(2^{31}-1)$ to $(2^{31}-1)$ inclusive, the error flag is set to 1.

Syntax

XVT Location	Parameters
X'78'	*in_decstring* *out_value* *out_error*

Parameters

in_decstring

Function Specifies the string to be converted to a numeric value.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The string can contain only the characters 0 to 9.

out_value

Function Returns the numeric value of the decimal string.

Format Fullword integer.

out_error

Function Specifies whether the input string was a valid decimal number, or if the value was in the proper range.

Format One-byte flag. Bit 0 is checked.

Remarks If bit 0 is set to 1, the conversion was not valid.

Example

```
CALL ICNTCDEC('256%',OUT_VALUE,OUT_ERROR)
```

This call sets *out_value* to 256 and *out_error* to '0'B.

ICNTCDTC—Convert a Number to a Decimal String

The ICNTCDTC utility converts a numeric value to a string representing the decimal value of the number. The routine generates a minus sign for negative numbers.

Syntax

XVT Location	Parameters
X'7C'	*in_number* *out_decstring*

Parameters

in_number

Function Specifies the number to be converted to a decimal string.

Format Fullword integer.

Remarks The number must be an integer value in the range $-(2^{31}-1)$ to $(2^{31}-1)$ inclusive.

out_decstring

Function Returns the character representation of the decimal value of the input number.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The string can contain only the characters 0 to 9 and the minus sign ('-').

Example

```
CALL ICNTCDTC(-256,OUT_DECSTRING)
```

This call sets *out_decstring* to '-256%'.

ICNTCDTH—Convert a Number to a Hexadecimal String

The ICNTCDTH utility converts the input number to a string representation of its hexadecimal value.

Syntax

XVT Location	Parameters
X'80'	*in_number* *out_hexstring*

Parameters

in_number

Function Specifies the number to be converted into a hexadecimal string.

Format Fullword integer.

Remarks The number must be an integer in the range $-(2^{31}-1)$ to $(2^{31}-1)$ inclusive.

out_hexstring

Function Returns the character representation of the hexadecimal value of the input string.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The output string can contain only the characters A to F and 0 to 9.

Example

```
CALL ICNCDTH(256,OUT_HEXSTRING)
```

This call sets *out_hexstring* to '100%'.

ICNTCHEX—Convert a Hexadecimal String to a Number

The ICNTCHEX utility converts the input hexadecimal string to its numeric value. If the input string does not represent a valid hexadecimal number, or if the resulting value is outside the range $-(2^{31}-1)$ to $(2^{31}-1)$ inclusive, the error flag is set to 1.

Syntax

XVT Location	Parameters
X'84'	*in_hexstring* *out_value* *out_error*

Parameters

in_hexstring

Function Specifies the hexadecimal string to be converted to a number.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The string may contain only the characters A to F and 0 to 9 and may have no more than 8 hexadecimal digits.

out_value

Function Returns the numeric value of the input hexadecimal string.

Format Fullword integer.

out_error

Function Specifies whether the conversion was valid.

Format One-byte flag. Byte 0 is checked.

Remarks If the hexadecimal string does not represent a valid hexadecimal number, or if the resulting value is outside the range $-(2^{31}-1)$ to $(2^{31}-1)$ inclusive, the error flag is set to 1.

Example

```
CALL ICNTCHEX('1C 4F%',OUT_VALUE,OUT_ERROR)
```

This call sets *out_value* to 7247 and *out_error* to '0'B.

ICNUSDTG—Get the Date and Time of Generation

The ICNUSDTG utility returns the date and time of generation contained in the DTG NCP control block created by NDF.

Syntax

XVT Location	Parameters
X'A8'	*out_date* *out_time*

Parameters

out_date

Function Contains the generation date.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The date is returned in the format yy/mm/dd.

out_time

Function Contains the time of generation.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The time is returned in the format hh:mm:ss.

Example

If the load module was generated on June 12, 1993, at 2:30 p.m., then the call:

```
CALL ICNUSDTG(out_date, out_time)
```

sets *out_date* to 93/06/12 and *out_time* to 14:30:38.

ICNUSGKI—Get Keyword Values

The ICNUSGKI utility returns all keyword values associated with a given keyword on the current statement. You can use this routine only for post-processing routines.

Values returned adhere to the following rules:

- All values are in string standard representation. They can be up to 256 characters long.
- Only keyword values coded at their lowest statement level are returned. For example, keywords that can be coded on the GROUP, LINE, and TERMINAL statements will be returned only if TERMINAL is the current statement, unless this keyword is required at a higher level.

  If a keyword is required at a higher level in certain configurations, and is coded at a lower level for the current configuration, it is returned as a null value and no errors are generated.
- Default values and values inherited from other statements will be returned as if they were coded on the current statement.
- Keywords that have no coded value and no default for the current statement will be returned as a null value.
- Keywords that have no coded value and are not immediately defaulted by NDF will be defaulted and returned during GENEND processing. If ICNUSGKI is coded for a keyword that will be defaulted during GENEND processing, ICNUSGKI gets a return code of 5.
- Keywords that are not valid for the current statement or that are not coded at their lowest statement level will be returned as a null value and ICNUSGKI returns an error code.
- If the keyword is no longer supported for the current statement, ICNUSGKI returns an error code.

Syntax

XVT Location	Parameters
X'AC'	*in_keyword* *out_ptrs* *out_count* *out_status*

Parameters

in_keyword

Function Specifies the name of the keyword for which your routine requires values.

Format Sixteen-character string in string standard representation.

out_ptrs

Function Contains the pointers to the keyword values.

Format An array of up to 100 fullword pointers, each pointing to an area of up to 256 bytes to receive the keyword value itself in string standard value. The number of pointers supplied must be equal to the largest number of subvalues that may be returned or the number supplied on the previous call to ICNUSGKI, whichever is larger. ICNUSGKI will initialize the value areas to null based on the number of values returned on the previous invocation.

Remarks The keyword values themselves are string standard values of up to 256 characters.

out_count

Function Contains the number of subvalues pointed to by *out_ptrs*.

Format Fullword integer.

out_status

Function Returns the status code from ICNUSGKI.

Format Fullword integer.

Remarks *Out_status* is 0 if the subvalues are returned successfully. Nonzero codes are shown below:

Status Code	Meaning
0	All keyword values are returned successfully.
1	The current statement is not an NCP statement. An error message has been issued.
2	The current keyword is not a valid NCP keyword. An error message has been issued.
3	The current keyword is not at its lowest level.
4	ICNUSGKI is invoked during pre-NCP statement keyword processing. An error message has been issued.
5	The current keyword is not coded on the current statement. The keyword value will be defaulted during GENEND processing.

Example

```
CALL ICNUSGKI('VERSION%',OUT_PTRS,OUT_COUNT,OUT_STATUS)
```

If BUILD is the current statement and VERSION=V4R3 is coded on the BUILD definition statement, the first element of *out_ptrs* would point to the string 'V4R3' and all other elements would contain zeros. *Out_count* would be set to 1 and *out_code* would be set to 0.

ICNUSKAD—Add a Keyword to the Generation Definition

The ICNUSKAD utility adds a keyword and its subvalues to the current generation definition statement. If you code the keyword in the generation definition, the subvalues are appended to the existing subvalues. For each subvalue added, NDF prints the message ADDED *keyword(x)=subvalue*. For each subvalue appended, NDF prints the message APPENDED *keyword(x)=subvalue*. *Keyword* is the keyword name, *x* is the subvalue number, and *subvalue* is the subvalue. If the keyword was not coded, the new keyword and subvalues are added.

Syntax

XVT Location	Parameters
X'88'	*in_uwga_name* *in_key_name* *in_subv_num* *in_subv_list* *out_code*

Parameters

in_uwga_name

Function Specifies the name of the user-written-generation-application load module specified on the USERGEN keyword of the OPTIONS definition statement.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks NDF uses this value for diagnostic purposes.

in_key_name

Function Specifies the name of the keyword to be added to the generation definition.

Format Character string in string standard representation. The maximum length is 16 characters.

in_subv_num

Function Specifies the number of subvalues to be added to the generation definition.

Format Fullword integer.

Remarks You can have a maximum of 100 subvalues.

in_subv_list

Function Specifies the subvalues to be added to the generation definition.

Format Array of 256-byte character strings in string standard representation.

Remarks You can have a maximum of 100 subvalues.

out_code

Function Returns the result of the operation.

Format Fullword integer.

Remarks Possible return codes are as follows:

Return Code	Meaning
0	Keyword and subvalues were added successfully.
4	Keyword value length exceeds 255 characters. An error message has been issued, and one or more subvalues have been ignored.
5	NDF storage error. An error message has been issued. This return code is not issued again for the current statement.
10	Maximum number of subvalues has been exceeded. An error message has been issued and one or more subvalues have been ignored.
11	Unusable input passed from the user application. An error message has been issued. The keyword and its subvalues have been ignored.

Examples

```
IN_SUBV_LIST='1%'
IN_SUBV_LIST='2%'

CALL ICNUSKAD('USER1%','REDIAL%',2,IN_SUBV_LIST,OUT_CODE)
```

If REDIAL was not coded on the current definition statement, this call adds the keyword REDIAL with the subvalues 1 and 2 to the current definition statement. The *out_code* parameter is set to 0. This gives the same results as if REDIAL=(1,2) had been coded on the current definition statement.

In the following example, INIT has been coded on the current definition statement as INIT=USER2:

```
IN_SUBV_LIST='USER1%'

CALL ICNUSKAD('USER1%','INIT%',1,IN_SUBV_LIST,OUT_CODE)
```

This call sets *out_code* to 0 and produces the same result as if `INIT=(USER2,USER1)` had been coded on the current definition statement.

ICNUSKRP—Replace Keyword Values in the Generation Definition

The ICNUSKRP utility replaces a keyword's subvalues with new subvalues passed from a user generation application. If the keyword was not coded on the current definition statement, the new keyword and its subvalues are added to the generation definition. If the keyword was coded, the keyword's current subvalues are deleted and the new subvalues are added. For each keyword that is deleted, NDF prints the message `DELETED` *keyword*. For each subvalue added, NDF prints the message `ADDED` *keyword*`(`*x*`)=`*subvalue* where *keyword* is the keyword name, *x* is the subvalue number, and *subvalue* is the subvalue.

Syntax

XVT Location	Parameters
X'8C'	*in_uwga_name* *in_key_name* *in_subv_num* *in_subv_list* *out_code*

Parameters

in_uwga_name

Function Specifies the name of the user-generation-application load module specified on the USERGEN keyword of the OPTIONS definition statement.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks NDF uses this value for diagnostic purposes.

in_key_name

Function Specifies the name of the keyword to be replaced in the generation definition.

Format Character string in string standard representation. The maximum length is 16 characters.

in_subv_num

Function Specifies the number of subvalues to be replaced in the generation definition.

Format Fullword integer.

Remarks You can have a maximum of 100 subvalues. If *in_subv_num* is 0, NDF deletes the keyword from the definition statement.

in_subv_list

Function Specifies the subvalues to be replaced in the generation definition.

Format Array of 256-byte character strings in string standard representation.

Remarks You can have a maximum of 100 subvalues.

out_code

Function Returns the result of the operation.

Format Fullword integer.

Remarks Possible return codes are as follows:

Return Code	Meaning
0	Keyword and subvalues were replaced successfully.
4	Keyword value length exceeds 255 characters. An error message has been issued and one or more subvalues have been ignored.
5	NDF storage error. An error message has been issued. This return code is not issued again for the current statement.
10	The maximum number of subvalues has been exceeded. An error message has been issued and one or more subvalues have been ignored.
11	Unusable input passed from the user application. An error message has been issued. The keyword and its subvalues have been ignored.
12	ICNUSKRP attempted to delete a duplicate keyword. An error message has been issued. An error was found in the generation definition. The keyword and its subvalues have not been replaced.

Example

```
CALL ICNUSKRP('USER1%','LEVEL2%',1,'USELVL2%',OUT_CODE)
```

This call passes the keyword LEVEL2 to NDF with the subvalue USELVL2. It is the same result as if LEVEL2=USELVL2 had been coded on the current definition statement.

ICNUSLIB—Search the SYSLIB Data Set for a Member

The ICNUSLIB utility searches the SYSLIB data set specified in the JCL and verifies the presence of a specific member. It returns a code indicating whether the member was found.

Syntax

XVT Location	Parameters
X'BC'	*status* *mem_name* *out_code*

Parameters

status

Function Specifies the NDF status word.

Format Fullword.

mem_name

Function Specifies the name of the member to be checked for.

Format Character string in string standard representation. The maximum length is as follows:

MVS	8 characters
VSE	7 characters
VM	16 characters

out_code

Function Returns the status of the operation.

Format Fullword integer.

Remarks Zero indicates that the member was found. Four (for MVS and VM) or 8 (for VSE) indicates that the member was not found. Error ICN623I is issued if an input/output error occurs.

Example

```
CALL ICNUSLIB(STATUS,'MEMBER01%',OUT_CODE)
```

If MEMBER01 is found in the SYSLIB data set, *out_code* is set to 0. If it is not found, *out_code* is set to 4 or 8.

ICNUSNEW—Get the NEWNAME Value for the Generation Definition

The ICNUSNEW utility returns the NEWNAME value for the current generation definition. The NEWNAME value is specified either on the BUILD definition statement in the generation definition or in the default generated by NDF. An error results if this routine is called before the BUILD definition statement has been processed by NDF.

Syntax

XVT Location	Parameters
X'90'	*in_uwga_name* *out_newname*

Parameters

in_uwga_name

Function Specifies the name of the user-generation-application load module specified on the USERGEN keyword of the OPTIONS definition statement.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks NDF uses this value for diagnostic purposes.

out_newname

Function Returns the NEWNAME value.

Format Character string in string standard representation. The maximum length is 16 characters.

Example

If the following BUILD definition statement has been coded:

```
BUILD NEWNAME=NCPGEN
```

this call:

```
CALL ICNUSNEW('USER1%',OUT_NEWNAME)
```

sets *out_newname* to 'NCPGEN%'.

ICNUSRNA—Get Network Addresses for a User-Defined Resource

The ICNUSRNA utility gets the network addresses associated with a user-defined resource in the generation definition. You can use this routine for post-processing only.

For all resources except those defined for dynamic reconfiguration, NDF determines the network address after it processes the definition statement on which the resource is defined. The addresses of dynamic reconfiguration resources defined by the BUILD, LUPOOL, LUDRPOOL, and PUDRPOOL definition statements are determined during processing of the NETWORK definition statement. If NETWORK is not coded, the addresses are determined when the GENEND definition statement is processed.

You cannot call this routine with an undefined statement symbol, with a statement symbol that is defined but has no associated network address, or before the network address for a resource has been determined.

Syntax

XVT Location	Parameters
X'B0'	*in_symbol* *out_sa* *out_na* *out_na_count* *out_status*

Parameters

in_symbol

Function Specifies the statement symbol associated with the network addresses requested.

Format Sixteen-character string in string standard representation.

out_sa

Function Returns the subarea associated with the statement symbol.

Format Fullword integer.

out_na

Function Returns the first network address associated with the statement symbol.

Format Fullword integer.

out_na_count

Function Returns the number of consecutive network addresses associated with the statement symbol.

Format Fullword integer.

out_status

Function Returns the status code from ICNUSRNA.

Format Fullword integer.

Remarks *Out_status* is 0 if a network address is returned. Nonzero codes are as follows:

Status Code	Meaning
0	Network address returned successfully.
1	Statement symbol is undefined. An error message has been issued.
2	No network address is associated with the statement symbol. An error message has been issued.
3	Network address associated with the statement symbol has not yet been determined. An error message has been issued.

Example

If the resource with the name SYMBOL1 has a subarea of 1 and the network addresses 25 and 26, then the call:

```
CALL ICNUSRNA('SYMBOL1%',OUT_SA,OUT_NA,OUT_NA_COUNT,OUT_STATUS)
```

sets *out_sa* to 1, *out_na* to 25, *out_na_count* to 2, and *out_status* to 0.

ICNUSSTA—Activate a Group of Statements

The ICNUSSTA utility activates a statement group. A statement group consists of one or more statements that have been passed to NDF by ICNUSSTS with the same data key. NDF uses the statement group that matches the data key passed to ICNUSSTA as input. Statements within a statement group are processed in the same order they were generated. Before each statement in a statement group is processed, NDF prints `GENERATED BY` *xxxxxxxx* in the NDF listing, where *xxxxxxxx* is the name of the user-written generation application that created the statement.

If ICNUSSTA is called to activate a statement group and another statement group is currently being processed, the statement group is queued for processing. If multiple statement groups are queued, they are processed in the order they were created.

Syntax

XVT Location	Parameters
X'94'	*in_sg_datakey* *out_code*

Parameters

in_sg_datakey

Function Specifies the data key of the statement group to be activated.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The first 3 characters of the data key must match the first 3 letters of one of the load module names specified on the USERGEN keyword on the OPTIONS definition statement.

out_code

Function Returns the status of the routine call.

Format Fullword integer.

Remarks The status codes returned are as follows:

Status Code	Meaning
0	The statement group has been activated or queued for activation.
1	The statement group does not exist. An error message has been issued.
3	The statement group has already been processed. An error message has been issued.

Example

```
CALL ICNUSSTA('UWGAKEY1%',OUT_CODE)
```

This call activates the statement group with the data key `'UWGAKEY1%'`. The *out_code* keyword is set to 0.

ICNUSSTC—Add a Comment to a User-Generated Statement Group

The ICNUSSTC utility builds a comment card from the contents of a global variable specified on the call and adds it to a user-generated statement set.

Syntax

XVT Location	Parameters
X'C4'	*in_datakey* *in_val* *out_code*

Parameters

in_datakey

Function Specifies the name of the generated statement group. This group will not create or begin a new group.

Format Character string in string standard representation. The maximum length is 9 characters.

in_val

Function Specifies the name of the global variable containing the comment string. ICNUSSTC precedes the string with an asterisk and a space.

Format Character string in string standard representation.

Remarks The length of the comment string should not exceed 70 characters.

out_code

Function Specifies the return code from the call.

Format Fullword integer.

Example

```
CALL ICNUSSTC('STMTGRP%',KYVAL,OUT_CODE)
```

This call will insert a comment into the generated statement group STMTGRP. If the variable KYVAL contains the comment:

```
This is a certain type of line%
```

then the comment would appear as:

```
* This is a certain type of line
```

ICNUSSTI—Set an Insertion Point for a Statement Group

The ICNUSSTI utility allows you to determine the location where the next statement generated by ICNUSSTS will be inserted into the statement group. Normally, ICNUSSTS appends statements to the end of a statement group. With ICNUSSTI, you can have the statement added at the top of the group or inserted before or after any label in the group.

You supply the name of the statement group and the label where the insertion is to take place, and specify whether the insertion is to be before or after the label. The position indicator for before is `'B%'` and for after is `'A%'`. If you want the statement to be added at the top of the group, specify `'%'`.

Note that ICNUSSTI sets the position only for the **next** statement generated by ICNUSSTS. Subsequent calls to ICNUSSTS will append statements to the end of the group unless ICNUSSTI is again used to set the insertion point.

Syntax

XVT Location	Parameters
`X'B8'`	*in_sg_datakey* *in_stmt_symbol* *in_position* *out_code*

Parameters

in_sg_datakey

Function Specifies the statement group to which a statement is to be added.

Format Sixteen-character string in string standard representation.

Remarks This value must have been set previously by your generation application.

in_stmt_symbol

Function Specifies the label of the statement symbol where the next statement is to be inserted.

Format Sixteen-character string in string standard representation.

in_position

Function Specifies where the next statement is to inserted.

Format Sixteen-character string in string standard representation.

Remarks Position indicators are as follows:

'B%' Next statement is inserted before the label.

'A%' Next statement is inserted after the label.

'%' Next statement is inserted at the top of the group, before any existing statements.

out_code

Function Specifies the return code.

Format Fullword integer.

Remarks Status codes are as follows:

Status Code	Meaning
0	The statement group has been activated or queued for activation.
1	The statement group does not exist. An error message has been issued.
3	The statement group has already been processed. An error message has been issued.

Examples

Starting with this statement group:

```
*         Statement group 1
LABEL1    STMNT  KEY1=VALUE
LABEL2    STMNT  KEY1=VALUE
```

you can insert a statement between LABEL1 and LABEL2, like so:

```
CALL ICNUSSTI(STMT_GROUP1,LABEL1,'A%',OUT_CODE)
```

or

```
CALL ICNUSSTI(STMT_GROUP1,LABEL2,'B%',OUT_CODE)
```

To insert a statement at the top of the statement group, you can do this:

```
CALL ICNUSSTI(STMT_GROUP1,LABEL2,'%',OUT_CODE)
```

Note that LABEL2 is ignored in this case.

ICNUSSTS—Save a Definition Statement

The ICNUSSTS utility allows your generation application to pass statements and their associated keywords to NDF for future processing. ICNUSSTS builds statements and their keywords for calling the user-written generation application. Multiple statements can be grouped together by giving them identical data keys. NDF does not immediately process statement groups created by ICNUSSTS. A statement group must first be activated using ICNUSSTA before NDF treats it as input. If bit 1 of *in_flag* was set when ICNUSSTS created the statement group, then when ICNUSSTA activates the statement group, it will be added to the NEWDEFN file, but will not be processed by NDF.

A statement is composed of a statement symbol, statement name, and keywords. The statement symbol is optional; the statement name is required. There may be any number of keywords. Each keyword is composed of a keyword name and a keyword value.

Every call to ICNUSSTS must have the data key as a parameter. You must specify the statement name on the first call to ICNUSSTS for a new statement. If there is to be a statement symbol, it must be a parameter on the first call. For each successive call to ICNUSSTS for the same statement, the statement name and statement symbol parameters must be null strings. On the first call to ICNUSSTS, the keyword name and keyword value strings can be null only if there are no keywords for the statement. In any other call, you must specify the keyword name and keyword value strings. Bit 0 of *in_flag* is the last-call indicator. For the last call to ICNUSSTS for a statement, you must set bit 0 of *in_flag* to 1. For all other calls to ICNUSSTS, you must set bit 0 of *in_flag* to 0.

Syntax

XVT Location	Parameters
X'98'	*in_sg_datakey* *in_stmt_symbol* *in_stmt_name* *in_key_name* *in_key_value* *in_flag* *out_code*

Parameters

in_sg_datakey

Function Specifies the data key of the statement group to be activated.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The first 3 characters of the data key must match the first 3 letters of one of the load module names specified on the USERGEN keyword of the OPTIONS definition statement.

in_stmt_symbol

Function Specifies the statement symbol.

Format Character string in string standard representation. The maximum length is 9 characters.

Remarks This parameter must be '%' for every call to ICNUSSTS after the first call.

in_stmt_name

Function Specifies the name of the statement being saved.

Format Character string in string standard representation. The maximum length is 13 characters.

Remarks This parameter must be coded with the statement name on the first call to ICNUSSTS. On successive calls, the parameter must be '%'.

in_key_name

Function Specifies the name of the keyword.

Format Character string in string standard representation. The maximum length is 13 characters.

Remarks The keyword name must be passed on every call to ICNUSSTS unless the statement has no keywords. If there are no keywords, the keyword name must be '%'.

in_key_value

Function Specifies the value of the keyword.

Format Character string in string standard representation, with a maximum length of 256 characters.

Remarks The keyword value must be passed on every call to ICNUSSTS unless the definition statement has no keywords. If there are no keywords, the keyword value must be '%'.

in_flag

Function Specifies the flag bits used by the ICNUSSTS routine.

Format One-byte flag. Bit 0 through bit 2 are checked.

Remarks If bit 0 is set to 1, the call is the last call for a statement. If bit 1 is set to 1, the statement, when activated, will be passed to the NEWDEFN file but will not be processed by NDF. If bit 2 is set, the statement, when passed to the NEWDEFN file, will not be surrounded by IGNORE and NOIGNORE statements.

out_code

Function Returns the status of the call.

Format Fullword integer.

Remarks The codes returned in *out_code* are as follows:

Status Code	Meaning
0	Data successfully added to statement group.
2	Storage for user application statements has been exceeded. An error message has been issued. Do not call this routine again.
4	Maximum number of statement groups exceeded. An error message has been issued. Do not call this routine again.
5	Incorrect sequence of calls to ICNUSSTS. An error message has been issued. The current statement being created is ended.
6	Statement group is either already processed, is being processed, or is queued for processing. The data is not added to the statement group. An error message has been issued.

Examples

```
CALL ICNUSSTS('UWGAKEY1%','LINE1%','LINE%','UACB%','X$1%','0'B,OUT_CODE)

CALL ICNUSSTS('UWGAKEY1%','%','%','CALL%','REDIAL=(1,2)%','1'B,OUT_CODE)

CALL ICNUSSTS('UWGAKEY1%','PU1%','PU%','%','%','1'B,OUT_CODE)
```

These calls create the following statement group with data key `'UWGAKEY1'` in NDF:

```
LINE1    LINE  UACB=X$1,CALL=REDIAL
PU1      PU
```

If you want this statement group to only be passed to the NEWDEFN file and not be processed by NDF, code the following:

```
CALL ICNUSSTS('UWGAKEY1%','LINE1%','LINE%','UACB%','X$1%','01'B,OUT_CODE)

CALL ICNUSSTS('UWGAKEY1%','%','%','CALL%','REDIAL=(1,2)%','11'B,OUT_CODE)

CALL ICNUSSTS('UWGAKEY1%','PU1%','PU%','%','%','11'B,OUT_CODE)
```

ICNUSTAD—Add an Entry to the Table Storage Facility

The ICNUSTAD utility creates an entry in the NDF table storage facility. You supply a data key with the data. If the data key is matched with an entry already in the table, the routine returns an error code of 1. If the data key is not in the table, ICNUSTAD creates the entry and stores the data with its key.

You can use the routine ICNUSTRT to retrieve the data in an entry and the routine ICNUSTUP to update the data in an entry.

Syntax

XVT Location	Parameters
X'9C'	*in_data_key* *in_uwga_data* *out_code*

Parameters

in_data_key

Function Specifies the data key used by NDF to identify an entry in the table storage facility.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The first 3 characters must match the first 3 letters of one of the load module names specified on the USERGEN keyword on the OPTIONS definition statement.

in_uwga_data

Function Specifies the data to be stored in the table storage facility entry.

Format 20 bytes of data of any type.

out_code

Function Returns the status of the operation.

Format Fullword integer.

Remarks The status codes returned are as follows:

Status Code	Meaning
0	NDF table storage facility entry successfully created.
1	NDF table storage facility entry already exists. No error message is issued.
2	Table facility error. An error message is issued and the data is not stored. Do not call the routine again.
11	*in_data_key* is not valid. An error message is issued and the data is not stored.

Example

```
CALL ICNUSTAD('USERKEY%','TWENTY BYTES OF DATA',OUT_CODE)
```

This call sets *out_code* to 0, creates a new entry in the table storage facility with the data key `'USERKEY%'`, and stores the 20 bytes in the NDF table storage facility.

ICNUSTRT—Get Data from the Table Storage Facility

The ICNUSTRT utility retrieves the data from an entry in the NDF table storage facility. If the data key is not matched, the routine returns an error code of 3. If the data key is matched with an entry already in the table, ICNUSTRT updates the data field of the matching table entry with the new data.

Syntax

XVT Location	Parameters
X'A0'	*in_data_key* *out_uwga_data* *out_code*

Parameters

in_data_key

Function Specifies the data key used by NDF to identify an entry in the table storage facility.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The first 3 characters must match the first 3 letters of one of the load module names specified on the USERGEN keyword on the OPTIONS definition statement.

out_uwga_data

Function Specifies the data to be retrieved from the table storage facility.

Format 20 bytes of data of any type.

out_code

Function Returns the status of the operation.

Format Fullword integer.

Remarks The status codes returned are as follows:

Status Code	Meaning
0	NDF table storage facility data successfully retrieved.
2	Table facility error. An error message is issued and the data is not retrieved. Do not call the routine again.
3	NDF table storage facility entry not found. No error message is issued and the data is not retrieved.
11	Incorrect input. An error message is issued and the data is not retrieved.

Example

The data `'TWENTY BYTES OF DATA'` has already been saved under the key `'USERKEY'` (see the example for ICNUSTAD).

```
CALL ICNUSTRT('USERKEY%',OUT_UWGA_DATA,OUT_CODE)
```

This call sets *out_uwga_data* to `'TWENTY BYTES OF DATA'` and sets *out_code* to 0.

ICNUSTUP—Update the Table Storage Facility

The ICNUSTUP utility updates the data field of an entry in the NDF table storage facility. If the data key is not matched with an entry already in the table, the routine returns an error code of 3. If the data key is matched, the data field of the matching table entry is updated with the new data.

Syntax

XVT Location	Parameters
X'A4'	*in_data_key* *in_uwga_data* *out_code*

Parameters

in_data_key

Function Specifies the data key used by NDF to identify an entry in the table storage facility.

Format Character string in string standard representation. The maximum length is 16 characters.

Remarks The first 3 characters must match the first 3 letters of the load module name specified on the USERGEN keyword on the OPTIONS definition statement.

in_uwga_data

Function Specifies the data to be stored in the table storage facility entry.

Format 20 bytes of data of any type.

out_code

Function Returns the status of the operation.

Format Fullword integer.

Remarks The status codes returned are as follows:

Status Code	Meaning
0	NDF table storage facility entry successfully updated.
2	Table facility error. An error message is issued and the data is not stored. Do not call the routine again.
3	*in_data_key* not matched with an existing entry. No error message is issued and the table is not updated.
11	Incorrect input. An error message is issued and the table is not updated.

Example

```
CALL ICNUSTUP('USERKEY%','MORE DATA FOR TABLE ',OUT_CODE)
```

This call changes the data stored under the key `'USERKEY'` from `'TWENTY BYTES OF DATA'` to `'MORE DATA FOR TABLE'` and sets *out_code* to 0.

Glossary, Bibliography, and Index

Glossary

This glossary includes terms and definitions from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The ANSI/EIA Standard—440-A, *Fiber Optic Terminology*. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The Network Working Group Request for Comments: 1208.

The following cross-references are used in this glossary:

Contrast with: This refers to a term that has an opposed or substantively different meaning.

Synonym for: This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the glossary.

Synonymous with: This is a backward reference from a defined term to all other terms that have the same meaning.

See: This refers the reader to multiple-word terms that have the same last word.

See also: This refers the reader to terms that have a related, but not synonymous, meaning.

Deprecated term for: This indicates that the term should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

A

abend. (1) Abnormal end of task. (2) Synonym for *abnormal termination.*

abnormal end. Synonym for *abnormal termination.*

abnormal end of task (abend). Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

abnormal termination. (1) The cessation of processing prior to planned termination. (T) (2) A system failure or operator action that causes a job to end unsuccessfully. (3) Synonymous with *abend* and *abnormal end.*

ACB. In NCP, adapter control block.

ACF. Advanced Communications Function.

ACF/TAP. Advanced Communications Function/Trace Analysis Program. Synonymous with *TAP.*

ACF/TCAM. Advanced Communications Function for the Telecommunications Access Method. Synonym for *TCAM.*

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method. Synonym for *VTAM.*

ACTLINK. Activate link.

ACTPU. Activate physical unit. In SNA, a command used to start a session on a physical unit.

adapter control block (ACB). In NCP, a control block that contains line control information and the states of I/O operations for BSC lines, SS lines, or SDLC links.

address. In data communication, the unique code assigned to each device or workstation connected to a network.

addressing. In data communication, the way in which a station selects the station to which it is to send data.

Advanced Communications Function (ACF). A group of IBM licensed programs, principally VTAM, TCAM, NCP, and SSP, that use the concepts of Systems Network Architecture (SNA), including distribution of function and resource sharing.

Advanced Communications Function/Trace Analysis Program (ACF/TAP). An SSP program service aid that assists in analyzing trace data produced by VTAM, TCAM, and NCP and provides network data traffic and network error reports. Synonymous with *Trace Analysis Program (TAP).*

alias address. An address used by a gateway NCP and a gateway system services control point (SSCP) in one network to represent a logical unit (LU) or SSCP in another network.

AND operation. Synonym for *conjunction.*

B

basic transmission unit (BTU). In SNA, the unit of data and control information passed between path control components. A BTU can consist of one or more path information units (PIUs). See also *blocking of PIUs.*

begin bracket. In SNA, the value (binary 1) of the begin-bracket indicator in the request header (RH) of the first request in the first chain of a bracket; the value denotes the start of a bracket. Contrast with *end bracket.* See also *bracket.*

BER. (1) Box event record. (2) Box error record.

binary synchronous communication (BSC). A form of telecommunication line control that uses a standard set of transmission control characters and control character sequences, for binary synchronous transmission of binary-coded data between stations. Contrast with *Synchronous Data Link Control (SDLC).*

binary synchronous transmission. Data transmission in which synchronization of characters is controlled by timing signals generated at the sending and receiving stations. See also *start-stop transmission* and *Synchronous Data Link Control (SDLC).*

BIND. In SNA, a request to activate a session between two logical units (LUs). See also *session activation request.* Contrast with *UNBIND.*

blocking of PIUs. In SNA, an optional function of path control that combines multiple path information units (PIUs) in a single basic transmission unit (BTU).

Note: When blocking is not done, a BTU consists of one PIU.

boundary function. (1) In SNA, a capability of a subarea node to provide protocol support for attached peripheral nodes, such as: (a) interconnecting subarea path control and peripheral path control elements, (b) performing session sequence numbering for low-function peripheral nodes, and (c) providing session-level pacing support. (2) In SNA, the component that provides these capabilities.

bracket. In SNA, one or more chains of request units and their responses that are exchanged between two session partners and that represent a transaction between them. A bracket must be completed before another bracket can be started. Examples of brackets are database inquiries/replies, update transactions, and remote job entry output sequences to workstations.

bracket protocol. In SNA, a data flow control protocol in which exchanges between two session partners are achieved through the use of brackets, with one partner designated at session activation as the first speaker and the other as the bidder. The bracket protocol involves bracket initiation and termination rules.

BSC. Binary synchronous communication.

BTU. Basic transmission unit.

bus. (1) A facility for transferring data between several devices located between two end points, only one device being able to transmit at a given moment. (T) (2) A computer configuration in which processors are interconnected in series.

C

CA. Channel adapter.

CCU. Central control unit.

channel. (1) A path along which signals can be sent, for example, data channel, output channel. (A) (2) A functional unit, controlled by the processor, that handles the transfer of data between processor storage and local peripheral equipment. See *input/output channel.*

channel adapter. A communication controller hardware unit that is used to attach the communication controller to a host channel.

channel-attached. (1) Pertaining to the attachment of devices directly by input/output channels to a host processor. (2) Pertaining to devices attached to a controlling unit by cables, rather than by telecommunication lines. Contrast with *link-attached.* Synonymous with *local.*

circuit. (1) One or more conductors through which an electric current can flow. See *physical circuit* and *virtual circuit.* (2) A logic device.

circuit switching. (1) A process that, on demand, connects two or more data terminal equipment (DTEs) and permits the exclusive use of a data circuit between

them until the connection is released. (I) (A) (2) Synonymous with *line switching*. (3) See also *message switching* and *packet switching*.

communication controller. A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit. It manages the details of line control and the routing of data through a network.

communication scanner processor (CSP). A processor in the 3725 Communication Controller that contains a microprocessor with control code. The code controls transmission of data over links attached to the CSP.

configuration report program (CRP). An SSP utility program that creates a configuration report listing network resources and resource attributes for networks with NCP, EP, PEP, or VTAM.

contention. In a session, a situation in which both NAUs attempt to initiate the same action at the same time, such as when both attempt to send data in a half-duplex protocol (half-duplex contention), or both attempt to start a bracket (bracket contention). At session initiation, one NAU is defined to be the contention winner; its action will take precedence when contention occurs. The contention loser must get explicit or implicit permission from the contention winner to begin its action.

control block. (1) A storage area used by a computer program to hold control information. (I) (2) In the IBM Token-Ring Network, a specifically formatted block of information provided from the application program to the Adapter Support Interface to request an operation.

control point (CP). (1) A component of an APPN or LEN node that manages the resources of that node. In an APPN node, the CP is capable of engaging in CP-CP sessions with other APPN nodes. In an APPN network node, the CP also provides services to adjacent end nodes in the APPN network. (2) A component of a node that manages resources of that node and optionally provides services to other nodes in the network. Examples are a system services control point (SSCP) in a type 5 subarea node, a network node control point (NNCP) in an APPN network node, and an end node control point (ENCP) in an APPN or LEN end node. An SSCP and an NNCP can provide services to other nodes.

control program. (1) A computer program designed to schedule and to supervise the execution of programs of a computer system. (I) (A) (2) The part of the AIX Base Operating System that determines the order in which basic functions should be performed. (3) See *VM/370 control program (CP)*.

control statement. In the NetView program, a statement in a command list that controls the processing sequence of the command list or allows the command list to send messages to the operator and receive input from the operator.

CP. (1) VM/370 control program. (2) Control point.

CRP. Configuration report program.

CSP. Communication scanner processor.

CT. Control terminal.

CWALL. An NCP threshold of buffer availability, below which the NCP will accept only high-priority path information units (PIUs).

D

DAF. Destination address field.

DAF'. Destination address field prime.

data channel. Synonym for *input/output channel*.

data flow control (DFC). In SNA, a request/response unit (RU) category used for requests and responses exchanged between the data flow control layer in one half-session and the data flow control layer in the session partner.

data flow control (DFC) layer. In SNA, the layer within a half-session that controls whether the half-session can send, receive, or concurrently send and receive, request units (RUs); groups related RUs into RU chains; delimits transactions via the bracket protocol; controls the interlocking of requests and responses in accordance with control modes specified at session activation; generates sequence numbers; and correlates requests and responses.

data flow control (DFC) protocol. In SNA, the sequencing rules for requests and responses by which network addressable units (NAUs) in a session coordinate and control data transfer and other operations; for example, bracket protocol.

data link. In SNA, synonym for *link*.

data link control (DLC). A set of rules used by nodes on a data link (such as an SDLC link or a token ring) to accomplish an orderly exchange of information.

data link control (DLC) layer. In SNA, the layer that consists of the link stations that schedule data transfer over a link between two nodes and perform error control for the link. Examples of data link control are SDLC for

serial-by-bit link connection and data link control for the System/370 channel.

Note: The DLC layer is usually independent of the physical transport mechanism and ensures the integrity of data that reaches the higher layers.

data link level. (1) In the hierarchical structure of a data station, the conceptual level of control or processing logic between high level logic and the data link that maintains control of the data link. The data link level performs such functions as inserting transmit bits and deleting receive bits; interpreting address and control fields; generating, transmitting, and interpreting commands and responses; and computing and interpreting frame check sequences. See also *higher level*, *packet level*, and *physical level*. (2) In X.25 communications, synonym for *frame level*.

definite response (DR). In SNA, a protocol requested in the form-of-response-requested field of the request header that directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request chain. Contrast with *exception response* and *no response*.

definition statement. In NCP, a type of instruction that defines a resource to the NCP. See Figure 1. See also *macroinstruction*.

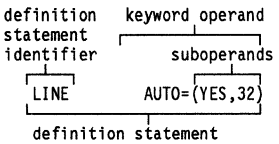


Figure 1. Example of an NCP Definition Statement

destination address. A code that identifies the location to which information is to be sent.

destination address field (DAF). In SNA, a field in a FID0 or FID1 transmission header that contains the network address of the destination.

destination subarea field (DSAF). In SNA, a field in a FID4 transmission header that contains a subarea address, which combined with the element address in the destination element field, gives the complete network address of the destination network addressable unit (NAU). Contrast with *origin subarea field*.

DFC. Data flow control.

disabled. (1) Pertaining to a state of a processing unit that prevents the occurrence of certain types of interruptions. (2) Pertaining to the state in which a transmission control unit or audio response unit cannot accept incoming calls on a line. See also *enabled*.

DLC. Data link control.

domain operator. In a multiple-domain network, the person or program that controls operation of resources controlled by one system services control point (SSCP). See also *network operator*.

DSAF. Destination subarea field.

DSECT. Dummy control section.

duplex. Pertaining to communication in which data can be sent and received at the same time. Synonymous with *full duplex*. Contrast with *half duplex*.

dynamic reconfiguration (DR). The process of changing the network configuration (peripheral PUs and LUs) without regenerating complete configuration tables or deactivating the affected major node.

E

ECB. Event control block.

echo. (1) In computer graphics, the immediate notification of the current values provided by an input device to the operator at the display console. (I) (A) (2) In word processing, to print or display each character or line as it is keyed in. (3) In data communication, a reflected signal on a communications channel. On a communications terminal, each signal is displayed twice, once when entered at the local terminal and again when returned over the communications link. This allows the signals to be checked for accuracy.

element. (1) A field in the network address. (2) In SNA, the particular resource within a subarea that is identified by an element address. See also *subarea*.

element address. In SNA, a value in the element address field of the network address identifying a specific resource within a subarea. See *subarea address*.

Emulation Program (EP). An IBM control program that allows a channel-attached 3705 or 3725 communication controller to emulate the functions of an IBM 2701 Data Adapter Unit, an IBM 2702 Transmission Control, or an IBM 2703 Transmission Control. See also *network control program*.

enabled. (1) Pertaining to a state of the processing unit that allows the occurrence of certain types of interruptions. (2) Pertaining to the state in which a transmission control unit or an audio response unit can accept incoming calls on a line. See also *disabled*.

end bracket. In SNA, the value (binary 1) of the end bracket indicator in the request header (RH) of the first request of the last chain of a bracket; the value denotes

the end of the bracket. Contrast with *begin bracket*. See also *bracket*.

entry point (EP). In SNA, a type 2.0, type 2.1, type 4, or type 5 node that provides distributed network management support. It sends network management data about itself and the resources it controls to a focal point for centralized processing, and it receives and executes focal-point initiated commands to manage and control its resources.

EP. Emulation Program.

ER. (1) Explicit route. (2) Exception response.

ERP. Error recovery procedures.

error recovery procedures (ERP). (1) Procedures designed to help isolate and, where possible, to recover from errors in equipment. The procedures are often used in conjunction with programs that record information on machine malfunctions. (2) A set of routines that attempt to recover from transmission errors.

ESC. Execution sequence control.

event control block (ECB). A control block used to represent the status of an event.

exception request (EXR). In SNA, a request that replaces another message unit in which an error has been detected and that carries sense data identifying the error.

exception response (ER). In SNA, a protocol requested in the form-of-response-requested field of a request header that directs the receiver to return a response only if the request is unacceptable as received or cannot be processed; that is, a negative response, but not a positive response, can be returned. Contrast with *definite response* and *no response*.

exchange identification (XID). A specific type of basic link unit that is used to convey node and link characteristics between adjacent nodes. XIDs are exchanged between link stations before and during link activation to establish and negotiate link and node characteristics, and after link activation to communicate changes in these characteristics.

explicit route (ER). In SNA, a series of one or more transmission groups that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number. Contrast with *virtual route (VR)*.

EXR. Exception request.

extended architecture (XA). An extension to System/370 architecture that takes advantage of continuing high performance enhancements to computer system hardware.

extended network addressing. The network addressing system that splits the address into an 8-bit subarea and a 15-bit element portion. The subarea portion of the address is used to address host processors or communication controllers. The element portion is used to permit processors or controllers to address resources.

F

FDX. Full duplex.

FID. Format identification.

FIFO. First-in-first-out. (A)

flow control. In SNA, the process of managing the rate at which data traffic passes between components of the network. The purpose of flow control is to optimize the rate of flow of message units with minimum congestion in the network; that is, to neither overflow the buffers at the receiver or at intermediate routing nodes, nor leave the receiver waiting for more message units. See also *adaptive session-level pacing*, *pacing*, and *session-level pacing*.

FMD. Function management data.

format identification (FID) field. In SNA, a field in each transmission header (TH) that indicates the format of the TH; that is, the presence or absence of certain fields. TH formats differ in accordance with the types of nodes between which they pass. Following are the six FID types:

FID0, used for traffic involving non-SNA devices between adjacent subarea nodes when either or both nodes do not support explicit route and virtual route protocols

FID1, used for traffic involving SNA devices between adjacent subarea nodes when either or both nodes do not support explicit route and virtual route protocols

FID2, used for traffic between a subarea node and an adjacent type 2 peripheral node

FID3, used for traffic between a subarea node and an adjacent type 1 peripheral node

FID4, used for traffic between adjacent subarea nodes when both nodes support explicit route and virtual route protocols

FIDF, used for certain commands (for example, for transmission group control) sent between adjacent

subarea nodes when both nodes support explicit route and virtual route protocols.

frame. (1) In Open Systems Interconnection architecture, a data structure pertaining to a particular area of knowledge and consisting of slots that can accept the values of specific attributes and from which inferences can be drawn by appropriate procedural attachments. (T) (2) The unit of transmission in some local area networks, including the IBM Token-Ring Network. It includes delimiters, control characters, information, and checking characters. (3) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures.

frame level. See *link level.*

full duplex (FDX). Synonym for *duplex.*

G

gateway. (1) A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures. (T) (2) In the AIX operating system, an entity that operates above the link layer and translates, when required, the interface and protocol used by one network into those used by another distinct network. (3) In TCP/IP, a device used to connect two systems that use either the same or different communications protocols. (4) The combination of machines and programs that provide address translation, name translation, and system services control point (SSCP) rerouting between independent SNA networks to allow those networks to communicate. A gateway consists of one gateway NCP and at least one gateway VTAM. (5) In the IBM Token-Ring Network, a device and its associated software that connect a local area network to another local area network or a host that uses different logical link protocols.

gateway NCP. An NCP that performs address translation to allow cross-network session traffic. The gateway NCP connects two or more independent SNA networks. Synonymous with *gateway node.*

gateway node. Synonym for *gateway NCP.*

generation definition. The definition statement of a resource used in generating a program.

generic alert. A product-independent method of encoding alert data by means of both (a) code points indexing short units of stored text and (b) textual data.

generic unbind. Synonym for *session deactivation request.*

H

half-duplex (HD, HDX). In data communication, pertaining to transmission in only one direction at a time. Contrast with *duplex.* See also *half-duplex operation* and *half-duplex transmission.*

host processor. (1) A processor that controls all or part of a user application network. (T) (2) In a network, the processing unit in which the data communication access method resides.

I

I frame. Information frame.

I/O. Input/output.

ICW. Interface control word.

INITIATE. A network services request sent from a logical unit (LU) to a system services control point (SSCP) requesting that an LU-LU session be established.

input/output channel. (1) In a data processing system, a functional unit that handles transfer of data between internal and peripheral equipment. (I) (A) (2) In a computing system, a functional unit, controlled by a processor, that handles transfer of data between processor storage and local peripheral devices. Synonymous with *data channel.* See *channel.* See also *link.*

interface. (1) A shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics, as appropriate. The concept includes the specification of the connection of two devices having different functions. (T) (2) Hardware, software, or both, that links systems, programs, or devices.

J

JCL. Job control language.

job control language (JCL). A control language used to identify a job to an operating system and to describe the job's requirements.

K

keyword. (1) In programming languages, a lexical unit that, in certain contexts, characterizes some language construct; for example, in some contexts, IF characterizes an if-statement. A keyword normally has the form of an identifier. (I) (2) One of the predefined words of

an artificial language. (A) (3) A significant and informative word in a title or document that describes the content of that document. (4) A name or symbol that identifies a parameter. (5) The part of a command operand that consists of a specific character string (such as DSNAME=). See also *definition statement* and *keyword operand*. Contrast with *positional operand*.

keyword operand. An operand that consists of a keyword followed by one or more values (such as DSNAME=HELLO). See also *definition statement*. Contrast with *positional operand*.

keyword parameter. A parameter that consists of a keyword followed by one or more values.

L

LCB. Local block common.

LEN. Low-entry networking.

LIC. (1) Last-in-chain. (2) In NCP, line interface coupler.

line. (1) The portion of a data circuit external to data circuit-terminating equipment (DCE), that connects the DCE to a data switching exchange (DSE), that connects a DCE to one or more other DCEs, or that connects a DSE to another DSE. (I) (2) Synonymous with *channel* and *circuit*.

line group. One or more telecommunication lines of the same type that can be activated and deactivated as a unit.

line switching. Synonym for *circuit switching*.

link. (1) The combination of the link connection (the transmission medium) and two link stations, one at each end of the link connection. A link connection can be shared among multiple links in a multipoint or token-ring configuration. (2) To interconnect items of data or portions of one or more computer programs: for example, the linking of object programs by a linkage editor, linking of data items by pointers. (T)

link-attached. Pertaining to devices that are connected to a controlling unit by a data link. Contrast with *channel-attached*. Synonymous with *remote*.

link level. A part of Recommendation X.25 that defines the link protocol used to get data into and out of the network across the full-duplex link connecting the subscriber's machine to the network node. LAP and LAPB are the link access protocols recommended by the CCITT. See *data link level*.

Link Problem Determination Aid (LPDA). A series of procedures that are used to test the status of and to control DCEs, the communication line, and the remote device interface. These procedures, or a subset of them, are implemented by host programs (such as the NetView program and VTAM), communication controller programs (such as NCP), and IBM LPDA DCEs. See also *LPDA-1* and *LPDA-2*.

local. Pertaining to a device accessed directly without use of a telecommunication line. Synonym for *channel-attached*.

logical unit (LU). A type of network accessible unit that enables end users to gain access to network resources and communicate with each other.

logical unit (LU) 6.2. A type of logical unit that supports general communication between programs in a distributed processing environment. LU 6.2 is characterized by (a) a peer relationship between session partners, (b) efficient utilization of a session for multiple transactions, (c) comprehensive end-to-end error processing, and (d) a generic application program interface (API) consisting of structured verbs that are mapped into a product implementation.

low-entry networking (LEN). A capability of nodes to attach directly to one another using basic peer-to-peer protocols to support multiple and parallel sessions between logical units.

low-entry networking (LEN) end node. A LEN node receiving network services from an adjacent APPN network node.

low-entry networking (LEN) node. A node that provides a range of end-user services, attaches directly to other nodes using peer protocols, and derives network services implicitly from an adjacent APPN network node, that is, without the direct use of CP-CP sessions.

LPDA. Link Problem Determination Aid.

LPDA-1. The first version of the LPDA command set. LPDA-1 is not compatible with LPDA-2. See also *Link Problem Determination Aid (LPDA)* and *LPDA-2*.

LPDA-2. The second version of the LPDA command set. LPDA-2 provides all of the functions of LPDA-1; it also supports commands such as the following:

- DCE configuration
- Dial
- Set transmit speed
- Commands to operate a contact that can control external devices.

See also *Link Problem Determination Aid (LPDA)* and *LPDA-1*.

LU. Logical unit.

LU-LU session. A logical connection between two logical units (LUs) in an SNA network that typically provides communication between two end users.

M

macroinstruction. (1) An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that may also specify values for parameters in the replaced instructions. (T) (2) In assembler programming, an assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macroinstruction in the program. See also *definition statement*.

maintenance and operator subsystem (MOSS). A subsystem of an IBM communication controller, such as the 3725 or the 3720, that contains a processor and operates independently of the rest of the controller. It loads and supervises the controller, runs problem determination procedures, and assists in maintaining both hardware and software.

mixed-media multilink transmission group (MMMLTG). See *transmission group (TG)*.

MLTG. Multilink transmission group.

MMMLTG. Mixed-media multilink transmission group.

MOSS. Maintenance and operator subsystem.

MSG. Console messages.

multilink transmission group (MLTG). See *transmission group (TG)*.

Multiple Virtual Storage (MVS). See *MVS*.

MVS. Multiple Virtual Storage. Implies MVS/370, the MVS/XA product, and the MVS/ESA product.

MVS/ESA product. Multiple Virtual Storage/Enterprise Systems Architecture.

MVS/XA product. Multiple Virtual Storage/Extended Architecture product, consisting of MVS/System Product Version 2 and the MVS/XA Data Facility Product, operating on a System/370 processor in the System/370 extended architecture mode. MVS/XA allows virtual storage addressing to 2 gigabytes. See also *MVS*.

N

native network. The subnetwork whose network identifier a node uses for its own network-qualified resource names.

NCP. Network Control Program.

NCP/EP definition facility (NDF). A program that is part of System Support Programs (SSP) and that is used to generate a load module for a partitioned emulation program (PEP), a Network Control Program (NCP), or an Emulation Program (EP).

NDF. NCP/EP definition facility.

NetView Performance Monitor (NPM). An IBM licensed program that collects, monitors, analyzes, and displays data relevant to the performance of a VTAM telecommunication network. It runs as an online VTAM application program.

network address. (1) In a subarea network, an address, consisting of subarea and element fields, that identifies a link, link station, physical unit, logical unit, or system services control point. Subarea nodes use network addresses; peripheral nodes use local addresses or local-form session identifiers (LFSIDs). The boundary function in the subarea node to which a peripheral node is attached transforms local addresses or LFSIDs to network addresses and vice versa. Contrast with *network name*. (2) According to ISO 7498-3, a name, unambiguous within the OSI environment, that identifies a set of network service access points.

network control (NC). In SNA, a request/response unit (RU) category used for requests and responses exchanged between physical units (PUs) for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjust peripheral nodes. See also *data flow control*, *function management data*, and *session control*.

network control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

network management vector transport (NMVT). A management services request/response unit (RU) that flows over an active session between physical unit management services and control point management services (SSCP-PU session).

network name. (1) The symbolic identifier by which end users refer to a network accessible unit, a link, or a link station within a given subnetwork. In APPN networks, network names are also used for routing purposes. Contrast with *network address*. (2) In a multiple-domain network, the name of the APPL statement defining a VTAM application program. The network name must be unique across domains. Contrast with *ACB name*. See *uninterpreted name*.

network performance analyzer (NPA). A function of NCP that collects performance data about devices. The data is recorded by NPM.

Network Routing Facility (NRF). An IBM licensed program that resides in NCP. NRF provides a path for routing messages between terminals and routes messages over this path without going through the host processor.

Network Terminal Option (NTO). An IBM licensed program, used in conjunction with NCP, that allows certain non-SNA devices to participate in sessions with SNA application programs in the host processor. When data is sent from a non-SNA device to the host processor, NTO converts non-SNA protocol to SNA protocol; and when data is sent from the host processor to the non-SNA device, NTO converts SNA protocol to non-SNA protocol.

NMVT. Network management vector transport.

no response. In SNA, a protocol requested in the form-of-response-requested field of the request header that directs the receiver of the request not to return any response, regardless of whether or not the request is received and processed successfully. Contrast with *definite response* and *exception response*.

node. (1) In a network, a point at which one or more functional units connect channels or data circuits. (I) (2) Any device, attached to a network, that transmits and receives data. (3) An endpoint of a link or a junction common to two or more links in a network. Nodes can be processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.

NOTIFY. A network services request that is sent by a system services control point (SSCP) to a logical unit (LU) to inform the LU of the status of a procedure requested by the LU.

NPA. Network performance analyzer.

NPM. NetView Performance Monitor.

NPSI. X.25 NCP Packet Switching Interface.

NRF. Network Routing Facility.

NTO. Network Terminal Option.

O

OAF. Origin address field.

OAF'. Origin address field prime.

operand. (1) An entity on which an operation is performed. (I) (2) That which is operated upon. An operand is usually identified by an address part of an instruction. (A) (3) Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor. (4) An expression to whose value an operator is applied. See also *definition statement*, *keyword*, *keyword parameter*, and *parameter*.

Operating System/Virtual Storage (OS/VS). A family of operating systems that control IBM System/360 and System/370 computing systems. OS/VS includes VS1, VS2, MVS/370, and MVS/XA.

origin address field (OAF). In SNA, a field in a FID0 or FID1 transmission header that contains the address of the originating network accessible unit (NAU). Contrast with *destination address field*. See also *format identification (FID) field* and *local session identification (LSID)*.

origin subarea field (OSAF). In SNA, a subarea field in a FID4 transmission header that contains a subarea address, which combined with the element address in the origin element field, gives the complete network address of the originating network accessible unit (NAU). Contrast with *destination subarea field*.

OS/VS. Operating System/Virtual Storage.

OSAF. Origin subarea field.

P

pacing. A technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion. See *session-level pacing*, *send pacing*, and *virtual route (VR) pacing*. See also *flow control*.

pacing group. Synonym for *pacing window*.

pacing response. In SNA, an indicator that signifies the readiness of a receiving component to accept another pacing group. The indicator is carried in a response header (RH) for session-level pacing and in a transmission header (TH) for virtual route pacing.

pacing window. (1) The path information units (PIUs) that can be transmitted on a virtual route before a virtual-route pacing response is received, indicating that the virtual route receiver is ready to receive more PIUs on the route. (2) The requests that can be transmitted on the normal flow in one direction on a session before a session-level pacing response is received, indicating that the receiver is ready to accept the next group of requests. (3) Synonymous with *pacing group.*

packet. In data communication, a sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole. The data, control signals, and, possibly, error control information are arranged in a specific format. (I)

packet level. (1) The packet format and control procedures for exchange of packets containing control information and user data between data terminal equipment (DTE) and data circuit-terminating equipment (DCE). See also *data link level*, *higher level*, and *physical level.* (2) A part of Recommendation X.25 that defines the protocol for establishing logical connections between two DTEs and for transferring data on these connections.

packet mode operation. Synonym for *packet switching.*

packet switching. (1) The process of routing and transferring data by means of addressed packets so that a channel is occupied only during transmission of a packet. On completion of the transmission, the channel is made available for transfer of other packets. (I) (2) Synonymous with *packet mode operation.* See also *circuit switching.*

PAD. Packet assembler/disassembler.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) (2) In Basic CUA architecture, a variable used in conjunction with a command to affect its result. (3) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (4) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure. See also *keyword*, *keyword parameter*, and *operand.*

path. (1) In a network, any route between any two nodes. A path may include more than one branch. (T) (2) The series of transport network components (path control and data link control) that are traversed by the information exchanged between two network accessible units. See also *explicit route (ER)*, *route extension*, and *virtual route (VR).*

path control (PC). The function that routes message units between network accessible units in the network and provides the paths between them. It converts the basic information units (BIUs) from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units containing one or more PIUs with data link control. Path control differs by node type: some nodes (APPN nodes, for example) use locally generated session identifiers for routing, and others (subarea nodes) use network addresses for routing.

path information unit (PIU). A message unit consisting of a transmission header (TH) alone, or a TH followed by a basic information unit (BIU) or a BIU segment. See also *transmission header.*

PCID. Procedure-correlation identifier.

PDF. Parallel data field.

peripheral logical unit (LU). In SNA, a logical unit in a peripheral node.

peripheral node. A node that uses local addresses for routing and therefore is not affected by changes in network addresses. A peripheral node requires boundary-function assistance from an adjacent subarea node. A peripheral node can be a type 1, 2.0, or 2.1 node connected to a subarea boundary node.

peripheral path control. The function in a peripheral node that routes message units between units with local addresses and provides the paths between them. See *path control* and *subarea path control.* See also *boundary function*, *peripheral node*, and *subarea node.*

peripheral PU. In SNA, a physical unit (PU) in a peripheral node.

physical level. In X.25, the mechanical, electrical, functional, and procedural media used to activate, maintain, and deactivate the physical link between the data terminal equipment (DTE) and the data circuit-terminating equipment (DCE). See *data link level* and *packet level.*

physical unit (PU). The component that manages and monitors the resources (such as attached links and adjacent link stations) associated with a node, as requested by an SSCP via an SSCP-PU session. An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links. This term applies to type 2.0, type 4, and type 5 nodes only. See also *peripheral PU* and *subarea PU.*

physical unit (PU) services. In SNA, the components within a physical unit (PU) that provide configuration

services and maintenance services for SSCP-PU sessions. See also *logical unit (LU) services*.

PIU. Path information unit.

PLU. Primary logical unit.

positional operand. An operand in a language statement that has a fixed position. See also *definition statement*. Contrast with *keyword operand*.

procedure-correlation identifier (PCID). In SNA, a value used to correlate all requests and replies associated with a given procedure.

protection key. An indicator that appears in the current program status word whenever an associated task has control of the system. This indicator must match the storage keys of all main storage blocks that the task is to use.

PU. Physical unit.

R

RAS. Reliability, availability, and serviceability.

real address. The address by which a logical unit (LU) is known within the SNA network in which it resides.

RECMS. Record maintenance statistics.

record maintenance statistics (RECMS). An SNA error event record built from an NCP or line error and sent unsolicited to the host.

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

REGS. Registers.

remote. Pertaining to a system, program, or device that is accessed through a telecommunication line. Contrast with *local*. Synonym for *link-attached*.

request header (RH). The control information that precedes a request unit (RU). See also *request/response header (RH)*.

request unit (RU). A message unit that contains control information, end-user data, or both.

request/response header (RH). Control information associated with a particular RU. The RH precedes the request/response unit (RU) and specifies the type of RU (request unit or response unit).

request/response unit (RU). A generic term for a request unit or a response unit. See *request unit (RU)* and *response unit (RU)*.

response header (RH). A header, optionally followed by a response unit (RU), that indicates whether the response is positive or negative and that may contain a pacing response. See also *negative response*, *pacing response*, and *positive response*.

response unit (RU). A message unit that acknowledges a request unit. It may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session parameters in response to BIND SESSION). If negative, the response unit contains sense data defining the exception condition.

REX. Route extension.

RH. Request/response header.

route. (1) An ordered sequence of nodes and transmission groups (TGs) that represent a path from an origin node to a destination node traversed by the traffic exchanged between them. (2) The path that network traffic uses to get from source to destination.

route daemon. A program that runs under 4BSD UNIX to propagate route information among machines on a local area network. Also referred to as *routed* (pronounced "route-d").

route extension (REX). In SNA, the path control network components, including a peripheral link, that make up the portion of a path between a subarea node and a network addressable unit (NAU) in an adjacent peripheral node. See also *explicit route (ER)*, *path*, and *virtual route (VR)*.

routed. Pronounced "route-d." See *route daemon*.

router. (1) A computer that determines the path of network traffic flow. The path selection is made from several paths based on information obtained from specific protocols, algorithms that attempt to identify the shortest or best path, and other criteria such as metrics or protocol-specific destination addresses. (2) An attaching device that connects two LAN segments, which use similar or different architectures, at the reference model network layer. Contrast with *bridge* and *gateway*. (3) In OSI terminology, a function that determines a path by which an entity can be reached.

RR. Receive ready.

RU. Request/response unit.

S

SC. Session control.

scanner. (1) A device that examines a spatial pattern one part after another, and generates analog or digital signals corresponding to the pattern. Scanners are often used in mark sensing, pattern recognition, or character recognition. (I) (A) (2) For the 3725 communication controller, a processor dedicated to controlling a small number of telecommunication lines. It provides the connection between the line interface coupler hardware and the central control unit.

scanner interface trace (SIT). A record of the activity within the communication scanner processor (CSP) for a specified data link between an IBM 3725 Communication Controller and a resource.

SCB. (1) Session control block. (2) String control byte.

SCF. Secondary control field.

SDB. Storage descriptor block.

SDLC. Synchronous Data Link Control.

sequence number. In communications, a number assigned to a particular frame or packet to control the transmission flow and receipt of data.

session. (1) In network architecture, for the purpose of data communication between functional units, all the activities which take place during the establishment, maintenance, and release of the connection. (T) (2) A logical connection between two network accessible units (NAUs) that can be activated, tailored to provide various protocols, and deactivated, as requested. Each session is uniquely identified in a transmission header (TH) accompanying any transmissions exchanged during the session.

session activation request. In SNA, a request that activates a session between two network accessible units (NAUs) and specifies session parameters that control various protocols during session activity; for example, BIND and ACTPU. Contrast with *session deactivation request.*

session control (SC). In SNA, either of the following:

- One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification.
- A request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

session control block (SCB). In NPM, control blocks in common storage area for session collection.

session deactivation request. In SNA, a request that deactivates a session between two network accessible units (NAUs); for example, UNBIND and DACTPU. Synonymous with *generic unbind.* Contrast with *session activation request.*

session-level pacing. A flow control technique that permits a receiving half-session or session connector to control the data transfer rate (the rate at which it receives request units) on the normal flow. It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them. See *pacing* and *virtual route pacing.*

session partner. In SNA, one of the two network accessible units (NAUs) having an active session.

SIT. Scanner interface trace.

SLU. Secondary logical unit.

SNA. Systems Network Architecture.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network accessible units (NAUs), boundary function, gateway function, and intermediate session routing function components; and the transport network.

SNA network interconnection (SNI). The connection, by gateways, of two or more independent SNA networks to allow communication between logical units in those networks. The individual SNA networks retain their independence.

SNI. SNA network interconnection.

SS. (1) Start-stop. (2) Session services.

SSCP. System services control point.

SSP. System Support Programs.

start-stop (SS) transmission. (1) Asynchronous transmission such that each group of signals representing a character is preceded by a start signal and is followed by a stop signal. (T) (A) (2) Asynchronous transmission in which a group of bits is (a) preceded by a start bit that prepares the receiving mechanism for the

reception and registration of a character, and (b) followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending reception of the next character. See also *binary synchronous transmission* and *synchronous data link control*.

subarea. A portion of the SNA network consisting of a subarea node, attached peripheral nodes, and associated resources. Within a subarea node, all network accessible units (NAUs), links, and adjacent link stations (in attached peripheral or subarea nodes) that are addressable within the subarea share a common subarea address and have distinct element addresses.

subarea address. A value in the subarea field of the network address that identifies a particular subarea. See also *element address*.

subarea node (SN). A node that uses network addresses for routing and maintains routing tables that reflect the configuration of the network. Subarea nodes can provide gateway function to connect multiple subarea networks, intermediate routing function, and boundary function support for peripheral nodes. Type 4 and type 5 nodes can be subarea nodes.

subarea path control. The function in a subarea node that routes message units between network accessible units (NAUs) and provides the paths between them. See *path control* and *peripheral path control*. See also *boundary function*, *peripheral node*, and *subarea node*.

subarea PU. In SNA, a physical unit (PU) in a subarea node.

suboperand. One of multiple elements in a list comprising an operand. See also *definition statement*.

subvector. A subcomponent of the NMVT major vector.

supervisor call (SVC). A request that serves as the interface into operating system functions, such as allocating storage. The SVC protects the operating system from inappropriate user entry. All operating system requests must be handled by SVCs.

supervisor call instruction. An instruction that interrupts a program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction.

SVC. (1) Supervisor call. (2) Switched virtual circuit.

switched line. A telecommunication line in which the connection is established by dialing. Contrast with *nonswitched line*.

switched virtual circuit (SVC). An X.25 circuit that is dynamically established when needed. The X.25 equivalent of a switched line.

synchronous. (1) Pertaining to two or more processes that depend upon the occurrence of specific events such as common timing signals. (T) (2) Occurring with a regular or predictable time relationship.

Synchronous Data Link Control (SDLC). A discipline conforming to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute (ANSI) and High-level Data Link Control (HDLC) of the International Organization for Standardization, for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. (I) Contrast with *binary synchronous communication (BSC)*.

system services control point (SSCP). A component within a subarea network for managing the configuration, coordinating network operator and problem determination requests, and providing directory services and other session services for end users of the network. Multiple SSCPs, cooperating as peers with one another, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its own domain.

system slowdown. A network control program mode of reduced operation invoked when buffer availability drops below a threshold level. The network control program limits the amount of new data that the system accepts while continuing normal output activity.

System Support Programs (SSP). An IBM licensed program, made up of a collection of utilities and small programs, that supports the operation of the NCP.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The layered structure of SNA allows the ultimate origins and destinations of information, that is, the end users, to be independent of and unaffected by the specific SNA network services and facilities used for information exchange.

T

TAP. Synonym for *ACF/TAP*.

TCAM. Telecommunications Access Method. Synonymous with *ACF/TCAM*.

Telecommunications Access Method (TCAM). An access method used to transfer data between main storage and remote or local terminals.

TG. Transmission group.

TH. Transmission header.

trace. (1) A record of the execution of a computer program. It exhibits the sequences in which the instructions were executed. (A) (2) For data links, a record of the frames and bytes transmitted or received.

Trace Analysis Program (TAP). Synonym for *Advanced Communications Function for the Trace Analysis Program (ACF/TAP)*.

transmission group (TG). (1) A connection between adjacent nodes that is identified by a transmission group number. See also *parallel transmission groups*. (2) In a subarea network, a single link or a group of links between adjacent nodes. When a transmission group consists of a group of links, the links are viewed as a single logical link, and the transmission group is called a *multilink transmission group (MLTG)*. A *mixed-media multilink transmission group (MMMLTG)* is one that contains links of different medium types (for example, token-ring, switched SDLC, nonswitched SDLC, and frame-relay links). (3) In an APPN network, a single link between adjacent nodes.

transmission group (TG) vector. A representation of an endpoint TG in a T2.1 network, consisting of two control vectors: the TG Descriptor (X'46') control vector and the TG Characteristics (X'47') control vector.

transmission header (TH). Control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network. See also *path information unit*.

transmission priority. A rank assigned to a message unit that determines its precedence for being selected by the path control component in each node along a route for forwarding to the next node in the route.

U

UNBIND. In SNA, a request to deactivate a session between two logical units (LUs). See also *session deactivation request*. Contrast with *BIND*.

UP. Unnumbered poll.

user-written generation application. A user-written program that runs with the NCP/EP definition facility (NDF) during NCP generation. It processes definition statements and operands.

V

virtual machine (VM). In VM, a functional equivalent of a computing system. On the 370 Feature of VM, a virtual machine operates in System/370 mode. On the ESA Feature of VM, a virtual machine operates in System/370, 370-XA, ESA/370, or ESA/390 mode. Each virtual machine is controlled by an operating system. VM controls the concurrent execution of multiple virtual machines on an actual processor complex.

Virtual Machine/Enterprise Systems Architecture (VM/ESA). An IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a *real* machine.

Virtual Machine/Extended Architecture (VM/XA). An operating system that facilitates conversion to MVS/XA by allowing several operating systems (a production system and one or more test systems) to run simultaneously on a single 370-XA processor. The VM/XA Migration Aid has three components: the control program (CP), the conversational monitor system (CMS), and the dump viewing facility.

virtual route (VR). In SNA, either a) a logical connection between two subarea nodes that is physically realized as a particular explicit route or b) a logical connection that is contained wholly within a subarea node for intranode sessions. A virtual route between distinct subarea nodes imposes a transmission priority on the underlying explicit route, provides flow control through virtual route pacing, and provides data integrity through sequence numbering of path information units (PIUs). See also *explicit route (ER)*, *path*, and *route extension (REX)*.

virtual route (VR) pacing. In SNA, a flow control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route. VR pacing can be adjusted according to traffic congestion in any of the nodes along the route. See also *pacing* and *session-level pacing*.

virtual route identifier (VRID). In SNA, a virtual route number and a transmission priority number that, when combined with the subarea addresses for the subareas at each end of a route, identify the virtual route.

Virtual Storage Extended (VSE). An IBM licensed program whose full name is the Virtual Storage Extended/Advanced Function. It is a software operating system controlling the execution of programs.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VM. Virtual machine.

VM/ESA. Virtual Machine/Enterprise Systems Architecture.

VM/SP. Virtual Machine/System Product.

VM/XA. Virtual Machine/Extended Architecture.

VM/370. IBM Virtual Machine Facility/370.

VM/370 control program (CP). The component of VM/370 that manages the resources of a single computer with the result that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of an IBM System/370 computing system.

VR. Virtual route.

VRID. Virtual route identifier.

VS. Virtual storage.

VSE. Virtual Storage Extended. Synonymous with *VSE/Advanced Functions.*

VSE/Advanced Functions. The basic operating system support needed for a VSE-controlled installation. Synonym for *VSE.*

VSE/ESA. Virtual Storage Extended/Enterprise Systems Architecture.

VSE/SP. Virtual Storage Extended/System Package.

VTAM. Virtual Telecommunications Access Method. Synonymous with *ACF/VTAM.*

X

X.25. An International Telegraph and Telephone Consultative Committee (CCITT) recommendation for the interface between data terminal equipment and packet-switched data networks. See also *packet switching.*

X.25 NCP Packet Switching Interface (NPSI). An IBM licensed program that allows SNA users to communicate over packet switching data networks that have interfaces complying with CCITT Recommendation X.25. It allows SNA programs to communicate with SNA or non-SNA equipment over such networks.

XA. Extended architecture.

XID. Exchange identification.

XMIT. Transmit.

Bibliography

NCP, SSP, and EP Library

The following paragraphs briefly describe the library for NCP, SSP, and EP. The other publications dealing with the networking systems products—NTune, VTAM, NPSI, the NetView program, and NPM—are listed without the accompanying descriptions.

NCP V7R2, SSP V4R2, and EP R12 Library Directory (SC31-6259)

This book helps users locate information on a variety of NCP, SSP, and EP tasks. It also provides a high-level understanding of NCP, SSP, and EP and summarizes the changes to these products and to the library for NCP V7R2, SSP V4R2, and EP R12.

NCP V7R2 Migration Guide (SC31-6258)

This book helps users migrate an NCP generation definition from an earlier release to NCP V7R2. It also describes how to add new functions for NCP V7R2.

NCP, SSP, and EP Resource Definition Guide (SC31-6223)

This book helps users understand how to define NCP and EP (in the PEP environment) using SSP. It describes functions and resources and lists the definition statements and keywords that define those functions and resources.

NCP, SSP, and EP Resource Definition Reference (SC31-6224)

This book helps users code definition statements and keywords to define NCP and EP (in the PEP environment) using SSP. It also provides a quick reference of definition statement coding order and keyword syntax.

NCP, SSP, and EP Generation and Loading Guide (SC31-6221)

This book provides detailed explanations of how to generate and load NCP and EP (in the PEP environment) using SSP. It contains information for generating and loading under MVS, VM, and VSE.

NCP and SSP Customization Guide (LY43-0031)

This book helps users who are familiar with the internal logic of NCP and SSP to modify these products. It describes how to change NCP and SSP to support stations that IBM-supplied programs do not support.

NCP and SSP Customization Reference (LY43-0032)

This book supplements the *NCP and SSP Customization Guide*. It describes the resources and macroinstructions provided by IBM for customizing NCP and SSP.

NCP, SSP, and EP Messages and Codes (SC31-6222)

This book is a reference book of abend codes issued by NCP and EP in the PEP environment, and messages issued by the System Support Programs associated with NCP. This information is also available through the online message facility, an IBM OS/2 application available on diskette.

NCP, SSP, and EP Trace Analysis Handbook (LY43-0037)

This book describes how to use the trace analysis program and how to read trace analysis program output.

NCP, SSP, and EP Diagnosis Guide (LY43-0033)

This book helps users isolate and define problems in NCP and EP (in the PEP environment) using SSP. The primary purpose of the book is to help the user interact with the IBM Support Center to resolve a problem. In addition, it explains some of the diagnostic aids and service aids available with SSP.

NCP, SSP, and EP Diagnosis Aid (LK2T-1999, diskettes)

The Diagnosis Aid is an IBM OS/2 application used to diagnose NCP, SSP, and EP problems. This tool helps programmers and program support personnel who are responsible for isolating, diagnosing, and debugging problems in NCP and EP (in the PEP environment) using SSP. The Diagnosis Aid, available on diskette, provides online access to all the information contained in the *NCP, SSP, and EP Diagnosis Guide*, the *NCP and EP Reference Summary and Data Areas*, the *NCP, SSP, and EP Messages and Codes*, and the *NCP, SSP, and EP Trace Analysis Handbook*.

NCP and EP Reference (LY43-0029)

This book describes various aspects of the internal processing of NCP and EP in the PEP environment. It provides information for customization and diagnosis.

NCP and EP Reference Summary and Data Areas (LY43-0030)

This two-volume book provides quick access to often-used diagnostic and debugging information about NCP and EP in the PEP environment.

Other Networking Systems Products Libraries

The following publications provide cross-product information for NTune, VTAM, NPSI, NetView, and NPM. For detailed information about these products, refer to the library for each.

Networking Systems Library

The following list shows the publications in the Networking Systems library (this library currently contains information about NCP at the V7R1 level).

Planning for NetView, NCP, and VTAM (SC31-7122)

Planning for Integrated Networks (SC31-7123)

Planning Aids: Pre-Installation Planning Checklist for NetView, NCP, and VTAM (SX75-0092)

IBM Networking Systems Softcopy Collection Kit (CD-ROM, SK2T-6012)

IBM Online Libraries: Softcopy Collection Kit User's Guide (GC28-1700)

NTune Library

The following list shows the publications in the NTune library.

NTune User's Guide (SC31-6247)

NTuneNCP Reference (LY43-0035)

VTAM Library

The following list shows the publications in the VTAM V4R2 library.

VTAM Migration Guide (GC31-6491)

VTAM Release Guide (GC31-6492)

Estimating Storage for VTAM (SK2T-2007)

VTAM Network Implementation Guide (SC31-6494)

VTAM Resource Definition Reference (SC31-6498)

VTAM Resource Definition Samples (SC31-6499, book and diskettes)

VTAM Customization (LY43-0063)

VTAM Operation (SC31-6495)

VTAM Operation Quick Reference (SX75-0205)

Using IBM CommandTree/2 (SC31-7013)

VTAM Messages and Codes (SC31-6493)

VTAM Licensed Program Specifications (GC31-6490)

VTAM Programming (SC31-6496)

VTAM Programming Quick Reference (SX75-0206)

VTAM Programming for LU 6.2 (SC31-6497)

VTAM Diagnosis (LY43-0065)

VTAM Diagnosis Quick Reference (LX75-0204)

VTAM Data Areas for MVS (LY43-0064)

NPSI Library

The following list shows the publications in the NPSI Version 3 library.

X.25 NCP Packet Switching Interface General Information (GC30-3469)

X.25 NCP Packet Switching Interface Planning and Installation (SC30-3470)

X.25 NCP Packet Switching Interface Host Programming (SC30-3502)

X.25 NCP Packet Switching Interface Diagnosis, Customization, and Tuning (LY30-5610)

X.25 NCP Packet Switching Interface Data Areas (LY43-0034)

X.25 NCP Packet Switching Interface Master Index (GC31-6206)

NetView Library

The following list shows the publications in the NetView V2R4 library.

NetView General Information (GC31-7098)

Learning about NetView (SK2T-6017, diskettes)

Learning about NetView Graphic Monitor Facility (SK2T-6018, diskettes)

NetView Graphic Monitor Facility Reference Poster (SX75-0100)

NetView Automation Planning (SC31-7083)

NetView Storage Estimates (SK2T-6016, diskette for a PS/2 or a PS/55)

NetView Installation and Administration Guide (SC31-7084 for MVS)

NetView Installation and Administration Facility/2 Guide (or *NIAF/2 Guide*, SC31-7099)

NetView Administration Reference (SC31-7080)

NetView Bridge Implementation (SC31-6131)

NetView Tuning Guide (SC31-7079)

NetView Automation Implementation (LY43-0016)

NetView Customization Guide (SC31-7091)

NetView Customization: Writing Command Lists (SC31-7092)

NetView Customization: Using PL/I and C (SC31-7093)

NetView Customization: Using Assembler (SC31-7094)

NetView Operation (SC31-7086)

NetView Graphic Monitor Facility User's Guide (SC31-7089)

NetView Command Quick Reference (SX75-0090)

NetView Messages (SC31-7096)

NetView Resource Alerts Reference (SC31-7097)

NetView Application Programming Guide (SC31-7081)

NetView Resource Object Data Manager Programming Guide (SC31-7095)

NetView Problem Determination and Diagnosis (LY43-0101)

NPM Library

The following list shows the publications in the NPM V2 library.

NetView Performance Monitor at a Glance (GH19-6960)

NetView Performance Monitor Concepts and Planning (GH19-6961)

NetView Performance Monitor User's Guide (SH19-6962)

NetView Performance Monitor Messages and Codes (SH19-6966)

NetView Performance Monitor Graphic Subsystem (SH19-6967)

NetView Performance Monitor Installation and Customization (SH19-6964)

NetView Performance Monitor Reports and Record Formats (SH19-6965)

NetView Performance Monitor Diagnosis (LY19-6381)

NetView Performance Monitor Desk/2 User's Guide (SH19-6963)

Related Publications

The following publications, though not directly related to NCP, may be helpful in understanding your network.

3704/3705 Communication Controllers Assembler Language (GC30-3003)

IBM 3745 Communication Controller Publications

The following list shows selected publications for the IBM 3745 Communication Controller.

IBM 3745 Communication Controller Introduction (GA33-0092 for the 3745-210, 3745-310, 3745-410, and 3745-610)

IBM 3745 Communication Controller Introduction (GA33-0138 for the 3745-130, 3745-150, and 3745-170)

IBM 3745 Communication Controller Configuration Program (GA33-0093)

IBM 3745 Communication Controller (All Models): Principles of Operation (SA33-0102)

SNA Publications

The following publications contain information on SNA.

Systems Network Architecture Technical Overview (GC30-3073)

Systems Network Architecture Format and Protocol Reference Manual: Management Services (SC30-3346)

Systems Network Architecture Formats (GA27-3136)

Index

C

D

E

F

G

H

I

K

L

Q

R

S

T

U

V

W

X

Communicating Your Comments to IBM

Network Control Program
System Support Programs
Customization Reference

NCP Version 7 Release 2
SSP Version 4 Release 2

Publication No. LY43-0032-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  United States and Canada: **1-800-227-5088**
- If you prefer to send comments electronically, use this network ID:
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink*: **CIBMORCF at RALVM13**
  - Internet: **USIB2HPD@VNET.IBM.COM**

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Help us help you!

Network Control Program
System Support Programs
Customization Reference

NCP Version 7 Release 2
SSP Version 4 Release 2

Publication No. LY43-0032-01

We hope you find this publication useful, readable and technically accurate, but only you can tell us! Your comments and suggestions will help us improve our technical publications. Please take a few minutes to let us know what you think by completing this form.

Overall, how satisfied are you with the information in this book?	Satisfied	Dissatisfied
	□	□

How satisfied are you that the information in this book is:	Satisfied	Dissatisfied
Accurate	□	□
Complete	□	□
Easy to find	□	□
Easy to understand	□	□
Well organized	□	□
Applicable to your task	□	□

Specific Comments or Problems:

Please tell us how we can improve this book:

Thank you for your response. When you send information to IBM, you grant IBM the right to use or distribute the information without incurring any obligation to you. You of course retain the right to use the information in any way you choose.

Name

Address

Company or Organization

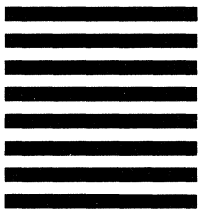
Phone No.

Help us help you!
LY43-0032-01

Cut o
Alon

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Information Development
Department E15
International Business Machines Corporation
PO BOX 12195
RESEARCH TRIANGLE PARK NORTH CAROLINA 27709-9990

Fold and Tape **Please do not staple** Fold and Tape

Cut
Alo

IBM®

File Number: S370/4300/30XX
Program Number: 5648-063
5655-041
5654-009
5686-064

Printed in U.S.A.

LY43-0032-01