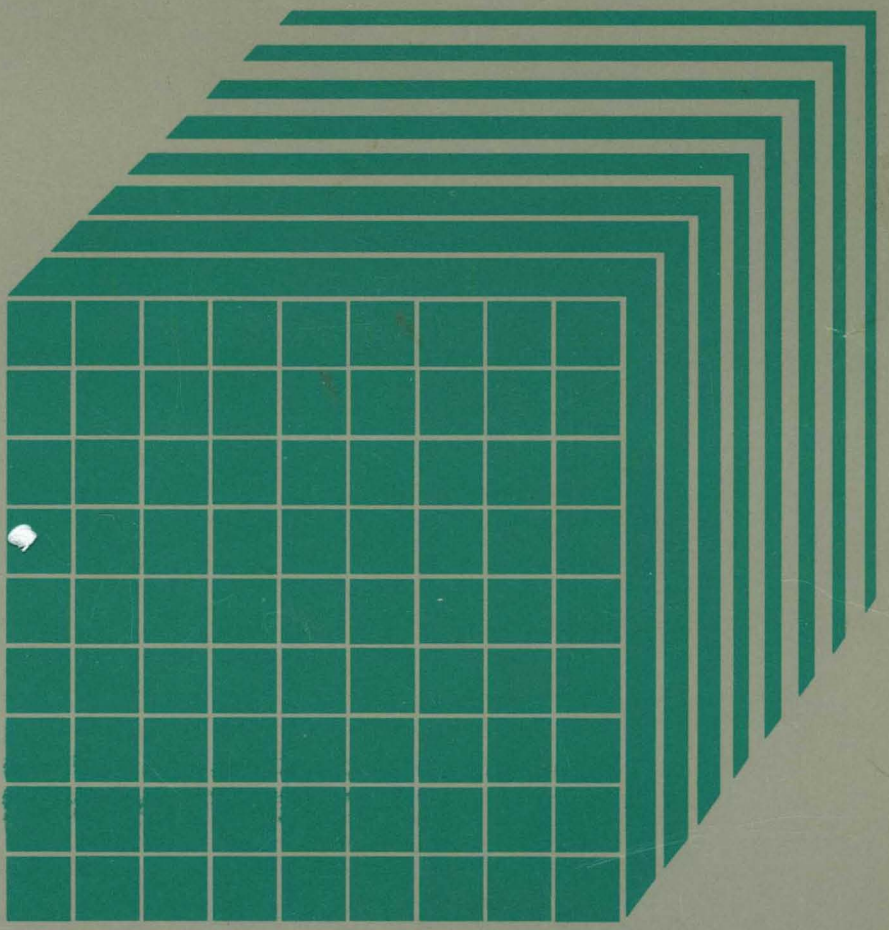
IBM

Virtual Machine/ System Product

System Facilities for Programming

Release 5

SC24-5288-0



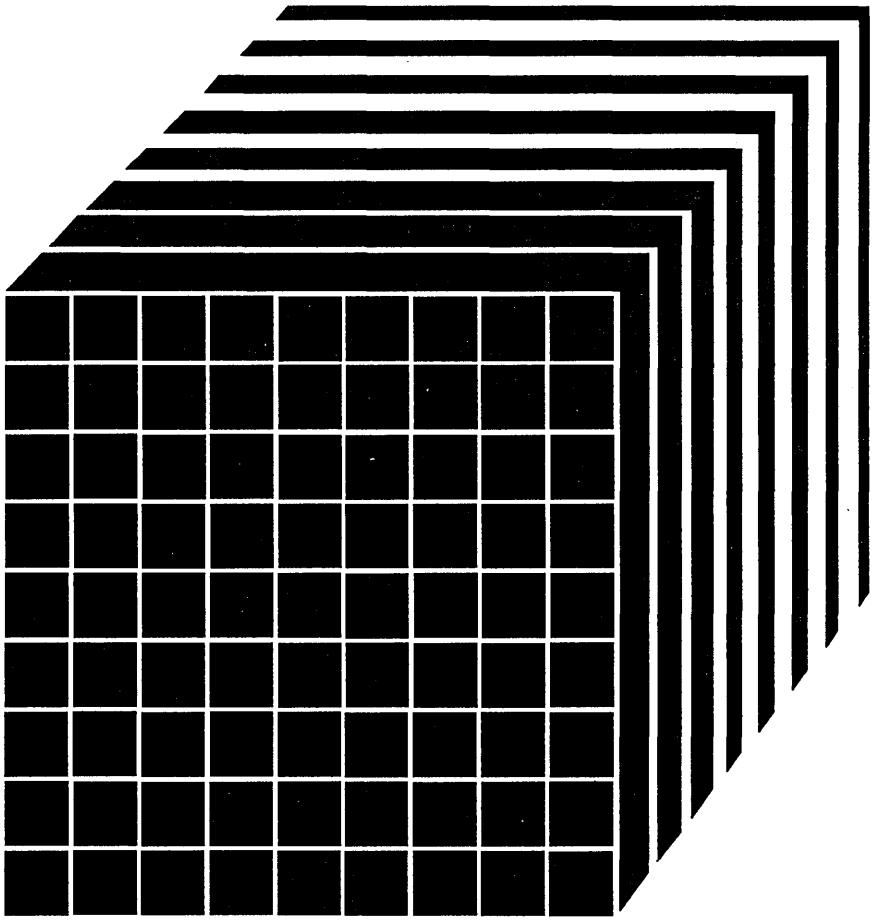
IBM

Virtual Machine/
System Product

System Facilities for Programming

Release 5

SC24-5288-0



First Edition (December 1986)

This edition, SC24-5288-0, applies to Release 5 of IBM Virtual Machine/System Product (VM/SP), program number 5664-167; Release 5 of IBM Virtual Machine/System Product High Performance Option (VM/SP HPO), program number 5664-173; and to all subsequent releases of these products until otherwise indicated in new editions or Technical Newsletters. It contains material formerly found in the *VM/SP System Programmer's Guide*, SC19-6203, and *VM/SP HPO System Programmer's Guide*, SC19-6224, (both discontinued after Release 4). Changes are made periodically to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a detailed list of changes, see "Summary of Changes" on page 431. This summary includes the changes from the last two editions of the *VM/SP System Programmer's Guide.*

Technical changes and additions to the text and illustrations are indicated by a vertical bar to the left of the change.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Ordering Publications

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Publications are *not* stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Dept. G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

VM System Facilities for Programming, SC24-5288, contains reference information pertaining to specific facilities of VM. It is intended for VM/SP and VM/SP HPO system programmers, system analysts, and others who know Basic Assembler Language and have experience with programming concepts and techniques. The material in this manual was previously found in the *VM/SP System Programmer's Guide*, SC19-6203 and the *VM/SP HPO System Programmer's Guide*, SC19-6224.

This manual provides reference information concerning facilities in VM, such as IUCV, VMCF, and the DIAGNOSE instruction. This manual is one of a set of reference manuals for VM system programmers. Other books in the set include

- *VM/SP CP for System Programming*, SC24-5285,
  or *VM/SP HPO CP for System Programming*, SC19-6224, if you have VM/SP HPO.
- *VM/SP CMS for System Programming*, SC24-5286
- *VM/SP Group Control System Command and Macro Reference*, SC24-5250
- *VM/SP Transparent Services Access Facility Reference*, SC24-5287
- *VM Diagnosis Guide*, LY24-5241

The order numbers for other books in the VM/SP and VM/SP HPO libraries can be found in the bibliography in the back of this manual.

This publication consists of two parts and two appendixes.

"Part 1. VM System Facilities" contains functional descriptions as well as guidance in using the following facilities and services provided with your VM system.

- The DIAGNOSE Instruction
- The Inter-User Communications Vehicle (IUCV)
- CMS IUCV
- CP System Services
- The Special Message Facility
- The Single Console Image Facility
- The Logical Device Support Facility
- The Virtual Machine Communication Facility (VMCF)

"Part 2. VM System Applications" contains a functional description, as well as guidance in using the following applications provided with your VM system:

- The Programmable Operator Facility
- Getting National Languages on Your System

"Appendix A: CP Device Classes, Types, Models, and Features for DIAGNOSE code X'24'" lists possible device type classes and device type values for DIAGNOSE code X'24'.

"Appendix B: Sample CMS IUCV Program" contains a short program using CMS IUCV support to use the CP Message System Service.

"Appendix C: Converting Programmable Operator Routing Tables" gives instruction on how to convert old routing tables (VM Release 2) to the current format.

In the back of the manual, five more sections may help you use this manual more easily:

Summary of Changes
: summarizes the enhancements made to this manual since the last edition was issued.

Abbreviations of Terms
: explains the acronyms and device numbers of IBM products that we use in this manual for convenience.

Glossary
: lists and defines technical terms used in this manual.

Bibliography
: lists publications that may be helpful to you while using your VM system and other related IBM licensed programs.

Index
: lists the content of this manual alphabetically with page numbers.

Contents

Part 1. VM System Facilities 1

Chapter 1. The DIAGNOSE Instruction in a Virtual Machine . . . 3
DIAGNOSE Code X'00' -- Store Extended-Identification Code 4
DIAGNOSE Code X'04' -- Examine Real Storage 7
DIAGNOSE Code X'08' -- Virtual Console Function 9
DIAGNOSE Code X'0C' -- Pseudo Timer 12
DIAGNOSE Code X'10' -- Release Pages 13
DIAGNOSE Code X'14' -- Input Spool File Manipulation 14
Subcode X'0000' -- Read Next Spool Buffer 15
Subcode X'0004' -- Read Next Print Spool File Block 16
Subcode X'0008' -- Read Next Punch Spool File Block 16
Subcode X'000C' -- Select a File For Processing 17
Subcode X'0010' -- Repeat Active File 'nnn' Times 17
Subcode X'0014' -- Restart Active File at Beginning 17
Subcode X'0018' -- Backspace One Record 17
Subcode X'001C' -- Read Next Monitor Spool File Block 18
Subcode X'0020' -- Read Next Monitor Spool Record 18
Subcode X'0024' -- Read Last Spool Buffer 18
Subcode X'0FFE' -- Select Next File Not Previously Selected 18
Subcode X'0FFF' -- Retrieve Subsequent File Descriptor 19
Programming Information for Users of DIAGNOSE Code X'14' 20
Reading a Virtual Spool File 21
DIAGNOSE Code X'18' -- Standard DASD I/O 22
DIAGNOSE Code X'1C' -- Clear Error Recording Cylinders 24
DIAGNOSE Code X'20' -- General I/O 25
DIAGNOSE Code X'24' -- Device Type and Features 27
DIAGNOSE Code X'28' -- Channel Program Modification 30
DIAGNOSE Code X'2C' -- Return DASD Start of LOGREC 31
DIAGNOSE Code X'30' -- Read One Page of LOGREC Data 32
DIAGNOSE Code X'34' -- Read System Dump Spool File 33
DIAGNOSE Code X'38' -- Read System Symbol Table 34
DIAGNOSE Code X'3C' -- Update the VM Directory 34
DIAGNOSE Code X'40' -- Clean-Up after Virtual IPL by Device 35
DIAGNOSE Code X'48' -- Issue SVC 76 from a Second Level VM/370 or VM Virtual Machine 36
DIAGNOSE Code X'4C' -- Generate Accounting Records for the Virtual User 36
Subcode X'0000' -- Subcode X'000C' 37
Subcode X'0010' 38
DIAGNOSE Code X'50' -- Save the 370X Control Program Image 39
DIAGNOSE Code X'54' -- Control The Function of the PA2 Function Key 40
DIAGNOSE Code X'58' -- 3270 Virtual Console Interface 40
Displaying Data 41
Full Screen Mode 43

DIAGNOSE Code X'5C' -- Error Message Editing 50
DIAGNOSE Code X'60' -- Determining the Virtual Machine Storage Size 52
DIAGNOSE Code X'64' -- Finding, Loading, and Purging a Named Segment 52
Subcodes X'0000' and X'0004' -- The LOADSYS Function 53
Subcode X'0008' -- The PURGESYS Function 54
Subcode X'000C' -- The FINDSYS Function 55
DIAGNOSE Code X'68' -- Virtual Machine Communication Facility (VMCF) 55
DIAGNOSE Code X'6C' -- Shadow Table Maintenance 57
DIAGNOSE Code X'70' -- Activating the Time-of-Day (TOD) Clock Accounting Interface 57
DIAGNOSE code X'74' -- Saving or Loading a 3800 Named System 59
DIAGNOSE Code X'78' -- MSS Communication 60
DIAGNOSE Code X'7C' -- Logical Device Support Facility 61
Description of Logical Device Support Facility Functions 65
External Interrupt Code X'2402' 67
Logical Device Interrupt Code X'2402' 68
Logical Device Restrictions 68
Migration/Coexistence 68
DIAGNOSE Code X'80' -- MSSFCALL 69
MSSF Command Words 70
DIAGNOSE Code X'84' -- Directory Update-In-Place 72
DIAGNOSE code X'8C' -- Access Certain Device Dependent Information 80
DIAGNOSE code X'94' -- VMDUMP Function 81
Supported Parameters 83
Dump Address Parameter List 85
DIAGNOSE Code X'98' -- Real Channel Program Support 89
Subcode X'0000' -- Lock a Virtual Page 90
Subcode X'0004' -- Unlock a Virtual Page 91
Subcode X'0008' -- Perform I/O on a Real CCW String 91
DIAGNOSE Code X'A0' -- Retrieve a Group Name 92
DIAGNOSE Code X'B0' -- Access Diagnostic Information Saved For Protected Application Facility Users 92
DIAGNOSE Code X'B4' -- Virtual Printer External Attribute Buffer Manipulation 94
External Attribute Buffer (XAB) 96
DIAGNOSE Code X'B8' -- Spool File External Attribute Buffer Manipulation 98
DIAGNOSE Code X'BC' -- Open A Spool File 100
DIAGNOSE Code X'C8' -- Set Language 101
DIAGNOSE Code X'CC' -- Saving the CP Message Repository 103
DIAGNOSE Code X'D0' -- Provide 3480 Tape Volume Serial Number .. 105
DIAGNOSE Code X'D4' -- Specify An Alternate Userid 106
DIAGNOSE Code X'D8'—System Spool Information 108
Subcode X'0000' -- Read next SFBLOK 110

Chapter 2. Inter-User Communications Vehicle 111
IUCV Paths 111
IUCV Messages 112
Message Data Transfer 112
Message Identification 113
IUCV External Interrupts 114

Avoiding IUCV External Interrupts 116
Security Considerations 116
Virtual Machine-to-Virtual Machine Communication 117
Using Data in a Buffer 117
Using Data in a Parameter List 120
Using Control Paths 121
Invoking IUCV Functions 123
IUCV Functional Descriptions 124
QUERY Function 127
DECLARE BUFFER Function 128
CONNECT Function 131
Connection Pending External Interrupt 135
ACCEPT Function 137
Connection Complete External Interrupt 140
SEND Function 142
Message Pending External Interrupt 147
RECEIVE Function 149
REPLY Function 154
Message Complete External Interrupt 158
REJECT Function 161
PURGE Function 164
SEVER Function 168
Connection Severed External Interrupt 170
RETRIEVE BUFFER Function 172
QUIESCE Function 173
Connection Quiesced External Interrupt 175
RESUME Function 176
Connection Resumed External Interrupt 178
TEST MESSAGE Function 179
DESCRIBE Function 181
TEST COMPLETION Function 184
SET MASK Function 188
SET CONTROL MASK Function 191
Trace Table Entries 193
IUCV Trace Table Entry Formats 194
Trace Table Entry Field Definitions 195
IUCV System Services 197

Chapter 3. CMS IUCV 199
HNDIUCV Macro 199
CMSIUCV Macro 204
Exits 211
Using CMS IUCV to Communicate Between Two Virtual Machines .. 212
Guidelines and Limitations of the CMS IUCV 215

Chapter 4. SNA Virtual Console Communication Services 219
System Structure 220
Environments Supported 221
Processing Descriptions 222
SNA CCS Entries in CP Internal Trace Table 233
Trace Table Entry Formats 233
Trace Table Entry Field Definitions 235

Chapter 5. The Message System Service 239

Establishing Communications with the Message System Service 239

Chapter 6. The Message All System Service 241

Chapter 7. The DASD Block I/O System Service 243
Establishing Communications with DASD Block I/O Service 243
IUCV CONNECT to the DASD Block I/O System Service 244
IUCV SEND to the DASD Block I/O System Service 245
Using the DASD Block I/O System Service from CMS 246

Chapter 8. The Signal System Service 251
Establishing Communications with the Signal System Service 251
IUCV CONNECT to the Signal System Service 252
Sending Signals 254
Receiving Signals 254
Leaving the Signal System Service 255

Chapter 9. The Error Logging System Service 257
Establishing Communications with the Error Logging System Service 257
Interactions 257

Chapter 10. The SPOOL System Service 259
Establishing Communications with the SPOOL System Service 259
IUCV CONNECT to the SPOOL System Service 260
IUCV SEND to the SPOOL System Service 261
The SELECT Function 262
The CLOSE Function 265
The MESSAGE Function 268
The READ Functions 270
The SPOOL System Service to a Logical Printer 273
The SEND Function 274
The NOTIFY Function 275
The PURGE Function 275

Chapter 11. The Special Message Facility 277

Chapter 12. Single Console Image Facility 279
Using the Single Console Image Facility 279

Chapter 13. Logical Device Support Facility 281

Chapter 14. The Virtual Machine Communication Facility 283
Using the Virtual Machine Communication Facility 284
VMCF Applications 285
Security and Data Integrity 286
Performance Considerations 287
General Considerations 288
VMCF Protocol 288
The SEND Protocol 289
The SEND/RECV Protocol 290
The SENDX Protocol 291
The IDENTIFY Protocol 292
Descriptions of VMCF Functions 293

The Control Functions 293
The Data Transfer Functions 296
Invoking VMCF Functions 300
DIAGNOSE Code X'68' 300
The VMCPARM Parameter List 301
External Interrupt Code X'4001' 305
The External Interrupt Message Header 306
VMCF User Doubleword 309
DIAGNOSE Code X'68' Return Codes 309
Data Transfer Error Codes 312

Part 2. VM System Applications 313

Chapter 15. The Programmable Operator Facility 315
Overview 315
Using the Programmable Operator Facility in Various System Environments 315
The Logical Operator 317
Routing Table Information 317
Action Routines 318
How it Works 318
Flow of Operation 319
Relationship to RSCS Networking 320
The Programmable Operator Virtual Machine 321
Installing the Programmable Operator Facility 321
Invoking the Programmable Operator Facility 322
Message Output Format 327
Stopping the Programmable Operator Facility 327
Running the Programmable Operator Facility from NCCF or NetView 328
The Programmable Operator/NCCF Message Exchange 328
Installing the PMX 328
Communication Between the Programmable Operator and NCCF or NetView 330
PMX Communication Protocol 331
Stopping the PMX 332
The Logical Operator 332
The Default Logical Operator 333
The NCCF or NetView Logical Operator 334
Assigning or Changing the Logical Operator 334
The Routing Table 336
How the Programmable Operator Facility Uses the Routing Table . 337
Routing Table Entry Formats 337
Tailoring the Routing Table 346
The Log File 354
Logging NCCF or NetView messages in the Log File 356
Ensuring a Complete Log 356
The Feedback File 358
Communications Checking 359
Invoking Programmable Operator Facility Commands 361
Programmable Operator Facility Command Descriptions 368
CMD Command 369
FEEDBACK Command 371
GET Command 372

LGLOPR Command 373
LOADTBL Command 375
LOG Command 377
QUERY Command 379
SET Command 382
STOP Command 385
Action Routines 386
The Action Routine Interface 387
Description of Supplied Action Routines 392
Exit EXECs 396
Exit EXEC Interface 396
Supplied Error Exit EXECs 397
Problem Determination - Debug Mode 398

Chapter 16. Getting National Languages on Your System 401
Contents of the Feature Tape 401
Source Files and Listing Files 402
Object Files 403
Installing National Language Files on Your VM System 404
Loading the National Language Files From Tape to Disk 404
Saving National Language Files for CP and CMS 404
Saving National Language Files for GCS 407
Adding National Language Information for an Application 407
Updating Files for an Existing National Language 409
Deleting a National Language 410
Deleting Language Information for an Application 410
The LANGMERG Command 411
LANGMERG's Control File 412
The LANGGEN Command 413
LANGGEN's Control File 415

Appendixes 417

Appendix A. CP Device Classes, Types, Models, and Features .. 419

Appendix B. Sample CMS IUCV Program 425

Appendix C. Converting Programmable Operator Routing Tables 429

Summary of Changes 431
Structural Changes 431
Technical Changes for VM System Facilities for Programming 433
Summary of Changes for VM/SP and VM/SP HPO System Programmer's Guides 435

Glossary of Terms and Abbreviations 443

Bibliography 449

Index 453

Figures

1. Data Returned to DIAGNOSE code X'00' 5
2. Format of Pseudo Timer Information 13
3. Addressable Storage Before and After a LOADSYS Function 53
4. DIAGNOSE Code X'84' -- Parameter List Operation Field 74
5. Suggested Format of an External Attribute Buffer 96
6. IUCV Two-Way Data Transfer 113
7. Sequence of Functions 118
8. CP System Services and Their Userids 197
9. Sequence of Instructions in Virtual Machine to Virtual Machine Communication 213
10. Virtual Console Support in CP 220
11. SNA Virtual Console Support Interfaces 224
12. Printer Subsystem Support 260
13. Summary of Logical Device Support Facility Functions 282
14. Virtual Machine Communication Facility (VMCF) Functions ... 284
15. The SEND Protocol 290
16. The SEND/RECV Protocol 291
17. The SENDX Protocol 292
18. The IDENTIFY Protocol 293
19. VMCF Functions, Parameters, and Return Codes 304
20. DIAGNOSE Code X'68' Return Codes 309
21. DIAGNOSE Code X'68' Data Transfer Error Codes 312
22. LGLOPR Command Authorization for an NCCF or NetView Operator 330
23. QUERY Command Authorization for an NCCF or NetView Operator 334
24. Sample LGLOPR Command Entries in a Routing Table 336
25. The Programmable Operator Facility in a Distributed System ... 341
26. Partial Routing Table 345
27. Routing Entries to Send Messages to an NCCF or NetView Operator 350
28. Routing Entries to Filter Responses to Routine Commands 351
29. Uncontrolled Authorization 352
30. Restricting Authorization by Nodeid 353
31. Restricting Authorization by Userid and Nodeid 353
32. Restricting Command Use to Specific Users 354
33. Example of Communication in the Single System Environment .. 363
34. Example of Communication in the Distributed Environment 364
35. Example of Communication in the Mixed Environment 366
36. Register Conventions for Invoking an Action Routine 388
37. CP Device Classes, Types, Models, and Features 419
38. New VM System Programming Manuals for Release 5 432

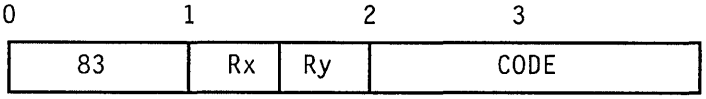
Part 1. VM System Facilities

Part 1 contains chapters about the following VM system facilities:

- The DIAGNOSE Instruction
- The Inter-User Communications Vehicle (IUCV)
- CMS IUCV
- The CP System Services
  - SNA Virtual Console Communication Services (*CCS)
  - The Message System Service (*MSG)
  - The Message All System Service (*MSGALL)
  - The DASD Block I/O System Service (*BLOCKIO)
  - The Signal System Service (*SIGNAL)
  - The Error Logging System Service (*LOGREC)
  - The SPOOL System Service (*SPL)
- The Special Message Facility
- The Single Console Image Facility
- The Logical Device Support Facility
- The Virtual Machine Communication Facility (VMCF)

Chapter 1. The DIAGNOSE Instruction in a Virtual Machine

The DIAGNOSE instruction cannot be used in a virtual machine for its normal function. If a virtual machine tries to execute a DIAGNOSE instruction, a program interrupt returns control to CP. Since a DIAGNOSE instruction issued in a virtual machine results only in returning control to CP and not in performing normal DIAGNOSE functions, the instruction is used for communication between a virtual machine and CP. The machine language format of DIAGNOSE is:



where:

83 is X'83' and is the S/370 operation code for the DIAGNOSE instruction.

Note: There is no mnemonic for DIAGNOSE.

Rx, Ry are general purpose registers that contain operand storage addresses or function codes passed to the DIAGNOSE functions, or return codes from the DIAGNOSE functions. If the registers contain addresses, those addresses must be real to the virtual machine issuing the DIAGNOSE[1]. The registers are specified as X'xy'. Unless otherwise noted, the register specified as Ry contains the return code on completion.

CODE is a two-byte hexadecimal value that CP uses to determine what DIAGNOSE function to perform, such as X'0008'. The codes defined for the general VM user are described in this section. The code must be a multiple of four. Codes X'00' through X'FC' are reserved for IBM use, and codes X'100' through X'1FC' are reserved for users. The privilege class for each code is indicated.

Because DIAGNOSE operates differently in a virtual machine than it does in a real machine, a program should determine that it is operating in a virtual machine before issuing a DIAGNOSE instruction, and prevent execution of a DIAGNOSE when in a real machine. The Store Processor ID (STIDP) instruction provides a program with information about the processor in which it is executing, including the processor version number. If STIDP is issued from a virtual machine, the version code, which precedes

---

[1] Except for DIAGNOSE code X'04' which examines real storage.

the CPUID field, will be X'FF'. For a preferred machine assist guest (VM/SP HPO), it is the real processor id. For a preferred machine assist guest with the control switch assist, CP returns X'FF'.

A virtual machine issuing a DIAGNOSE instruction should run with interrupts disabled. This prevents loss of status information about the DIAGNOSE operation such as condition codes and sense data.

Notes:

1. *A DIAGNOSE instruction with invalid parameters may at times result in a specification exception, protection exception, or addressing exception. Unauthorized use (not having the correct CP authorization class) results in a privileged operation exception.*

2. *If you change the privilege class for DIAGNOSE instructions using the OVERRIDE command, the privilege classes mentioned in this manual for DIAGNOSE instructions may no longer be correct for your installation.*

3. *If you have VM/SP HPO, you cannot use the DIAGNOSE instruction in a virtual machine when preferred machine assist is active unless control switch assist is also active. In the latter case, the following DIAGNOSE codes can be used:*

DIAGNOSE Code	Description
X'00'	*Store Extended-Identification Code*
X'04'	*Examine Real Storage*
X'08'	*Virtual Console Funciton*
X'40'	*Cleanup after Virtual IPL by Device*
X'4C'	*Generate Accounting Records*
X'68'	*Virtual Machine Communication Facility*
X'6C'	*Shadow Table Maintenance*
X'78'	*MSS Communication*
X'80'	*MSSFCALL*

DIAGNOSE Code X'00' -- Store Extended-Identification Code

All privilege classes (except ANY)

DIAGNOSE code X'00' allows a virtual machine to examine the extended-identification code. For example, an OS/VS1 virtual machine issues a DIAGNOSE code X'00' instruction to determine if the version of VM under which it is executing supports the VM/VS Handshaking feature. If the extended-identification code is returned to VS1, VM supports handshaking; otherwise, it does not.

Note: If you have VM/SP HPO, DIAGNOSE code X'00' is supported by a preferred machine assist guest with the control switch assist active.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'00':

Rx
Contains the doubleword aligned virtual storage address where the VM extended-identification code is to be stored.

Ry
Contains the number of bytes to be stored entered as an unsigned binary number.

Exit Values: If the VM system currently executing does not support the DIAGNOSE code X'00' instruction, no data is returned to the virtual machine. If it does support the DIAGNOSE code X'00' instruction, the data in Figure 1 is returned to the virtual machine (at the location specified by Rx):

Field	Description	Characteristics
System Name	"VM/SP"	8 bytes, EBCDIC
RESERVED	(for IBM use)	3 bytes, zeroes
Version Code	VM/SP executes the STIDP (Store Processor ID) instruction to determine the version code.	1 byte, hexadecimal
MCEL	VM/SP executes the STIDP instruction to determine the maximum length of the MCEL (Machine Check Extended Logout) area.	2 bytes, hexadecimal
Processor Address	VM/SP executes the STAP (Store Processor Address) instruction to determine the processor address.	2 bytes, hexadecimal
Userid	The userid of the virtual machine issuing the DIAGNOSE.	8 bytes, EBCDIC

Figure 1 (Part 1 of 2). Data Returned to DIAGNOSE code X'00'

Field	Description	Characteristics
Licensed Program Bit Map	Identifies the licensed programs that are installed. Valid values and the licensed programs each identifies are: Value — Licensed Program X'8000000000000000' Basic System Extensions 2 X'4000000000000000' System Extensions, Release 2 X'2000000000000000' VM/SP, Release 1 X'1000000000000000' VM/SP, Release 2 X'0800000000000000' VM/SP, Release 3 X'0400000000000000' VM/SP, Release 4 X'0200000000000000' VM/SP, Release 5 X'0080000000000000' VM/SP HPO, Release 1 X'0040000000000000' VM/SP HPO, Release 2 X'0020000000000000' VM/SP HPO, Release 2.5 X'0010000000000000' VM/SP HPO, Release 3 X'0008000000000000' VM/SP HPO, Release 3.2 X'0004000000000000' VM/SP HPO, Release 3.4 X'0002000000000000' VM/SP HPO, Release 3.6 X'0001000000000000' VM/SP HPO, with 3880 Model 21 support X'0000800000000000' VM/SP HPO, with VDLE support X'0000400000000000' VM/SP HPO, Release 4.0 X'0000200000000000' VM/SP HPO, Release 4.2 X'0000100000000000' VM/SP HPO, with Scheduler/Monitor changes X'0000080000000000' VM/SP HPO, Release 5.0 *Note: These bits are cumulative. For example, if your system is VM/SP Release 4, the bits would be on for all releases of VM/SP giving you X'FC000000000000000'. If your system is VM/SP HPO Release 4.2 and has all of the listed features, all the bits would be on (including all the bits for VM/SP since VM/SP Release 4 is a prerequisite) giving you X'FCFDF000000000000'.*	8 bytes, hexadecimal
Time Zone Value	Represents the time zone differential in seconds from Greenwich Mean Time. *Note: The Time Zone Value is a signed hexadecimal fullword value in seconds. Negative values represent differentials west of Greenwich Mean Time and positive values represent differentials east of Greenwich Mean Time.*	4 bytes, hexadecimal
Version Number	The first byte is the release number, the second byte is the release modification level, the third and fourth bytes are the PLC (Program Level Change) number.	4 bytes, hexadecimal

Figure 1 (Part 2 of 2). Data Returned to DIAGNOSE code X'00'

If VM is executing in a virtual machine, another 40 bytes, or less, of extended identification data is appended to the first 40 bytes described above. Up to five nested levels of VM virtual machines are supported by this DIAGNOSE instruction resulting in a maximum of 200 bytes of data

that can be returned to the virtual machine that initially issued the DIAGNOSE instruction.

On return, Ry contains its original value less the number of bytes that were stored.

Condition and Return Codes: No return code is received, and the condition code remains unchanged.

DIAGNOSE Code X'04' -- Examine Real Storage

Privilege class C or E

DIAGNOSE code X'04' allows a user to examine real storage.

Note: If you have VM/SP HPO, DIAGNOSE code X'04' is supported by a preferred machine assist guest with the control switch assist active.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'04':

Rx
Contains the virtual address of a list of CP (real) addresses to be examined.

Ry
Contains the count of entries in the list.

Ry+1
Contains the virtual address of the result field. The result field contains the values retrieved from the specified real locations.

Exit Values: For each address in the list of CP addresses, VM provides a fullword of data obtained from the specified address in real storage. VM stores this data into the result field identified by Ry+1.

There is a one-to-one correspondence between entries in the list of addresses and entries in the result field. For example, data obtained from the address in the first entry of the address list is stored in the first entry of the result field, data obtained from the second entry of the address list is stored in the second entry of the result field, and so forth.

Program Exceptions: If DIAGNOSE code X'04' is specified incorrectly, the following exceptions are received:

Addressing
If the requested CP storage is outside the real machine storage size.

Specification

If Ry contains a zero or negative count, or Ry is greater than 1024.

If the request list and the result list are not in the same virtual page.

If the request list and the result list cross a page boundary.

If the requested real page frame is disabled.

Notes:

1. *The request and result tables must be in the same page of virtual storage, and that page must be resident in real storage, at the time the DIAGNOSE is executed. This is guaranteed if the instruction itself is also in the same page.*

2. *In the attached processor or multiprocessor environment, each processor has a prefix register to relocate addresses between 0 and 4095 to another page frame in main storage. The prefix register enables each processor to use a different page frame to avoid conflict with the other processor for such activity as interrupt code recording. Thus, the range 0 through 4095 refers to different areas of storage, depending on which processor generates the address.*

   In attached processor mode, all references to main storage from either processor are handled as if they were made on the main processor. In multiprocessor mode, references to main storage from either processor are handled as if they were made on the IPL processor. Existing user programs remain valid for performance data; they receive the statistics for the main (or IPL) processor.

   References to the PSA of the attached processor (or non-IPL processor, in multiprocessor mode) may be made as follows: first, retrieve the value of PREFIXB, the value of the prefix register for the other processor (the attached processor in this case). Next, specify addresses that are the sum of the value of PREFIXB and the PSA displacement. References to 0 through 4095 are made by summing the value of PREFIXA and the PSA displacement to form the request address. Several system values that are processor independent are maintained in 0 through 4095, such as the restart PSW and the trace table vectors.

   For details on attached processor and multiprocessor environments, please see VM/SP CP for System Programming or VM/SP HPO CP for System Programming.

3. *If a reference is made to a real page frame that CP has determined to be disabled, results cannot be predicted. The CORETABLE entry corresponding to the real page address is checked and, if a disabled condition is found, the operation is terminated and a program check for a specification exception is presented to the virtual machine.*

4. *If you have VM/SP HPO, extended storage support allows CP to use storage above the 16Mb line. However, DIAGNOSE code X'04' cannot access storage above 16Mb.*

DIAGNOSE Code X'08' -- Virtual Console Function

All privilege classes (except ANY)

DIAGNOSE code X'08' enables a virtual machine running in supervisor state to issue CP commands. The virtual machine must specify the command, the command parameters, and whether CP is to return the command response to the user's terminal or to a buffer. In addition to returning the command response, CP sets a return code in Ry and may set a condition code.

Note: If you have VM/SP HPO, DIAGNOSE code X'08' is supported by a preferred machine assist guest with the control switch assist active.

Entry values: Set up the input registers as follows when invoking DIAGNOSE code X'08':

Rx
Must point to the character string in virtual storage that contains the CP commands and parameters. If the character string contains multiple commands, each command and its associated parameters must be separated from adjacent commands by the value X'15'.

Ry
Contains flag bits in the high-order byte; the other three bytes specify, in bytes, the length of the CP commands and parameters. The maximum allowable length is 240 characters.

Set the flag bits in the high-order byte of Ry as follows:

X'80' For CP to reject a password entered on the same line as a LINK command. CP rejects passwords only if the installation specified password suppression during system generation.

X'40' For CP to return the command response in a buffer.

X'20' To make the virtual machine responsible for prompting the user for the LINK or AUTOLOG password and reissuing the LINK or AUTOLOG command with the correct password in the command buffer.

If the command response is to be returned in a buffer, Rx and Ry cannot be consecutive registers nor can either be register 15. In addition, Rx+1 and Ry+1 must be setup as follows:

Rx+1
Must point to the buffer in virtual storage where CP is to return the command response.

Ry+1
Must specify, in bytes, the length of the buffer.

Exit values: If Ry contains the value X'00000000', the DIAGNOSE code acts as a no-operation (NOP) instruction. As a consequence, the issuing virtual machine is placed into a CP-READ state.

Condition Codes: If the command response is to be returned in a buffer, CP sets a condition code and returns information as follows:

CC=0 The request was successful. Rx+1 points to the buffer that contains the command response. Ry+1 specifies the length of the response.

CC=1 The request was unsuccessful. The response does not fit into the buffer. Ry+1 contains a value that specifies how many bytes of the response would not fit into the buffer.

Return Codes: When CP returns to a program executing a DIAGNOSE code X'08' instruction, the length value that was supplied in Ry is replaced by the CP return code value. This value is 0 if the CP console function was successfully executed. If an error occurred, the return code is the numeric value expressed in the message describing the error (unless X'20' is specified in the high-order bit of Ry). For example, if error message DMKCFM045E is issued, CP sets a return code of 45.

If the virtual machine assumes responsibility for prompting and a password is not in the command buffer on a LINK or AUTOLOG command issued through DIAGNOSE code X'08', and the information in the command buffer is valid, one of five unique return codes is passed back to the virtual machine. These return codes indicate which password prompt the virtual machine should issue. The short logon prompt and extended logon prompt return codes are 8013 and 8014, respectively, for the AUTOLOG command. The read, write, and multi password prompt return codes are 8015, 8016, and 8017, respectively, for the LINK command. The link or autolog request should then be reissued via another DIAGNOSE code X'08' with the password in the command buffer.

If the user has not specified a command response buffer, error messages and informational messages are generated according to the current values established by SET EMSG, SET IMSG, and SET MSG commands.

If a command response buffer is used, error and informational messages are always put into the buffer instead of being written to the console. Each line of the response is followed by a new line character (X'15'). If the buffer is not long enough to contain all of the response lines, only as many complete lines as can fit into the buffer are supplied, so the last character written into the response buffer by CP is always a new line character. Any unused portion of the response buffer is not changed. The setting of EMSG

determines only whether the error message code is retained. (SET EMSG OFF is treated the same as SET EMSG ON; SET EMSG TEXT suppresses error message codes.) Messages affected by SET MSG are not put into the command response buffer unless MSG is set on (SET MSG ON).

The return code values returned by CP are not affected by the values of EMSG and IMSG, or by using a command response buffer.

If CP is executing multiple commands and encounters an invalid command, processing stops and CP ignores the remaining commands.

Service Virtual Machine System Integrity: Certain virtual machines process or manage either data or other resources for multiple applications or users. At the interfaces these service virtual machines provide to user virtual machines, there is the potential for accidental or intentional misuse of the interface. Such misuse could lead to exposures to data security or data integrity.

Where such interfaces allow the user directly or indirectly to request or specify functions to be performed by the control program, it is the responsibility of the service virtual machine to determine that the functions to be performed are valid for the environment in which they will be performed.

For example, the CP Virtual Console Function facility allows the character string pointed to by Rx to contain more than one CP command. If a service virtual machine were to accept such a string from an unauthorized user virtual machine, and then pass the string to CP's Virtual Console Function facility without checking the validity of the command or commands and the command parameters in the string, then it is possible that a data security or data integrity problem could occur.

Programs that accept character data, which is embedded in strings passed to CP, should ensure that such data does not contain the X'15' character unless it is specifically desired that multiple commands be permitted.

Example: Following are two examples showing how to specify DIAGNOSE code X'08'. The first example shows how a program issues the QUERY FILES command. In this example the response is returned to the user's terminal. Note that in virtual storage environment, a load real address (LRA) instruction must be used to load Rx.

```
          LA1     6,CMMD
          LA      10,CMMDL
          DC      X'83',X'6A',XL2'0008'
           .
           .
           .
CMMD      DC      C'QUERY FILES'
CMMDL     EQU     *-CMMD
           .
           .
```

The second example shows how to specify a string of commands when multiple commands are to be issued.

```
             LA1     6,CMMD
             LA      10,CMMDL
             DC      X'83',X'6A',XL2'0008'
              .
              .
              .
CMMD         DC      C'QUERY FILES'
             DC      X'15'
             DC      C'PURGE PRINTER'
CMMDL        EQU     *-CMMD
```

Notes:

1. *If you are in EC mode you must code a LRA instruction instead of a LA instruction if you are running a virtual storage system (for example, MVS) in a virtual machine and want to specify the address of the CMMD parameter.*

2. *The logical line editing characters (described in the VM/SP Terminal Reference and under the TERMINAL command in the VM/SP CP Command Reference or the VM/SP HPO CP Command Reference) are only recognized by CP when entered from a terminal, not when passed to CP via DIAGNOSE code X'08'. Therefore a command such as #CP is not recognized by CP when issued via DIAGNOSE code X'08' and results in error message "DMKCFC001E Unknown CP command: name". The value X'15', as described in the "Entry Values", is the only value recognized by DIAGNOSE code X'08' for issuing multiple commands.*

DIAGNOSE Code X'0C' -- Pseudo Timer

All privilege classes (except ANY)

DIAGNOSE code X'0C' causes CP to store four doublewords of time information in the user's virtual storage.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'0C':

Rx
Contains the address of the 32-byte area where the time information is to be stored. The address must be on a doubleword boundary. The information returned is in the format shown below.

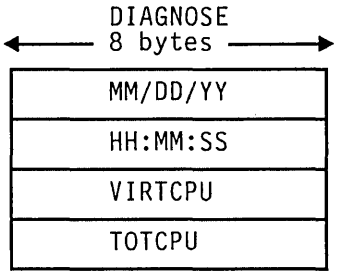


Figure 2. Format of Pseudo Timer Information

The first eight bytes contain the Month/Day-of-Month/Year. The next eight bytes contain the time of day in Hours:Minutes:Seconds. The last 16 bytes contain an unsigned binary number that represents the virtual and total processor time (in microseconds) of the virtual machine that issued the DIAGNOSE.

Condition and Return Codes: No return code is received, and the condition code remains unchanged.

DIAGNOSE Code X'10' -- Release Pages

All privilege classes (except ANY)

Pages of virtual storage can be released by issuing DIAGNOSE code X'10'. A released page is considered all zero.

Do not use DIAGNOSE code X'10' to release noncontiguous storage; use DIAGNOSE code X'64' for this purpose.

Entry values: Set up the input registers as follows when invoking DIAGNOSE code X'10':

Rx
Contains the address of the first page to be released.

Ry
Contains the address of the last page to be released.

Both addresses must be on page boundaries. A page boundary is a storage address whose low-order three digits, expressed in hexadecimal, are zero.

Condition and Return Codes: No return code is received, and the condition code remains unchanged.

DIAGNOSE Code X'14' -- Input Spool File Manipulation

All privilege classes (except ANY)

DIAGNOSE code X'14' causes DMKDRDER to manipulate the input spool files.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'14':

Rx
Contains a buffer address, a copy count, or a spool file identifier depending on the value of the function specified.

Ry
Which must be an even register, contains either the virtual address of a spool input card reader or, if Ry+1 contains X'0FFF', a spool file ID number.

Ry+1
Contains a hexadecimal code indicating the file manipulation to be performed, and a flag with the optional size of the spool file block.

The function subcodes are:

Code	Function
0000	Read next spool buffer (data record)
0004	Read next print spool file block (SFBLOK)
0008	Read next punch spool file block (SFBLOK)
000C	Select a file for processing
0010	Repeat active file *nnn* times
0014	Restart active file at beginning
0018	Backspace one record
001C	Read next monitor spool file block
0020	Read next monitor spool record
0024	Read last spool buffer (active file)
0FFE	Select next file not previously selected
0FFF	Retrieve subsequent file descriptor

Notes:

1. *Subcodes X'001C' and X'0020' are the only subcodes of DIAGNOSE code X'14' that can be used for monitor files.*

2. *For subcodes X'0000', X'0004', X'0008', X'000C', X'001C', and X'0020', held files are skipped.*

Condition Codes: On return Ry+1 may contain error codes that further define a returned condition code of 3.

Condition Code	Ry+1	Error
0		Data transfer successful
1		End of file or if subcode X'0018' and file is at first record
2		File not found
3	4	Device address invalid
3	8	Device type invalid
3	12	Device busy, reader not ready, or device is a real device
3	16	Fatal paging I/O error
3	20	Page already locked for I/O
3	24	File in use by system; probable paging or spooling error.

Subcode X'0000' -- Read Next Spool Buffer

Rx = start address of full-page virtual buffer
Ry = virtual spool reader address
Ry+1 = function subcode

The specified device is checked for a file activated via DIAGNOSE. If one is found, the next full-page buffer is made available to the virtual machine via a call to DMKRPAGT. If a file is not found, the chain of reader files is searched for a file for the calling user and connected to the virtual device for further reading. If no file is found, virtual condition code 2 is set. When the end of an active file is reached, the device status settings are tested for "spool continuous." If not set, virtual condition code 1 is set, indicating end of file. If the device is set for continuous input, the active file is examined to determine if it is a multiple-copy file. If it is, reading is restarted at the beginning of the file. If it is not, the file is closed via DMKVSUCR and the reader chain is searched for another input file. If no other file is found, virtual condition code 1 is set. A specific DIAGNOSE code X'14' subcode X'0000' must be issued to get the first spooled page again.

Notes:

1. *Subcode X'0000' returns a 3 condition code if an active monitor file or CP dump file is found.*

2. *Issuing DIAGNOSE code X'14' subcode X'0000' against a locked page causes the page to become unlocked.*

3. *For Subcode X'0000', held files are skipped.*

Subcode X'0004' -- Read Next Print Spool File Block

Rx = virtual address of an SFBLOK buffer
Ry = virtual spool reader address
Ry+1 = flag, optional size of SFBLOK in doublewords, and function subcode.

If the specified device is in use via DIAGNOSE, the VSPLCTL block is checked to see if this is a repeated call for printer SFBLOKs. If it is, then the chain search continues from the point where the last SFBLOK was given to the virtual machine. In this case, CC=1 is set when there are no more print files. If this is the first call for an SFBLOK, or if there have been intervening calls for file reading, the spool input chain is searched from the beginning, and CC=2 is set if no files are found.

If the high-order byte of the subcode register (Ry+1) is zero, then only 13 doublewords of the SFBLOK are returned and the rest could be truncated. However, if bit zero of the register is on, then bits 2 to 7 specify the amount of data to be returned (in doublewords). If the actual SFBLOK is shorter, the extra space is filled with zeroes.

Notes:

1. *If the virtual buffer specified by Rx crosses a page boundary, a specification exception results.*

2. *For Subcode X'0004', held files are skipped.*

3. *For Subcode X'0004', the format definition for a VM SFBLOK can be found in the system macro library.*

Subcode X'0008' -- Read Next Punch Spool File Block

Rx = virtual address of an SFBLOK buffer
Ry = virtual spool reader address
Ry+1 = flag, optional size of SFBLOK in doublewords, and function subcode.

Processing for subcode X'0008' is the same as for subcode X'0004', except that only punch files are processed.

Notes:

1. *For Subcode X'0008', held files are skipped.*

2. *For Subcode X'0008', the format definition for a VM SFBLOK can be found in the system macro library.*

Subcode X'000C' -- Select a File For Processing

Rx = file identifier of requested file
Ry = virtual spool reader address
Ry+1 = function subcode

The spool input chain is searched for the file specified. If it is not found, CC=2 is set. If it is found, the file is moved to the head of the chain so that it is the next file processed by any of the other functions.

Note: For Subcode X'000C', held files are skipped.

Subcode X'0010' -- Repeat Active File 'nnn' Times

Rx = new copy count (nnn) for the active file
Ry = virtual spool reader address
Ry+1 = function subcode

The specified device is checked for an active file. If no file is active, CC=2 is set. Otherwise, the copy COUNT (nnn) for the file is set to the specified value, with a maximum of 255. If the specified count is not positive, a specification exception is generated.

Subcode X'0014' -- Restart Active File at Beginning

Ry = virtual spool reader address
Ry+1 = function subcode

The specified device is checked for an active file. If no active file is found, CC=2 is set. Otherwise, the VSPLCTL pointers are reset to the beginning of the file.

Subcode X'0018' -- Backspace One Record

Rx = start address of virtual full-page buffer
Ry = virtual spool reader address
Ry+1 = function subcode

The specified device is checked for an active file. If no active file is found, CC=2 is set. Otherwise, the file is backspaced one record and the record is given to the user as in subcode X'0000'. If the file is already positioned at the first record, the first record is given to the user.

Subcode X'001C' -- Read Next Monitor Spool File Block

Rx = virtual address of an SFBLOK buffer
Ry = virtual spool reader address
Ry+1 = flag, optional size of SFBLOK in doublewords, and function subcode.

Processing is the same as subcode X'0008', except that only monitor spool files, as identified by the SFBMON flag is SFBFLAG2, can be handled.

Note: For Subcode X'001C', held files are skipped.

Subcode X'0020' -- Read Next Monitor Spool Record

Rx = start address of virtual full-page buffer
Ry = virtual spool reader address
Ry+1 = function subcode

Processing is the same as subcode X'0000', except that only monitor spool files, as identified by the SFBMON flag in SFBFLAG2, can be handled.

Note: For Subcode X'0020', held files are skipped.

Subcode X'0024' -- Read Last Spool Buffer

Rx = start address of virtual full-page buffer
Ry = virtual spool reader address
Ry+1 = function subcode

The specified device is checked for an already active file. If there is one, the last full-page buffer is made available to the virtual machine via a call to DMKRPAGT. If there is no active file, CC=2 is set.

Subcode X'0FFE' -- Select Next File Not Previously Selected

Rx = virtual address of a 332-byte buffer
Ry = code to further determine function
Ry+1 = flag, optional size of SFBLOK in doublewords, and function subcode.

If Ry code = 0, the next reader spool file that was not previously seen is selected and returns data[2] to the user's buffer.

---

[2] The **data** for the X'0FFE' and X'0FFF' subcodes of DIAGNOSE code X'14' are SFBLOK, 40 bytes of the 3800 data from the first SPLINK (if requested), the first CCW, the following TIC, and up to 136 bytes of TAG data.

If Ry code = 1, the bit in the SFBLOK is be reset to indicate that the spool file was previously selected and data[2] from the first spool file is returned to the user. CC=1 is returned if no file is found.

If Ry is neither 0 or 1, a specification exception error is reflected.

Subcode X'0FFE' waits for a file being used by a system function. If, however, the file is not available within the 250 millisecond time limit, a condition code of 3, RC of 24 is returned. This condition indicates system problems because of performance or errors in the spooling area.

Subcode X'0FFF' -- Retrieve Subsequent File Descriptor

Rx = virtual address of a 332-byte buffer
Ry = spool file ID number
Ry+1 = flag, optional size of SFBLOK in doublewords, and function subcode.

If Ry is nonzero, the spool input chain is searched for a file with a matching ID number: If none is found or if one is found that is owned by a different virtual machine (VM/SP only), CC=2 is set. The chain search is continued from the file that was found, or from the anchor if Ry is zero, for the next file owned by the caller, independent of file type, class, etc. If none is found, CC=1 is set. If a file is found but it has the INUSE flag on, CC=3 (RC=12) is returned. Otherwise, the data[2] is returned to the user's buffer.

As with subcode X'0FFE', subcode X'0FFF' also waits for a file being used by a system function. If, however, the file is not available within the 250 millisecond time limit, a condition code of 3, RC of 24 is returned. This condition indicates system problems because of performance or errors in the spooling area.

Note: Data chaining may occur when 3800 load CCW's are present in a spool file. If the data following a 3800 load CCW is more than 4080 bytes long, that data cannot be contained in one DASD spool file buffer. Instead, the CCW is data-chained to succeeding DASD buffers until all the data has been entered into the spool file. If the file contains 3800 load CCW's, either the SFBLDBEG or the SFBLDMID flags are set in the SFBLOK.

The amount of SFBLOK data returned is calculated as described under subcode X'0004'. In addition, if bit zero of the subcode register (Ry+1) is on, 40 bytes of 3800 data is returned immediately following the SFBLOK and preceding the TAG data. The data returned is described in the SPLINK DSECT starting at label SPCHAR.

Programming Information for Users of DIAGNOSE Code X'14'

To optimize storage and performance of spooled data transmitted between VM and other systems, CP automatically:

- Truncates blanks from the end of each data line, and
- Merges carriage control CCW commands.

This processing can cause the loss of significant data.

The rest of this section explains how SPOOL File Compression support corrects the problem of lost data. Programmers should find the information useful if their applications require a complete record image (i.e., all trailing blanks intact).

The DIAGNOSE code X'14' interface includes subcodes that allow an application program to read the control blocks associated with a virtual spool file:

SFBLOK
The **Spool File Block** retains all the information relating to a spool file.

SPLINK
The **Spool Page Buffer Linkage Block** resides in auxiliary storage and contains one page (4096 bytes) of unit record spool information consisting of data and all required CCWs.

Note: A detailed map of the SFBLOK and SPLINK fields is given in *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* and *VM/SP HPO Data Areas and Control Block Logic - CP.*

Data records are stored in the **spool buffer data area** which begins at X'10' displacement into the SPLINK. The format of the data records are as follows:

- For a printer (PRT), punch (PUN), or console (CON) file created by a virtual output device the spool buffer data area takes the form of a "Virtual Channel Program." That is, it is formatted like a channel program (containing CCWs interspersed with data), but every address is specified as a relative displacement from the beginning of the closest "data moving" CCW.
- For a reader (RDR) file created by a real card reader, the spool buffer data area takes the form of a "Real Channel Program." In this case each address is specified as a complete address, but since the address is based on the 4K page of storage that was used at the time of creation, it is not a reliable value for use by an application program.

Reading a Virtual Spool File

1. Read an SFBLOK (subcodes X'0004', X'0008', X'0FFE', or X'0FFF'). SFBTYPE can be used to determine whether the file was created by a system reader or a virtual output device. SFBRECSZ contains the logical record length for the file. If the SFBVLEN bit is on in SFBFLAG4, the the files contains "Original Length" information for each record.

2. Read a Spool Page Buffer (subcodes X'0000', X'0018', or X'0024'). SPRECNUM contains the number of data records in the buffer.

3. Start working at a displacement of SPSIZE into the buffer.

   For a file created by a system reader:

   a. Use a fixed displacement (SPSIZE = 96) to the beginning of each entry.

   b. The 80-byte data record is located 12 bytes into the entry.

   c. SFBRECNUM specifies the number of entries in the buffer.

   For a file created by a virtual output device:

   a. Immediate operations (e.g., Skip to Channel 1) take up 8 bytes.

   b. Data movers (e.g., Write and Space 2) occupy a variable amount of space depending on the length of the data and the type of operation:

```
CCW ................ 8 bytes
TIC CCW ............ 4 bytes
Data Record ........ variable
Original Length[3] .... 2 bytes
Backchain .......... 2 bytes (PUNCH only)
Pad ................ 0-7 bytes
```

   Control Records do not contain an original length field. Only Punch records reserve an additional 2 bytes to insure that a 4-byte work area is available after the last byte of the data record.[4]

---

[3] The "Original Length" field is the length of the record presented to the virtual machine's spooling device before the trailing blank suppression algorithm.

[4] An "Original Length" field is not included on files created in VM Release 4 or earlier.

DIAGNOSE Code X'18' -- Standard DASD I/O

All privilege classes (except ANY)

DIAGNOSE code X'18' allows a virtual machine to perform input/output operations to a direct access device, of the type used by CMS. CP returns no I/O interrupts to the virtual machine; the DIAGNOSE instruction completes only when the READ or WRITE commands associated with the DIAGNOSE complete[5].

Note: If you have VM/SP HPO, DIAGNOSE code X'18' only supports CCW read/write codes of X'05' and X'06'.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'18':

Rx
Contains the virtual device address of the direct access device.

Ry
Contains the address of a chain of CCWs.

R15
Contains with the number of READs or WRITEs in the CCW chain. The CCW chain must be in a standard format that CP expects when DIAGNOSE code X'18' is used, as shown below.

Use: DIAGNOSE code X'18' checks that the byte count from the user's read or write CCW does not exceed 4096 bytes. If the byte count exceeds 4096 bytes, CP flags it as an error. If the byte count is less than or equal to 4096 bytes, DIAGNOSE code X'18' makes an additional check for valid CMS standard block-sizes. The standard CMS block-sizes are 512, 800, 1K, 2K, or 4K bytes. The latter check is necessary and only pertinent in the event that the user's channel program is directed to a device that is capable of executing extended count-key-data channel commands (for example, a 3380 attached to a 3880 Control Unit equipped with the Speed Matching Buffer Feature).

CP converts user's channel programs to the extended count-key-data (CKD) format when the channel programs:

- Are directed to a 3380 attached to a 3880 Control Unit equipped with the Speed Matching Buffer (Feature #6550). (The Speed Matching Buffer is not supported for 3380 Models AD4/BD4 or AE4/BE4.)
- Are directed to a 3375 attached to a 3880 Control Unit equipped with the Speed Matching Buffer (Feature #6560).

---

[5] For non-standard channel programs (more than one consecutive READ or WRITE CCWs chained together), no extended CCW is transformed if this is directed to a 3380 with the Speed Matching Buffer.

And when they:

- Contain READ or WRITE CCW's with valid CMS block-sizes.
- Contain no READs chained to READs or WRITEs which are, themselves, chained to WRITEs.

An example of a channel program converted to an extended count-key-data channel program is shown below.

DIAGNOSE code X'18' must not be used to read or write record-overflow-formatted data.

A typical CCW string to read or write two 800-byte records is as follows:

```
   SEEK,A,CC,6
   SET SECTOR (not used for 2314/2319)
   SRCH,A+2,CC,5
   TIC,*-8,0,0
   RD or WRT,DATA,CC+SILI,800
   SEEK HEAD,B,CC,6 (omitted if HEAD number unchanged)
   SET SECTOR
   SRCH,B+2,CC,5
   TIC,*-8,0,0
   RD or WRT,DATA+800,SILI,800

A  SEEK and SRCH arguments for first RD/WRT
B  SEEK and SRCH arguments for second RD/WRT
```

If you are reading from or writing to either a 3380 or 3375 attached to a 3880 Control Unit equipped with the respective Speed Matching Buffer, the above sample channel program would be converted to the following extended count-key-data CCWs:

```
DEFINE EXTENT,C,CC,16
LOCATE RECORD,D,CC,16
RD OR WRT,DATA,CC+SILI,800
LOCATE RECORD,E,CC,16
RD OR WRT,DATA+800,SILI,800

C = DEFINE EXTENT argument
D = LOCATE RECORD argument for first RD/WRT
E = LOCATE RECORD argument for second RD/WRT
```

Note: The second LOCATE RECORD CCW shown in this example is not generated in all cases. That is, LOCATE RECORD CCWs, after the first one, are generated only when one of the following is encountered:

- A READ is followed by a WRITE, or vice versa, with the normal SEEK, SET SECTOR, SRCH in between them.
- The length of a READ or WRITE is not the same as the length of the preceding READ or WRITE.
- The READ or WRITE that follows a previous READ or WRITE is not for the next sequential record on the track.

Condition and Return Codes: The codes returned are as follows:

CC=0 I/O complete with no errors. R15 is set to 0.

CC=1 Error condition. Register 15 contains one of the following return codes:

R15	Meaning
1	Device not attached
2	Device not 2319, 2314, 3330, 3340, 3350, 3375, or 3380
3	Attempt to write on a read-only disk
4	Cylinder number not in range of user's disk
5	Virtual device is busy or has an interrupt pending

CC=2 Error condition. Register 15 contains one of the following return codes:

R15	Meaning
5	Pointer to CCW string not doubleword-aligned.
6	SEEK/SEARCH arguments not within range of user's storage.
7	CCW is not a SEEK, SEEK HEAD, SET SECTOR, SEARCH ID, TIC*-8, READ, or WRITE or an invalid CCW string was submitted.
8	READ/WRITE byte count=0
9	READ/WRITE byte count greater than 4096
10	READ/WRITE buffer not within user's storage
11	The value in R15, at entry, was not a positive number from 1 through 15, or was not large enough for the given CCW string.
12	Cylinder number on seek head was not the same number as on the first seek.

CC=3 Uncorrectable I/O error:

R15	Meaning
13	CSW (8 bytes) returned to user. Sense bytes are available if the user issues a SENSE command.

Note: This code does not support fixed-block DASD devices. If a program issues a DIAGNOSE code X'18' to a fixed-block DASD device, CP sets CC=1 and places a return code of 2 in register 15.

DIAGNOSE Code X'1C' -- Clear Error Recording Cylinders

Privilege class F

DIAGNOSE code X'1C' allows a user to clear the error recording data on disk. The DMKIOEFM routine performs the clear operation.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'1C':

Rx
Contains a one-byte code value in the low-order byte as follows:

Code	*Function*
X'01'	Clear and reformat all error recording, leaving any frame records intact
X'02'	Clear and reformat all error recording cylinders, erasing both frame records and error records

DIAGNOSE Code X'20' -- General I/O

All privilege classes (except ANY)

With DIAGNOSE code X'20', a virtual machine user can specify any valid CCW chain to be performed on a tape, disk (including FBA) or unit record device. (An exception: DIAGNOSE must not be used to read or write record-overflow-formatted data on DASD devices.) No I/O interrupts are reflected to the virtual machine; the DIAGNOSE instruction is completed only when all I/O commands in the specified CCW chain are finished.

Notes:

1. *Virtual spooled devices, such as, card readers and punches, are not supported for this DIAGNOSE. That is, unless the virtual device is a minidisk, a real device must be attached to the virtual machine.*

2. *If you have VM/SP HPO, DIAGNOSE code X'18' only supports CCW read/write codes of X'05' and X'06'.*

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'20':

Rx
Contains the virtual device address.

Ry
Contains the address of the CCW chain, and CP uses the high-order byte of the register as a storage key for accessing the user's virtual storage.

The CCWs are processed via DMKCCWTR through DMKGIOEX, providing full virtual I/O in a synchronous fashion (self-modifying CCWs are not permitted, however) to any virtual machine specified. Control returns to the virtual machine only after the operation is completed or a fatal error condition is detected. EREP support is provided for tape and DASD devices only; all other devices present an error condition in the PSW to the virtual

machine. Condition codes and return codes are returned to the virtual system.

With VM/SP HPO, for virtual I/O to a 3880 Model 13 or Model 23 that is not dedicated, the following CCW commands are valid if the virtual machine is a cache owner:

- Set Subsystem Mode
- Sense Subsystem Status
- Sense Subsystem Counts.

If the virtual machine is not a cache owner, CP treats the above CCWs as invalid.

Also, for the above CCWs, CP does not try to translate storage director data (track addresses and device addresses) to or from virtual machine addresses. For the storage director data to be meaningful to a virtual machine that uses these CCWs, the virtual machine addresses must map to the real device addresses. Virtual channel addresses need not map to real channel addresses.

Set High Performance Limits is a 3880 Model 13-only command. If this command is issued to a 3880 Model 13 or Model 23, CP treats it as an invalid command.

Note: If multiple errors occur on error recovery during DIAGNOSE code X'20', the CCW address in the CSW may be unpredictable.

Completion and Condition Codes: The condition codes and return codes are as follows:

CC=0 I/O completed with no errors

CC=1 Error condition. Register 15 contains the following return codes:

R15	Meaning
1	Device is either not attached or the virtual channel is dedicated, the device is virtual and not DASD (minidisk).
5	Virtual device is busy or has an interrupt pending.

CC=2 Exception conditions. Register 15 contains one of the following return codes:

R15	Meaning
2	Unit exception bit in device status byte=1
3	Wrong length record detected.

CC=3 Error Condition:

R15	Meaning
13	A permanent I/O error occurred or an unsupported device was specified. The user's Ry contains four sense bytes. Sense bytes 2 and 3 are in the two leftmost positions in Ry; sense byte 0 and 1 are in the two rightmost positions in Ry. For an imprecise ending error condition, the CSW does not reflect the failing CCW. In this case residual count is contained in sense byte 3 providing for the calculation of the correct failing CCW.

Ry

Sense Byte 2	Sense Byte 3	Sense Byte 0	Sense Byte 1

DIAGNOSE Code X'24' -- Device Type and Features

All privilege classes (except ANY)

DIAGNOSE code X'24' requests CP to provide a virtual machine with identifying information and status information about a specified virtual device. The virtual machine must specify the virtual device for which information is requested. CP returns information about the virtual device and associated real device in Rx, Ry, and Ry+1. CP also provides a condition code identifying the specific device information returned to the virtual machine.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'24':

Rx
Must contain the virtual device address for which information is requested or the value negative 1 (-1). Specify -1 when the device is a virtual console whose address is unknown to the virtual machine.

Exit Values: When CP returns control to the virtual machine, Ry, Ry+1, and Rx contain device information. Ry contains information about the virtual device and Ry+1 contains information about the real device. If -1 was specified and CP located the virtual console, Rx contains the address of the virtual console.

CP obtains device information from three control blocks: virtual device information from the virtual device block (VDEVBLOK), and real device information from the real device block (RDEVBLOK) and from NICBLOK. The following diagrams identify specific information returned by CP and show how to locate this information in the Rx, Ry, and Ry+1 registers. The symbolic names used in these diagrams are the symbolic names used with VDEVBLOK, RDEVBLOK, and NICBLOK in *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* and *VM/SP HPO Data Areas and*

Control Block Logic - CP. For more device information see Appendix A, "CP Device Classes, Types, Models, and Features."

Note: For a DIAGNOSE code X'24' to an SNA device though VCNA or VSCS, the model (RDEVMDL) information is correct; however, the RDEVTYPE may not be reliable.

Rx

Byte 0	Byte 1	Byte 2	Byte 3
RDEVTMCD —or— NICTMCD		virtual device address	

Symbolic Name	*Meaning*
RDEVTMCD - or - NICTMCD	Terminal code bits defining the type of console and the translate table the console is using. RDEVTMCD is for a local virtual console; NICTMCD for a remote 3270 virtual console.

Ry

Byte 0	Byte 1	Byte 2	Byte 3
VDEVTYPC	VDEVTYPE	VDEVSTAT	VDEVFLAG

Symbolic Name	*Meaning*
VDEVTYPC	Virtual device type class
VDEVTYPE	Virtual device type
VDEVSTAT	Virtual device status
VDEVFLAG	Virtual device flags

Ry+1

Byte 0	Byte 1	Byte 2	Byte 3
RDEVTYPC	RDEVTYPE - or - NICDTYPE	RDEVMDL - or - NICMDL	RDEVFTR - or - RDEVLLEN - or - NICLLEN

Symbolic Name	*Meaning*
RDEVTYPC	Real device type class
RDEVTYPE	Real device type

RDEVMDL	Real device model number. To determine if the speed matching buffer for the 3380 or 3375 is present, check if bits 0 and 1 are set on. For VM/SP HPO, the Speed Matching Buffer is not supported for 3380 Models AD4/BD4 or AE4/BE4.
RDEVFTR	Real device feature code for a device other than a virtual console
RDEVLLEN	Current device line length for a local virtual console
NICDTYPE	Real device type for a remote 3270 virtual console
NICMDL	Real device model number for a remote 3270 virtual console
NICLLEN	Current device line length for a remote virtual console

Notes:

1. *RDEVTYPE may not be reliable for SNA devices through VCNA or VSCS.*

2. *Remote dialed terminals and remote dedicated printers appear to be local devices as RDEVTYPC will contain the value CLASGRAF.*

3. *Also note that a remote dialed 3275 appears as a local 3277 and a remote dialed 3276 appears as a local 3278.*

4. *Remote dedicated printers internally carry a virtual device class and type of CLASGRAF and TYP3277 to follow code for remote dialed terminals. DIAGNOSE code X'24' returns VDEVTYPC = CLASGRAF and VDEVTYPE = TYP3284 for remote dedicated printers.*

Condition Codes: The condition codes CP can return for DIAGNOSE code X'24' are listed below. Please note that Rx contains information only when DIAGNOSE code X'24' specifies a virtual console whose address is unknown. And if Ry is register 15, CP returns only virtual device information; no information is returned in Ry+1.

Condition Code	Meaning
0	Normal completion. Data is returned in Rx, Ry, and Ry+1.
1	Undefined.
2	The virtual device exists but is not associated with a real device. Data is returned in Rx and Ry.
3	An invalid address is specified, or the virtual device does not exist.

DIAGNOSE Code X'28' -- Channel Program Modification

All privilege classes (except ANY)

DIAGNOSE code X'28' allows a virtual machine to correctly execute some channel programs modified after the Start I/O (SIO) instruction is issued and before the input/output operation is completed. The channel command word (CCW) modifications allowed are:

- A Transfer in Channel (TIC) CCW modified to a No Operation (NOP) CCW
- A TIC CCW modified to point to a new list of CCWs
- A NOP modified to a TIC CCW.

When a virtual machine modifies a TIC CCW, it is modifying a virtual channel program. CP has already translated that channel program and is waiting to execute the real CCWs. The DIAGNOSE instruction, with DIAGNOSE code X'28', must be issued to inform CP of the change in the virtual channel program, so that CP can make the corresponding change to the real CCW before it is executed. In addition, when a NOP CCW is modified to point to a new list of CCWs, CP translates the new CCWs.

To be sure that the DIAGNOSE instruction is recognized in time to update the real CCW chain, the virtual machine issuing the DIAGNOSE instruction should have a high favored execution value and a low dispatching priority value. The CP SET command should be issued:

```
SET FAVORED xx

SET PRIORITY nn
```

where xx has a high numeric value and nn has a low numeric value. The virtual machine issuing DIAGNOSE code X'28' must be in the supervisor mode at the time it issues the DIAGNOSE instruction.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'28':

Rx
Contains the address of the TIC or NOP CCW that was modified by the virtual machine.

Ry
Contains the device address in bits 16 through 31.

Rx and Ry cannot be the same register. The addresses specified in Rx, the new address in the modified TIC CCW, and the new CCW list to which the modified TIC CCW points must all be addresses that appear real to the virtual machine: CP knows these addresses are virtual, but the virtual machine thinks they are real.

Condition and Return Codes: The condition codes (CC) and return codes are as follows:

CC=0 The real channel program was successfully modified; register 15 contains a zero.

CC=1 The channel program was not modified. There was probably an error in coding the DIAGNOSE instruction. Register 15 (R15) contains one of the following return codes:

R15	Meaning
1	The same register was specified for Rx and Ry.
2	The device specified by Ry was not found.
3	The address specified by Rx was not within the user's storage space.
4	The address specified by Rx was not doubleword aligned.
5	A CCW string corresponding to the device (Ry) and address (Rx) specified was not found.
6	The CCW at the address specified by Rx is not a TIC nor a NOP, or the CCW in the channel program is not a TIC nor a NOP.
7	The new address in the modified TIC CCW is not within the user's storage space.
8	The new address in the modified TIC CCW is not doubleword aligned.
11	The new virtual CCW is a NOP, but the corresponding real CCW is a TIC with command chaining and is at the end of the real channel program.

CC=2 The real channel program cannot be modified because of the state of the system or the device. A channel end or device end has already occurred. Register 15 (R15) contains the following:

R15	Meaning
9	The virtual machine should restart the modified channel program.

DIAGNOSE Code X'2C' -- Return DASD Start of LOGREC

Privilege class C, E, or F

DIAGNOSE code X'2C' allows a user to find the location on the disk of the error recording area, the number of error recording cylinders, and the location of the first error record.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'2C':

Rx
Contains a one-byte code in the low-order byte, indicating the function to be performed:

Code	Function
X'01'	Return the DASD location of the start of the error recording area, and the number of error recording cylinders.
X'02'	Return the HDRSTART value (DASD location of first error record).
X'04'	Return indication of whether there are frame records on the error recording cylinders.

Exit Values: On return to the issuer of DIAGNOSE code X'2C' the registers contain the following:

If code X'01' is specified: Rx contains the DASD location (in VM control program internal format) of the start of the error recording area. Ry contains, in the low-order halfword, the number of error recording cylinders.

If code X'02' is specified: Rx contains the DASD location of the first error record (in CCPD format). The value actually points to the last frame record written, or record 2 if no frame records present.

If code X'04' is specified: Ry contains a X'20' in the low-order byte if frame records are present on the error recording cylinders; X'00' if no frame records present.

Note: Codes X'02' and X'04' may both be specified (code X'06') on invoking DIAGNOSE. Both an Rx and Ry value must be specified.

DIAGNOSE Code X'30' -- Read One Page of LOGREC Data

Privilege class C, E, or F

DIAGNOSE code X'30' allows a user to read one page of the system error recording area.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'30':

Rx
Contains the DASD location (in CP internal format) of the desired record.

Ry
Contains the virtual address of a page-size buffer to receive the data. The DMKRPAGT routine supplies the page of data.

Condition Codes: The condition codes returned are:

Condition Code	Meaning
0	Successful read, data available
1	End of cylinder, no data
2	I/O error
3	Invalid location, outside recording area

Note: Issuing DIAGNOSE code X'30' against a locked page unlocks the page.

DIAGNOSE Code X'34' -- Read System Dump Spool File

Privilege class C or E

A user can read the system dump spool file by issuing a DIAGNOSE code X'34' instruction. However, this DIAGNOSE code cannot read spool files that contain VMDUMP records -- use DIAGNOSE code X'14' for this purpose. If a program tries to use DIAGNOSE code X'34' to read VMDUMP records, CP returns a condition code of 2.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'34':

Rx
Contains the virtual address of a page-size buffer to receive the data.

Ry
Which must not be register 15, contains the virtual address of the spool input card reader.

Condition Codes: Ry+1, on return, may contain error codes as follows:

Condition Code	Ry+1 Error Code	Meaning
0		Data transfer successful
1		End of file
2		File not found
3	4	Device address invalid
3	8	Device type invalid
3	12	Device busy
3	16	Fatal paging I/O error

The DMKDRDMP routine searches the system chain of spool input files for the dump file belonging to the user issuing the DIAGNOSE instruction. The first (or next) record from the dump file is provided to the virtual

machine via DMKRPAGT and the condition code is set to zero. The dump file is closed via VM console function CLOSE.

Note: Issuing DIAGNOSE code X'34' against a locked page unlocks the page.

DIAGNOSE Code X'38' -- Read System Symbol Table

Privilege class C or E

DIAGNOSE code X'38' causes the routine DMKDRDSY to read the system symbol table into storage.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'38':

Rx
Contains the address of the page buffer to contain the symbol table.

Condition Codes: When complete, Ry, which must not be register 15, contains a condition code. On return, Ry+1 may contain an error code.

Condition Code	Ry+1 Error Code	Meaning
0		Full page of data available to virtual machine
1		No symbol table is available
3		Page buffer is locked for an I/O operation
3	16	Fatal paging I/O error

Notes:

1. *The format of the symbol table entries is described in CP macro SYM.*

2. *Issuing DIAGNOSE code X'38' against a locked page unlocks the page.*

DIAGNOSE Code X'3C' -- Update the VM Directory

Privilege class A, B, or C

DIAGNOSE code X'3C' lets a user dynamically update the VM directory. The routine DMKUDRDS dynamically updates the directory.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'3C':

Rx
Contains the first 4 bytes of the volume identification.

Ry
Contains the last 2 bytes of the volume identification in its first 2 bytes. The last 2 bytes of Ry contain the volume address.

Condition Codes: The PSW condition code is set depending on the success of the operation or the meaning of the condition code. The condition codes are set as follows:

Condition Code	Meaning
0	Operation is successful.
2	Volume not found, not mounted, or not a valid directory volume.
3	Fatal I/O error trying to read the directory

DIAGNOSE Code X'40' -- Clean-Up after Virtual IPL by Device

All privilege classes (except ANY)

DIAGNOSE code X'40' is valid only during virtual IPL. Clean-up restores the user's page and frees the real page if it is not in the V=R machine. If the real page is in the V=R machine, the real page is not freed. The PSW from location zero of the virtual machine is loaded and made the current PSW.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'40':

Rx
Must contain a zero.

Ry
Must point to the virtual machine registers to be loaded.

Use: DIAGNOSE X'40' is used in DMKVMI to clean up after a virtual IPL.

Program Exceptions: If DIAGNOSE code X'40' is specified incorrectly, the following exception is generated:

Specification
If the DIAGNOSE is issued outside of its use in DMKVMI.

DIAGNOSE Code X'48' -- Issue SVC 76 from a Second Level VM/370 or VM Virtual Machine

All privilege classes (except ANY)

A second level VM/370, VM/SP, or VM/SP HPO operating system issues SVC 76 using this DIAGNOSE. SVC 76 handles I/O error recording for virtual operating systems. For instance, a virtual machine issues SVC 76 to record data about hardware errors that occur on devices dedicated to it.

Note: If you have VM/SP HPO, DIAGNOSE code X'48' is supported by a preferred machine assist guest with the control switch assist active.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'48':

Rx
Is R1, which must contain either of two values

X'04' indicates an SVC 76 request from a virtual machine
X'08' indicates that a virtual machine issued DIAGNOSE code X'48'.

Ry
Is not used in this DIAGNOSE.

Use: CP checks first for the X'04' value. If it is present, CP sets VMSPMFLG in the virtual machine's VMBLOK to X'04' and processes the SVC 76 request on behalf of the virtual machine.

If R1 contains a X'08' value, CP sets VMSPMFLG in the virtual machine's VMBLOK to X'08'. It then reflects the SVC 76 back to the virtual machine. The virtual machine then handles its own error recording.

For more information on SVC 76 and I/O error recording procedures, refer to *VM/SP OLTSEP and Error Recording Guide.*

DIAGNOSE Code X'4C' -- Generate Accounting Records for the Virtual User

All privilege classes (except ANY)

DIAGNOSE code X'4C' can be issued only by a user with the account option (ACCT) in his directory.

Note: If you have VM/SP HPO, DIAGNOSE code X'4C' is supported by a preferred machine assist guest with the control switch assist active.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'4C':

Rx
Contains the virtual address of either a 24-byte parameter list identifying the "charge to" user, or a variable length data area that is to be stored in the accounting record. The interpretation of the address is based on a hexadecimal subcode supplied in Ry. If the virtual address represents a parameter list, it must be doubleword aligned; if it represents a data area, the area must not cross a page boundary. If Rx is interpreted as pointing to a parameter list and the value in Rx is zero, the accounting record is spooled with the identification of the user issuing the DIAGNOSE instruction.

Ry
Contains a hexadecimal subcode interpreted by DIAGNOSE code X'4C' as follows:

Subcode	Rx points to:
0000	A parameter list containing only a userid.
0004	A parameter list containing a userid and account number.
0008	A parameter list containing a userid and distribution number.
000C	A parameter list containing a userid, account number, and distribution number.
0010	A data area containing up to 70 bytes of user information to be transferred to the accounting card starting in column 9.

Ry+1
Contains the length of the data area pointed to by Rx. If Rx points to a parameter list (Ry not equal to X'0010'), Ry+1 is ignored.

DIAGNOSE code X'4C' checks the VMACCOUN flag in VMPSTAT to verify that the user has the account option and if not, returns control to the user without generating an accounting record.

Subcode X'0000' -- Subcode X'000C'

DIAGNOSE code X'4C' verifies that the user has the account option, and if not, returns control to the user. If the user has the ACCT option specified, control is passed to DMKCPV to generate the record. DMKCPV passes control to DMKACO to complete the "charge to" information; either from the User Accounting Block (ACCTBLOK), if a pointer to it exists, or from the user's VMBLOK. DMKCPV passes control back to DIAGNOSE code X'4C' to release the storage for the ACCTBLOK, if one exists. DIAGNOSE code X'4C' then checks the parameter list address. If the parameter list address is zero, control is returned to the user with a condition code of zero.

Program Exceptions: If DIAGNOSE code X'4C' is specified incorrectly, the following exceptions are generated:

Addressing
If an invalid parameter list address is specified.

Specification
If the parameter list address is not aligned on a doubleword boundary.

Subcode X'0010'

If both the virtual address and the length are valid, DMKFREE is called to obtain storage for an account buffer (ACNTBLOK) which is then initialized to blanks. The userid of the user issuing the DIAGNOSE instruction is placed in columns 1 through 8 and an accounting record identification code of "C0" is placed in columns 79 and 80. The user data pointed to by the address in Rx is moved to the accounting record starting at column 9 for a length equal to the value in Ry+1. A call to DMKACOQU collects the accounting records on the system accounting chain (DMKRSPAC) and puts them in spool format. DIAGNOSE code X'4C' then returns control to the user with a condition code of zero.

Program Exceptions: If Ry contains a subcode of X'0010', and DIAGNOSE code X'4C' is specified incorrectly, the following exceptions are generated:

Addressing
If the address specified in Rx is negative or greater than the size of the user's virtual storage.

Specification
If the combination of the address in Rx and the length in Ry+1 indicates that the data area crosses a page boundary.

If the value in Ry+1 is zero, negative, or greater than 70.

Notes:

1. *For subcode X'0010', the only valid accounting record identification code (ACNTCODE field of the ACNTBLOK) is "C0". For the other four subcodes listed above, the accounting record identification code can be "C1", "C2", etc. For more information on accounting record identification codes, see VM/SP Data Areas and Control Block Logic Volume 1 (CP) or VM/SP HPO Data Areas and Control Block Logic - CP.*

2. *If Ry contains subcode X'0010', Ry cannot be register 15.*

Condition Codes: The condition codes returned are as follows:

Condition Codes	Meaning
CC=0	Both userid and function hexadecimal subcode are valid and an accounting record is generated. If subcodes X'0000' through X'000C' have been specified, the User Accounting Block (ACCTBLOK) is built and the userid, account number, and distribution number are moved to the block from the parameter list or the User Machine Block belonging to the userid in the parameter list.
CC=1	The ACCT option is not set in the directory entry. The user is not authorized to issue this DIAGNOSE.
CC=2	(Applies only to subcodes X'0000' through X'000C'). The userid provided in the parameter list is not found when checked against the directory list. The parameter list address must be nonzero and valid. No ACCTBLOK is built.
CC=3	(Applies only to subcodes X'0000' through X'000C'). The function hexadecimal subcode is invalid or an error occurs getting the user machine block (UMACBLOK). No ACCTBLOK is built.

DIAGNOSE Code X'50' -- Save the 370X Control Program Image

Privilege class A, B, or C

This section applies only to EP (Emulator Program) generations as defined, created, and loaded by VM.

DIAGNOSE code X'50' invokes the CP module DMKSNC to validate the parameter list and write the page-format image of the 370X control program to the appropriate system volume.

When a 370X control program load module is created, the CMS service program SAVENCP builds a communications controller list (CCPARM) of control information. It passes this information to CP via a DIAGNOSE code X'50'.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'50':

Rx
Contains the virtual address of the parameter list (CCPARM).

Ry
Is ignored on entry.

Exit Values: Upon return, Ry contains the following error codes:

Code	Meaning
044	'ncpname' was not found in system name table.
171	System volume specified not currently available.
178	Insufficient space reserved for program and system control information.
179	System volume specified is not a CP-owned volume.
435	Paging error while writing saved system.

DIAGNOSE Code X'54' -- Control The Function of the PA2 Function Key

All privilege classes (except ANY)

DIAGNOSE code X'54' controls the function of the PA2 function key. The PA2 function key can be used either to simulate an external interrupt to a virtual machine or to clear the output area of a display screen.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'54'.

Rx
Determines the function of the PA2 key. If Rx contains a nonzero value, the PA2 key simulates an external interrupt to the virtual machine.

If Rx contains a value of zero, the PA2 key clears the output area of the display screen.

The external interrupt is simulated only when the display screen is in the VM READ, HOLD, or MORE status and the TERMINAL APL ON command has been issued.

DIAGNOSE Code X'58' -- 3270 Virtual Console Interface

All privilege classes (except ANY)

DIAGNOSE code X'58' enables a virtual machine to communicate with 3270 display stations. Using DIAGNOSE code X'58', a virtual machine may:

- Display up to a full screen of data using only one write operation.
- Provide attribute characters along with data that is sent to the display station. An attribute character provides control information for the data, for example, a request to intensify the data when it is displayed.
- Place a 3270 display station under control of the virtual machine (full screen mode).

Note: If you want to run multiple applications in a virtual machine, use the CMS CONSOLE I/O macro. CONSOLE, a CMS interface to fullscreen I/O, provides improved usability for 3270 applications. Refer to the *VM/SP CMS Macros and Functions Reference* for the macro format for CONSOLE. *VM/SP CMS for System Programming* contains details for using CONSOLE.

Entry Values: When a virtual machine issues DIAGNOSE code X'58', the virtual machine must provide one or more channel command words (CCWs). These CCWs specify the 3270 operation to be performed, provide control information for the display station, and specify the address of data to be displayed during a write operation or the address of a buffer where data is to be stored during a read operation.

Set up the input registers as follows when invoking DIAGNOSE code X'58':

Rx
Must contain the CCW address, if only one CCW is used.

If CCWs are chained, Rx must contain the address of the first CCW in the chain.

Ry
Must contain the virtual address of the virtual console where the operation is to be performed. This value must be right-justified.

Displaying Data

To display up to a full screen of data, code a CCW using the following assembler language instructions:

```
DS 0D
DC AL1(CCWCODE),AL3(DATADDR),AL1(FLAGS),AL1(CTL),AL2(COUNT)
```

where:

CCWCODE is the command code X'19'.

DATADDR is the virtual storage address of the first byte of data to be displayed.

FLAGS are standard CCW flags. The suppress-incorrect-length indicator, bit 34, must be set to a value of one. Set other bits as needed.

CTL is a control byte defined as follows:

The high-order bit (0), if set on, enables the screen "MORE" status to be active before the displaying of data.

- Bits 2-7 identify the line on the display screen where the display is to start. A value of 0 (B'xx00 0000') corresponds to the first or top line, a value of 1 (B'xx00 0001') corresponds to the second line and so forth.

- If the control byte contains the value X'FF', CP erases the display station's screen. No new data is displayed.

- CCW's may be command chained to combine several operations in one DIAGNOSE. When CP builds the real CCW string, it will data chain as many CCW's as possible to reduce the number of real I/O operations. If the control byte contains a value of X'FE', CP will:

  - Not data chain this operation to any previous CCW in the real CCW string.

  - Erase the entire screen.

  - Rewrite the attribute bytes for the CP screen format.

  - Reset the cursor to the beginning of the input area.

COUNT specifies the number of bytes of data to be displayed. The maximum amount of data that can be displayed at one time depends upon the 3270 model of the display station:

- A model 2 can display up to 1760 bytes
- A model 3 can display up to 2400 bytes
- A model 4 can display up to 3280 bytes
- A model 5 can display up to 3300 bytes

To provide attribute characters for the data, place the attribute character in the data stream immediately following a 3270 start-field order. The start-field order, a one-byte value, notifies the 3270 display system that the next byte in the data stream is an attribute character. For a description of how the 3270 display system uses attribute characters, and to determine the values to specify for attribute characters and the start-field order, see the *IBM 3270 Information Display System Library User's Guide.*

Note: Through the use of the attribute character, it is possible to define a display field as selector-pen detectable. However, when the selector pen is used to select the field, CP does not return data from the field to the virtual machine.

Condition Codes: After processing DIAGNOSE code X'58', CP sets a condition code. If the operation was successful - that is, no I/O errors occurred - CP sets a condition code of zero. If an I/O error occurred, CP sets a condition code of one.

If an I/O error occurred, the application program can check the I/O status and the error type by:

- Issuing a TEST I/O (TIO) instruction
- Examining the returned condition code
- Examining the virtual CSW

The returned condition codes and CSW status are the standard condition codes and status defined in the *IBM System/370 Principles of Operation.*

You must also make sure that the interrupt for the virtual device is enabled by setting the appropriate bit and channel mask in the PSW. For example, if the virtual address of your console is 009, bit 0 in the channel mask must be set to one (that is, bit 0 must be on). This may be the case if you are loading programs in the transient area.

Full Screen Mode

DIAGNOSE code X'58' provides a means by which a virtual machine may share, with CP, control of a 3270 display station. Three CCW operations, X'29', X'2A', and X'49', in addition to performing the requested I/O, notify CP that the display station is operating under the control of the virtual machine.

CCW code X'29' performs a WRITE, ERASE/WRITE, ERASE/WRITE ALTERNATE, or WRITE STRUCTURED FIELD operation, depending on the value of the control field. For the WRITE, ERASE/WRITE, and ERASE/WRITE ALTERNATE, the virtual machine must provide appropriate control information beginning with the Write Control Character (WCC) and including 3270 orders following the WCC. Data may be written anywhere on the screen. The virtual machine must provide the address where the write is to begin; it uses a SET BUFFER ADDRESS (SBA) order to do this. Writing can also start at the current cursor address.

CCW code X'29' performs a WRITE STRUCTURED FIELD operation when the value of the control field is X'20'. The WRITE STRUCTURED FIELD instruction sends control information to a 3274 controller. The application program must provide the control information in the data stream in the format required by the instruction. (See the *3270 Component Description* for more information on WRITE STRUCTURED FIELD operation.)

CCW code X'2A' performs a READ BUFFER or a READ MODIFIED operation, depending on the value of the control field.

When the virtual machine issues CCW code X'49', CP treats the channel program as a normal 3215 channel program, with the following exceptions:

- The data stream is processed as if TERMINAL LINESIZE OFF has has been issued. That is, the data stream will be broken up only when a X'15' is encountered in the data stream.

- Each time a X'15' is encountered CP counts exactly one line.

- CP will not translate any code point from X'40' to X'FE', inclusive. CP also will not translate code points X'0E' and X'0F'.

CCW code X'49' is invoked the same as CCW code X'29' except that both the CCWs and data stream should be for a 3215 instead of a 3270.

Before issuing DIAGNOSE code X'58' code X'49', issue DIAGNOSE code X'8C' to determine the width of the screen. Then, use this information to determine the number of character positions that can be taken up between X'15's. The width of the screen minus 2 should be the maximum for that number. If this restriction is not observed, CP will not be able to manage the screen correctly.

To specify the full screen mode CCW, use the following assembler language instructions:

```
DS 0D
DC AL1(CCWCODE),AL3(DATADDR),AL1(FLAGS),AL1(CONTROL),AL2(COUNT)
```

where:

CCWCODE is a CCW code (X'29' or X'2A')

DATADDR for a write operation, specifies the first byte of the data stream (WCC) to be written. For a read operation, specifies the address of the read buffer.

FLAGS is the standard CCW flag field.

CONTROL for a write operation (CCW code of X'29') the following control field values cause the following operations to be performed:

Value	Operation Performed
X'80'	ERASE/WRITE
X'C0'	ERASE/WRITE ALTERNATE
X'40'	ERASE/WRITE ALTERNATE
X'20'	WRITE STRUCTURED FIELD
all other values	WRITE

For a read operation (CCW code of X'2A') the following control field values cause the following operations to be performed:

Value	Operation Performed
X'80'	READ MODIFIED
all other values	READ BUFFER

By adding X'10' to the CONTROL field values for ERASE/WRITE or ERASE/WRITE ALTERNATE, making them X'90' or X'D0' respectively, the PA1 key interrupt is reflected to the virtual machine. This replaces the normal PA1 key function of returning the virtual machine to CP mode, and allows a virtual machine to have full control of the keyboard.

Normal PA1 key function is restored when full screen mode is reset.

The control field has no meaning for CCW code X'49'.

COUNT for a write operation, specifies the number of bytes to be displayed plus the number of bytes of control information. For a read operation, specifies the number of display characters to be read plus the number of bytes of control information. The maximum number of bytes that can be specified is 65503. The maximum number of displayable positions for the supported devices is:

3277 and 3275 Model 2 - 1920 bytes
3278, 3276 and 3279 Model 2 - 1920 bytes
3278, 3276 and 3279 Model 3 - 2560 bytes
3278 and 3276 Model 4 - 3440 bytes
3278 Model 5 - 3564 bytes
3290 - 9920 bytes

Valid Channel Programs

The channel program that is the input to DIAGNOSE code X'58' can be composed of both full screen diagnose channel commands (X'29', X'2A') and 3215 channel commands. The channel program must follow these rules:

1. The first CCW must be a full screen operation.
2. Subsequent CCWs in the channel program can be either full screen operations, 3215 operations, TRANSFER IN CHANNELs or NO-OPs.
3. Command chaining is allowed except when a CCW is chained from a WRITE STRUCTURED FIELD CCW.
4. The total buffer length for data chained CCWs is 65,504 bytes.
5. A 3215 operation within a channel program resets full screen mode. The next full screen operation must be an ERASE/WRITE operation.
6. WRITE STRUCTURED FIELD operations are rejected if the display does not support extended data characteristics.

Full Screen Interactions

The virtual machine console exists in either of two modes, CP mode and full screen mode. CP mode is the default screen mode and is indicated by the screen status field in the lower right-hand corner of the screen. When in CP mode, the screen format is controlled by CP, and the data that appears on the screen is provided by CP and the programs running in the virtual machine. Full screen mode is initiated by the application program running in the virtual machine. When in full screen mode, the screen format and data are under complete control of the program running in the virtual machine.

If TERMINAL BREAKIN GUESTCTL is specified, the screen mode changes only when the break-in key is used. An audible alarm is sounded when CP messages are queued. Priority CP messages and DIAGNOSE code X'08' output take over the full screen.

CP mode is terminated and full screen mode is initiated when the application program issues an ERASE/WRITE instruction. Full screen mode may be terminated by a CP mode type I/O to the screen any time the keyboard is in a locked state.

Interactions between CP and the application program in the virtual machine using full screen support are listed below. The application programmer must be familiar with the operation of the IBM 3270 display station. For detailed information on its operation, see the appropriate 3270 Information Display System Description and Programmer's Guide listed in the Bibliography. Also listed below are general programming considerations that must be followed to effectively use the DIAGNOSE code X'58' instruction for full screen I/O.

1. A full screen ERASE/WRITE or ERASE/WRITE ALTERNATE operation establishes full screen mode.

2. The application program is responsible for all I/O status and error checking, just as if START I/O (SIO) were being used instead of DIAGNOSE. This is done by using the TEST I/O (TIO) instruction and examining the returned condition code, and by examining the virtual CSW. The returned condition codes and CSW status are the standard condition codes and status as defined in the *IBM System/370 Principles of Operation,* with one exception noted below in number 5.

3. When in full screen mode, all CP messages are queued. The entire queue of CP messages is processed after each of the following operations:

   a. A full screen READ operation (any READ operation that locks the keyboard).

   b. A full screen WRITE operation that does not place the keyboard in the active status.

   c. The expiration of a 60-second timer for CP priority messages.

4. If a priority CP message (such as a warning message from the system operator) is to be displayed while in full screen mode, an attention interruption is posted to the application program and a 60-second timer is set. This informs the application program that a READ operation should be initiated. If a READ is not issued before the 60 seconds have expired, CP erases the screen and displays all queued messages.

   The only exception is if the application program has issued a Full Screen Support WRITE STRUCTURED FIELD instruction. CP does not take over the screen if the user has issued a WRITE STRUCTURED

FIELD. This exception does not apply to terminals controlled by VM/VTAM.

5. When a mode switch has occurred and the screen is in CP mode, the application program is notified by an X'8E' in the CSW unit status byte following a full screen I/O operation. An ERASE/WRITE or ERASE/WRITE ALTERNATE instruction should be issued to reestablish full screen mode and reformat the screen. If control of the PA1 key interrupt had been transferred to the virtual machine via the CONTROL option, it must be specified again to return PA1 key control back to the virtual machine. Otherwise, pressing the PA1 key places the display in CP mode.

   An X'8E' in the CSW unit status byte following an ERASE/WRITE or ERASE/WRITE ALTERNATE instruction indicates that non-full screen data (CP mode) is waiting to be read. The application program should issue a non-full screen READ and then reissue the ERASE/WRITE instruction.

6. Other non-full screen virtual machine messages are displayed immediately when in full screen mode.

7. The application program must establish an environment to handle attention interruptions. This could be done using the CMS macros HNDINT and WAITD. There are two conditions when CP posts an attention interruption to the application program:

   a. When CP receives an attention interruption indicating that the virtual machine console operator has caused an interruption. (For example, the operator pressed ENTER or a PF key on the display keyboard).

   b. When a CP priority message is to be displayed.

   In either case, the application program should respond by issuing a READ.

8. The application program must also establish an environment to handle I/O interruptions and must ensure that channel end and device end have been received before processing continues.

9. If the test request key is depressed from a local 3270 when in full screen mode, X'604040' is returned to the application program in the read buffer. The test request key is not supported for remote 3270 terminals.

10. If you press the PA1 key in full screen mode, CP posts an attention interrupt to your virtual machine. If the virtual machine does not respond with a READ and you press the PA1 key a second time, your virtual machine is put in CP mode and "CP READ" is displayed in the screen's status area. However, if you set bit X'10' of the CONTROL option on before the initial ERASE/WRITE or ERASE/WRITE ALTERNATE, and press the PA1 key, the interrupt is reflected to your virtual machine for handling. If you have not set bit X'10' of the

CONTROL option on and you press the PA1 key, your virtual machine is put in CP mode and "CP READ" is displayed in the screen's status area.

11. The application programmer must be aware that long data streams may result in very high CP storage use and possible system degradation. In addition, long data streams sent over BSC lines may cause degradation of response time on other terminals on the same BSC line.

Full Screen Interactions (3270 SIO)

Full screen console (3270 SIO) support enables a guest virtual machine and CP to share a locally attached display terminal controlled by CP. The virtual machine can use the display terminal as a graphics device in full screen mode; CP can use the same terminal as a line device. When the terminal is in full screen mode, the screen format, data checking, and error checking are under the complete control of the application program running in the virtual machine. A guest virtual machine can use either DIAGNOSE code X'58' or the SIO instruction to initiate full screen mode, but not both.

Before the guest virtual machine can issue 3270 SIO commands, it must first issue the CP TERMINAL command with the CONMODE 3270 option to be able to issue 3270 SIO commands. In addition, the SCRNSAVE ON option of the CP TERMINAL command gives a virtual machine (that has also specified CONMODE 3270) the ability to save the full screen display when the screen enters CP mode. If SCRNSAVE ON is specified, the screen is automatically displayed again when the console returns to full screen mode. If SCRNSAVE OFF has been specified by a virtual machine that has specified CONMODE 3270 and CP takes over a screen, CP presents a CLEAR attention interrupt to the virtual machine when CP is ready to give up control of the screen. It is the responsibility of the application program to issue an ERASE/WRITE to refresh the screen. If the virtual machine issues only a WRITE that does not cover the entire screen, information that CP displayed can remain on the screen. To use the CP TERMINAL SCRNSAVE OFF:

1. Always issue a WRITE after a READ.

2. CP can break into a CCW chain containing WRITEs (with the WCC byte making the keyboard locked) and take over the screen. Upon return to full screen mode, the next CCW in the chain is processed as if it is the first CCW. The guest system must provide a means to handle this situation.

3. Refresh the screen with an ERASE/WRITE when CP issues a CLEAR attention interrupt.

4. When ATTENTION from the console is received, the guest program must issue a READ.

The TERMINAL BREAKIN GUESTCTL option allows a guest operator to control break-ins (when CP takes over the full screen). Each time a CP request is received, it is put on a defer queue and an audible alarm sounds. The guest operator can switch to CP mode by hitting the break-in key.

The TERMINAL BRKKEY option allows the user to specify a PF key as the break-in key in full screen mode. The default break-in key is PA1. PA1 attentions are sent to the virtual machine when PA1 is not defined as the BRKKEY. Some applications may interpret this PA1 attention as a user request to enter the CP environment. TERMINAL BRKKEY NONE disables the break-in key. Local 3270 terminal users can set the break-in key to PA1, any PF key, or to NONE. Remote and VM/VTAM terminal users can set the break-in key to PA1 or NONE.

Notes:

1. *DIAGNOSE code X'58' may be issued from the CONSOLE service. The CONSOLE macro accesses CMS full-screen console services and performs 3270 I/O operations.*

2. *DIAGNOSE code X'58' is a 3215 command and causes command rejects if executed with CONMODE 3270.*

3. *When invoking CCW code X'49' both APL and TEXT must be off. Having either APL or TEXT on causes command rejects.*

4. *Issuing CCW code X'49' from a device other than a 3270 causes command rejects.*

5. *DIAGNOSE code X'58' can be used with BREAKIN and BRKKEY.*

6. *CONMODE must be 3215 to run CMS. If CMS sets CONMODE to 3270 while CMS is running, results are unpredictable.*

7. *SCRNSAVE ON must be specified if running a guest SCP such as MVS with CONMODE 3270. Otherwise results are unpredictable. The SCRNSAVE option of the TERMINAL command is not supported for VM/VTAM and remote 3270 terminals.*

8. *Since the only devices known to a virtual machine appear to the virtual machine as local, the CCW strings built for DIAGNOSE code X'58' should be constructed for local devices.*

9. *When IPLing the loader from your virtual machine to create a CP nucleus, CONMODE should be set to 3215 mode. Otherwise, console messages generated by the loading process are not displayed at the terminal.*

10. *CONMODE 3270 is not supported for disconnected users.*

DIAGNOSE Code X'5C' -- Error Message Editing

All privilege classes (except ANY)

DIAGNOSE code X'5C' causes the editing of an error message according to the user's setting of the EMSG function.

A message consists of three parts:

- The message identifier
- A separator character, which is usually a blank
- The message text.

If you are unfamiliar with the format of a message, please see *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes.*

A user may set EMSG to receive error messages in specific formats.

SET EMSG ON	Returns the entire error message; the message identifier, the separator character, and the message text.
SET EMSG CODE	Returns the message identifier only; no separator character, or message text.
SET EMSG TEXT	Returns the message text only; no separator character, or message identifier.
SET EMSG OFF	Returns no message at all.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'5C':

Rx
Contains the address of the message to be edited.

Rx+1
Contains the (user-specified) length of the message identifier. Rx+1 is not used unless subcode X'40' is specified in the high-order byte of Ry. The specified length must be greater than or equal to 0.

Ry
Contains a one-byte subcode in the high-order byte; the other three bytes specify the length of the message to be edited.

Set the subcode in the high-order byte of Ry as follows:

X'00' For the default identifier length of 10

X'40' For a user-specified identifier length.

Specifying subcodes other than those listed above result in setting Ry to 0.

Exit Values: The values returned in Rx and Ry depend on the setting of the EMSG function and the subcode in Ry. Rx contains the address of the message that should be issued. Ry contains the length of the message to be issued. Rx+1 remains unchanged. Rx and Ry are modified as follows:

SET EMSG	Subcode	Rx+1	Rx	Ry
ON	X'00'	not used	Rx	Ry
CODE	X'00'	not used	Rx	10 (Length of identifier)
TEXT	X'00'	not used	Rx + 11 (pointer to text part of message)	Ry - 11 (length of text alone)
OFF	X'00'	not used	Rx	0
ON	X'40'	Rx+1	Rx	Ry
CODE	X'40'	Rx+1	Rx	Rx+1
TEXT	X'40'	Rx+1	Rx + (Rx+1 + 1) (pointer to text part of message)	Ry - (Rx+1 + 1) (length of text alone)
OFF	X'40'	Rx+1	Rx	0

Note: DIAGNOSE code X'5C' does not write the message; it merely rearranges the starting pointer and length. For CMS error messages, a console write is performed following the DIAGNOSE unless Ry is returned with a value of 0.

Upon completing DIAGNOSE code X'5C', check Ry. If it is 0, then this message should not be issued. Ry is set to 0 for the following conditions:

- SET EMSG ON if initially Ry is 0
- SET EMSG CODE if initially Rx+1 is 0
- SET EMSG TEXT if initially Ry is less than or equal to (Rx+1) + 1
- SET EMSG OFF
- If Rx+1 is less than 0
- If a subcode other than X'00' or X'40' is specified.

Program Exceptions: If DIAGNOSE code X'5C' is specified incorrectly, the following exception is received:

Specification
If Rx is specified as R15 and the subcode in Ry is X'40'.

DIAGNOSE Code X'60' -- Determining the Virtual Machine Storage Size

All privilege classes (except ANY)

DIAGNOSE code X'60' allows a virtual machine to determine its size.

Exit Values: On return, Rx contains the virtual machine storage size.

DIAGNOSE Code X'64' -- Finding, Loading, and Purging a Named Segment

All privilege classes (except ANY)

DIAGNOSE code X'64' controls the linkage of discontiguous saved segments.

Entry Values: The type of linkage that is performed depends upon the function subcode in Ry.

Set up the input registers as follows when invoking DIAGNOSE code X'64':

Rx
Must contain the address of the name of the segment. The segment name must be 8 bytes long, on a doubleword boundary, left justified, and padded with blanks.

Ry
Contains one of the function subcode listed below.

Subcode	Function
X'0000'	LOADSYS -- Loads a named segment in shared mode
X'0004'	LOADSYS -- Loads a named segment in nonshared mode
X'0008'	PURGESYS -- Releases the named segment from virtual storage
X'000C'	FINDSYS -- Finds the starting and ending address of the named segment

Subcodes X'0000' and X'0004' -- The LOADSYS Function

When the LOADSYS function is executed, CP finds the system name table entry for the segment and builds the necessary page and swap tables (two sets one for each processor, when running in attached processor mode). CP releases all the virtual pages of storage that are to contain the named segment and then loads the segment in those virtual pages. When the LOADSYS function is executed, CP expands the virtual machine size dynamically, if necessary. CP also expands the segment tables to match any expansion of virtual storage.

Note: If the named saved system is designated as Virtual Machine Group via the VMGROUP=YES option on the NAMESYS macro, it cannot be loaded using the LOADSYS function.

When LOADSYS executes successfully, the address of where the named segment was loaded is returned in Rx. When the LOADSYS function loads a segment in shared mode, it resets instruction and branch tracing if either was active.

After a LOADSYS function executes, the storage occupied by the named segment is addressable by the virtual machine, even if that storage is beyond the storage defined for the virtual machine. However, any storage beyond that defined for the virtual machine and below that defined for the named segment is not addressable. Figure 3 shows the virtual storage that is addressable before and after the LOADSYS function executes.

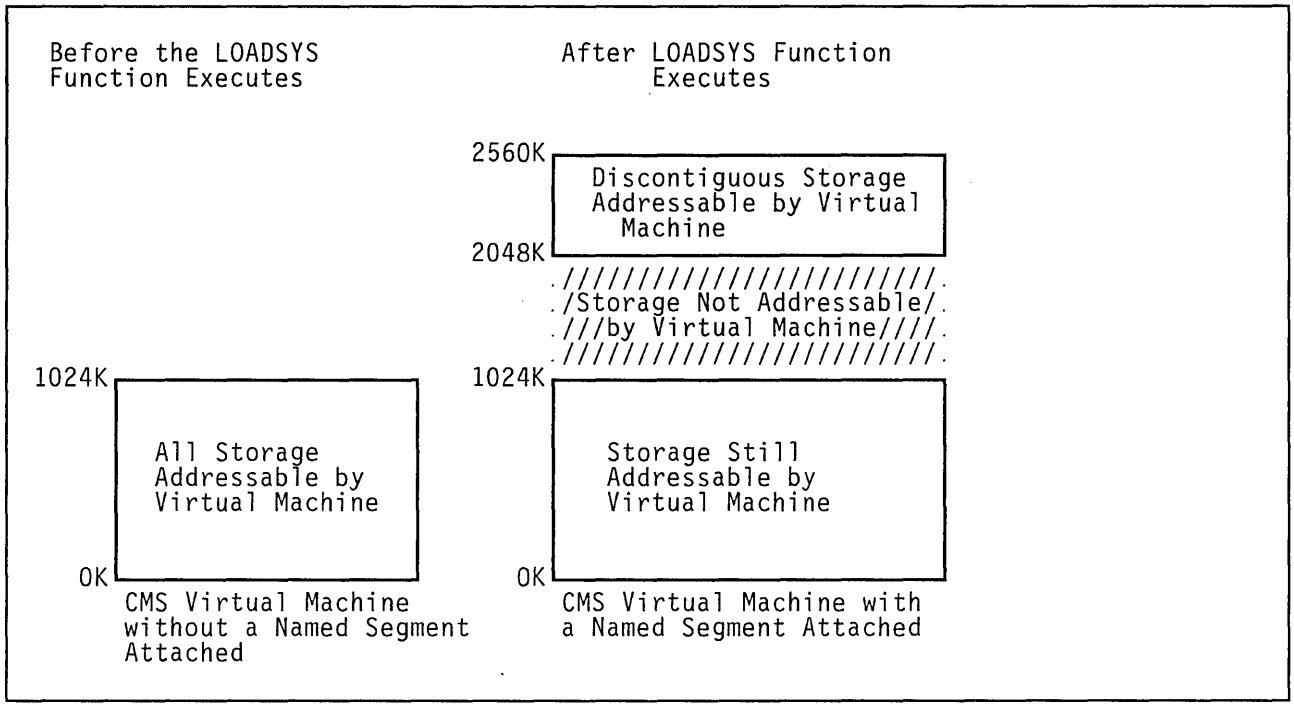


Figure 3. Addressable Storage Before and After a LOADSYS Function

When you save a named segment that is later loaded by the LOADSYS function, you must be sure that the addresses at which segments are saved are correct and that they do not overlay required areas of storage in the virtual machine. This is crucial because the LOADSYS function invokes the PURGESYS function before it builds the new page and swap tables. CP purges all saved systems that are overlayed in any way by the saved system it is loading.

Condition and Return Codes: A condition code of 0 in the PSW indicates that the named segment was loaded successfully; Rx contains the load address.

A condition code of 1 in the PSW indicates the named segment was loaded successfully within the defined storage of the virtual machine. Rx contains the address at which the named segment was loaded. Ry contains the ending address of the storage released before the named segment was loaded.

Note: CMS only allows named segments to be attached beyond the defined size of the virtual machine.

A condition code of 2 in the PSW indicates the LOADSYS function did not execute successfully. Examine the return code in Ry to determine the cause of the error.

Return Code	Meaning
1	Named segment defined as a VMGROUP
44	Named segment does not exist
174	Paging I/O errors
179	The DASD volume specified by "SYSVOL" in the NAMESYS macro is not a CP-owned volume.
203	User in V=R area

Subcode X'0008' -- The PURGESYS Function

When the PURGESYS function is executed; CP releases the storage, and associated page and swap tables, that were acquired when the corresponding LOADSYS function was executed. If the storage occupied by the named segment was beyond the defined virtual machine storage size, that storage is no longer addressable by the virtual machine.

When a PURGESYS function is executed for a segment that was loaded in nonshared mode, the storage area is cleared to binary zeroes and the keys are reset to zeroes. If PURGESYS is invoked for a named segment that was not previously loaded via LOADSYS, the request is ignored.

Condition and Return Codes: A condition code of 0 in the PSW indicates successful completion.

A condition code of 1 in the PSW indicates that the named segment was not found in the virtual machine.

A condition code of 2 in the PSW and a return code of 44 in Ry indicate that the named segment either does not exist or was not previously loaded via the LOADSYS function.

Subcode X'000C' -- The FINDSYS Function

When the FINDSYS function is executed, CP checks that the named segment exists and that it has not been loaded previously. If the named saved segment is designated as Virtual Machine Group, the FINDSYS function cannot be used.

Condition and Return Codes: A condition code of 0 in the PSW indicates that the named segment is already loaded. The address at which it was loaded is returned in Rx and its highest address is returned in Ry.

A condition code of 1 in the PSW indicates that the named segment exists but has not been loaded. In this case, the address at which the named segment is to be loaded is returned in Rx and the highest address of the named segment is returned in Ry.

A condition code of 2 in the PSW indicates the FINDSYS function did not execute successfully. Examine the return code in Ry to determine the error that occurred.

Return Code	Meaning
1	Named segment defined as a VMGROUP
44	Named segment does not exist
174	Paging I/O errors
203	User in V=R area

DIAGNOSE Code X'68' -- Virtual Machine Communication Facility (VMCF)

All privilege classes (except ANY)

DIAGNOSE code X'68' is used by a virtual machine to initiate a function of the Virtual Machine Communication Facility (VMCF).

Note: If you have VM/SP HPO, DIAGNOSE code X'68' is supported by a preferred machine assist guest with the control switch assist active.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'68':

Rx

Contains the virtual address of a parameter list (VMCPARM). The address of VMCPARM is doubleword aligned. One of the entries in this parameter list is a function subcode, specifying the particular request being initiated. The functions and their subcodes are as follows:

Subcode	Function
X'0000'	AUTHORIZE
X'0001'	UNAUTHORIZE
X'0002'	SEND
X'0003'	SEND/RECV
X'0004'	SENDX
X'0005'	RECEIVE
X'0006'	CANCEL
X'0007'	REPLY
X'0008'	QUIESCE
X'0009'	RESUME
X'000A'	IDENTIFY
X'000B'	REJECT

A description of all the fields of the VMCPARM is contained in Chapter 14, "The Virtual Machine Communication Facility" on page 283.

Return Codes: Ry contains the return code upon completion of DIAGNOSE code X'68' or the detection of an error condition.

Return Code	Meaning
0	The normal response, successful completion
1	Invalid virtual buffer address or length
2	Invalid function subcode
3	Protocol violation
4	Source virtual machine not authorized
5	User not available
6	Protection violation
7	SENDX data too large
8	Duplicate message
9	Target virtual machine in QUIESCE status
10	Message limit exceeded
11	REPLY canceled
12	Message not found
13	Synchronization error
14	CANCEL too late
15	Paging I/O error
16	Incorrect length
17	Destructive overlap
18	User not authorized for PRIORITY messages
19	Data transfer error
20	CANCEL - busy

For more detail of the return codes, see Chapter 14, "The Virtual Machine Communication Facility" on page 283.

Note: Rx and Ry can be any general purpose register, R0 through R15. They may also be the same register.

DIAGNOSE Code X'6C' -- Shadow Table Maintenance

All privilege classes (except ANY)

DIAGNOSE code X'6C' is an internal DIAGNOSE instruction issued by MVS to VM that is used only to pass the virtual address of a page table entry that maps to real page zero for the low storage protection facility.

Note: If you have VM/SP HPO, for a preferred machine assist guest, with control switch assist, issuing DIAGNOSE code X'6C' is a NOP because page 0 for a preferred machine assist guest with control switch assist is real page 0.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'6C':

Rx
Contains the virtual address of the a page table entry. The virtual address is stored in the EXTVPORL field of the ECBLOK. The VMVPOREL flag in the VMBLOK is set on.

Ry
Is not used.

Condition Codes: If a guest virtual machine (without EC mode capability, EC mode is set off), tries to issue a DIAGNOSE code X'6C' a condition code of 3 will be returned.

DIAGNOSE Code X'70' -- Activating the Time-of-Day (TOD) Clock Accounting Interface

All privilege classes (except ANY)

DIAGNOSE code X'70' enables an operating system that is running in a virtual machine to request timing information from CP. Each time the virtual machine is dispatched, CP provides the accumulated processor time the virtual machine has used and the time of day the virtual machine was dispatched. Programs that are running in the virtual machine may use the timing information to calculate the amount of processor time used by each job, by each job step, and so forth.

DIAGNOSE code X'70' should be used by operating systems that use the store clock (STCK) instruction to obtain the time of day to calculate processor use. Because there is no virtual TOD clock, calculations that use multiple STCK instructions may not reflect the time used by just one

virtual machine. They may also include the time used by all virtual machines and by CP.

A virtual machine should issue DIAGNOSE code X'70' only one time. Once issued, it is effective until the virtual machine is reset.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'70':

Rx
Must contain the address of a 16-byte area, the communication area.

Ry
Is not used.

The communication area must be aligned on a doubleword boundary and must be in the virtual machine's real storage, preferably in page zero. Page zero is preferred because CP always locks page zero and must also lock the page that contains the communication area. Thus, when page zero is used, CP does not have to lock an additional page.

Use: After DIAGNOSE code X'70' is issued, CP updates the communication area each time the virtual machine is dispatched. The first 8 bytes of the communication area contain the total processor time the virtual machine has used. The last 8 bytes contain the time of day CP last dispatched the virtual machine. Programs running in the virtual machine should not alter the communication area.

To use the information that CP has stored in the communication area, perform the following steps:

1. Obtain the current time of day by issuing the STCK instruction.

2. Compute the difference between the tome of day obtained in step 1 and the time of day stored in the communication area. This difference is the amount of processor time the virtual machine has used since it was last dispatched.

3. To calculate the total amount of processor time the virtual machine has used up to the present time, add the processor time that is stored in the communication area to the difference obtained in step 2.

4. Ensure that the TOD value stored in the communication area has not changed since step 2 was performed. If it has changed, repeat the procedure from step 1.

Program Exceptions: If DIAGNOSE code X'70' is specified incorrectly, the following exceptions are received:

Specification
If the virtual machine does not have the ECMODE option.

If the communication area is not aligned on a doubleword boundary.

If the address in Rx is not within the virtual machine's real address range.

If DIAGNOSE code X'70' has already been issued for the virtual machine.

DIAGNOSE code X'74' -- Saving or Loading a 3800 Named System

Privilege class A, B, or C

DIAGNOSE code X'74' saves an image library as a 3800 named system or loads a named system into virtual storage when that named system is required by the 3800 printer.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'74':

Rx and Rx+1
Must contain the eight-character name of the system to be saved or loaded, left-justified and padded with blanks.

Ry
Must contain the virtual address at which to start saving or loading the named system. Ry must start on a page boundary.

Ry+1
Must contain a X'00' in the high order byte if a LOAD operation is required, and a X'04' for a SAVE operation. The remainder of Ry must contain the number of bytes to be saved or loaded into virtual storage. CP rounds the byte count up to the nearest whole page before the pages are saved or loaded. Partial pages are not saved or loaded.

Program Exceptions: If DIAGNOSE code X'74' is specified incorrectly, the following exceptions are received:

Addressing
If the area to be saved or loaded extends beyond the user's virtual storage.

Privileged operation
If the user does not have privileged class A, B, or C.

Specification
If Register 15 is specified in either Rx or Ry.

If the virtual address specified in Ry is not on a page boundary.

These exceptions cause abnormal termination (abend) and the user is notified.

Return Codes: When DIAGNOSE code X'74' processing completes, one of the following return codes is placed into Ry and returned to CP:

Return Code	Meaning
X'00'	Load/save successfully performed
X'04'	Named system not found
X'08'	Named system currently active
X'0C'	Volid for system not CP owned
X'10'	Volid for system not mounted
X'14'	Too many bytes to load/save; residual byte count is in Ry+1
X'18'	Paging error during load/save
X'1C'	Too few bytes to LOAD/SAVE. Needed byte count is in Ry+1.

DIAGNOSE Code X'78' -- MSS Communication

All privilege classes (except ANY)

DIAGNOSE code X'78' is used to communicate with the VM control program about MSS volume mounts and demounts.

Note: If you have VM/SP HPO, DIAGNOSE code X'78' is supported by a preferred machine assist guest with the control switch assist active.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'78':

Ry
Contains a function subcode. The valid subcodes and their meanings are:

Subcode	Meaning
X'00'	The virtual machine issuing the DIAGNOSE instruction is running OS/VS with MSS support and the DMKMSS program for MSS communication. Rx contains the device address of the virtual machine's MSS communicator virtual device.
X'04'	The virtual machine is ready to process an MSS request. The MSSCOM block representing the request should be placed at the virtual machine address indicated by Rx.
X'08'	An MSS request represented by the MSSCOM block located at the virtual machine address indicated by Rx has been accepted by the MSC.

X'0C' An MSS request represented by the MSSCOM block located at the virtual machine address indicated by Rx has been rejected by the MSC.

X'10' The DMKMSS program is no longer available to process MSS requests.

X'14' The DMKMSS program has created a list of all VUAs associated with this processor (cpuid) and requests CP to build its shared and non-shared SDG tables from that list.

Program Exceptions: If DIAGNOSE code X'78' is specified incorrectly, CP terminates the user program with one of the following exceptions:

Protection
If no DMKSSV module exists.

Specification
If MSSCOM crosses a page boundary.

Condition and Return Codes: DIAGNOSE code X'78' condition codes and return codes are:

Condition Codes	Meaning
0	Successful completion.
1	Error Condition. Register 15 contains one of the following return codes:

RC	Meaning
4	Subcode was either less than zero or greater than 16.
8	Subcode was within the valid range, but not a multiple of 4.
12	Addressing exception trying to bring in the buffer page.
16	Issuer is not the issuer of subcode zero.

DIAGNOSE Code X'7C' -- Logical Device Support Facility

All privilege classes (except ANY)

DIAGNOSE code X'7C' allows an application running in a virtual machine to drive a logical 3270 as if it were a real display station locally attached to the VM system. Communication between the application and the logical device is done via the DIAGNOSE interface and a new external interrupt code.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'7C':

Rx
Is a user-specified register (*not* GPR15) containing the logical device number that is used to coordinate CP and local system operations. For the INITIATE function, Rx may contain a specific address in the low-order two bytes. The address must be X'0000' to X'0FFF'.

Rx+1
For the INITIATE function, contains the following information about the pseudo device to be created:

- The first byte indicates optional features:

  Bit 0 - 3270 extended features to be supported
  Bit 1 - ACCEPT function must be followed by the STATUS function (must be on for logical printers)
  Bit 2 - specific device address is requested and specified in Rx.

- The second byte contains the model number. For the 3279 display and for the 3284, 6, 7, 8, and 9 printers, the last digit is included in the first four bits of the model byte.

- The third byte indicates the device class, and must be a X'40'.

- The fourth byte is the device type:

  X'01' = 3278 or 3279
  X'02' = 328x
  X'04' = 3277

Therefore, the following are acceptable in Rx+1:

X'xxyy4004'
will give a 3277 Model 2 regardless of yy

where xx = X'00'

Note: X' 80' is not accepted because 3277s do not have extended features

X'xxyy4002'
will give a logical printer (xx must include X'40')

where yy =

X'00' = 3286
X'40' = 3284
X'60' = 3286
X'70' = 3287
X'80' = 3288
X'90' = 3289
X'94' = 3289 Model 4

X'xxyy4001'
will give a 3278 or 3279

where xx =

X'00' for no extended features
X'80' for extended features

where yy =

X'02' = 3278 Model 2
X'03' = 3278 Model 3
X'04' = 3278 Model 4
X'05' = 3278 Model 5
X'92' = 3279 Model 2
X'93' = 3279 Model 3

For the ACCEPT function, Rx+1 contains the address of a data buffer.

For the PRESENT function, Rx+1 contains either an address or a complemented address. If an address, it is the address of a single buffer of data 4096 bytes or less in length. If a complemented address, it is the address of a list that describes a data stream occupying multiple data buffers and/or greater than 4096 bytes in length.

For the PRESENT function to a printer, if Rx+1 is zero, a device end interrupt is reflected.

For the STATUS function, Rx+1 contains the sense data in the low order byte.

Ry
Is a user-specified register (*not GPR 15*) containing the subcode of the logical device function to be executed:

Subcode	Function
X'0001'	INITIATE
X'0002'	ACCEPT
X'0003'	PRESENT
X'0004'	TERMINATE
X'0005'	TERMINATE (all)
X'0006'	STATUS

Ry+1
Contains, for the ACCEPT and PRESENT functions, the length of the data buffer when Rx+1 specifies a buffer address.

Exit Values: On completion of an ACCEPT function, Ry contains the length of the data.

On completion of any DIAGNOSE operation, Ry+1 contains the return code.

Condition and Return Codes: Return codes are received from this facility in Ry+1. PSW condition codes and return codes are described below. Functions that apply specifically to given combinations of condition and return codes are shown in parentheses.

Condition Codes **Meaning**

0 Function completed with no errors.

RC	Meaning
0	(Any function) Normal completion.
1	(ACCEPT) Indicates another ACCEPT required for another data stream.
2	(ACCEPT) Indicates another ACCEPT required for next segment of current data stream.
3	(PRESENT) Indicates that an external interrupt is presented when the data is finished.
4	(ACCEPT) Indicates that the STATUS function must be issued to end the ACCEPT.
5	(ACCEPT or STATUS) Indicates that a READ-CCW was chained to the WRITE-CCW.

1 Error condition.

RC	Meaning
1	(Any function) Invalid function used in Ry.
2	(ACCEPT) No data available.
3	(ACCEPT) Buffer too short. No data transferred. Another ACCEPT is required to retrieve the data. Ry contains the required data length unless data chained CCWs are being used. If data chained CCWs are being used, the byte count returned in Ry only contains the amount of data up to the CCW that exhausted the ACCEPT buffer and may not be the total data amount.
4	(ACCEPT or PRESENT) One of the following: • Buffer is greater than 4096 bytes. • Buffer length is not positive. • Buffer not in user's address space. • Paging I/O error.
9	(INITIATE) Maximum of 512 logical devices per virtual machine reached, or maximum of 4096 logical devices (for the entire system) reached.
10	(ACCEPT or PRESENT) FETCH or STORE protection violation.

2 Busy condition

RC	Meaning
1	(PRESENT) CP has pending data that must be accepted first. The PRESENT is not performed.

2 (PRESENT) A previous PRESENT has not completed execution. The current PRESENT is not performed. An external interrupt is issued to indicate when this PRESENT should be reissued.
3 (PRESENT) CP has an active READ BUFFER command. The PRESENT issued is for READ MODIFIED data.
4 (PRESENT) The data presented is from a READ BUFFER. No CP READ is outstanding, or the READ is a READ MODIFIED.
5 (PRESENT) CP is waiting for STATUS to be returned for a WRITE.

3 One of the following conditions:

- (INITIATE) Logical device type, class, or model is invalid.
- (Other functions) Logical device number in Rx is invalid.

RC Meaning
1 (ACCEPT, PRESENT, TERMINATE) CP is in the process of terminating the logical device.
2 (ACCEPT, PRESENT, TERMINATE) The logical device number does not exist.
3 (INITIATE) The logical device type or model is invalid, or bit 1 of register Rx+1 was not specified when initiating a printer.
4 (INITIATE) Specific device address in invalid.
5 (INITIATE) The requested device is already created.
6 (INITIATE) Paging I/O error.

Description of Logical Device Support Facility Functions

Logical device functions manage communications and the transfer of data between CP and the virtual machine for which the logical device was created.

INITIATE: DIAGNOSE CODE X'7C' SUBCODE X'0001'

The INITIATE function opens a logical communications path between the calling virtual machine issuing the DIAGNOSE and the VM Control Program. It causes a logical device to be created and the Virtual Machine/System Product logo to be directed to it. This results in an external interrupt to the issuing virtual machine to indicate that CP has data to be processed.

Rx+1 must contain the model number in byte 1, and the device class and type in bytes 2 and 3. Rx is not used for input.

The address of the logical device is placed in Rx. This value is used on subsequent DIAGNOSE operations to indicate the logical device being used. This address is also provided with the external interrupt so that the issuing virtual machine can associate the interrupt with a specific logical device.

ACCEPT: DIAGNOSE CODE X'7C' SUBCODE X'0002'

The ACCEPT function reads data that CP has directed to a logical device. It is invoked after the virtual machine that created the logical device is notified via external interrupt that output data is to be processed. Upon invocation, Rx+1 must contain the data buffer address and Ry+1 the buffer length. If the data buffer supplied was too short to contain the data, the function returns the required buffer size in Ry and no data is moved. This action can be overridden by setting an indicator in the length register (bit zero in Ry+1 set to 1) when the function is invoked. In this case, the data *is* be moved to the short buffer and a CC=0, RC=2 is sent. The system moves the next portion of the data on the next ACCEPT. Upon successful completion of function processing, the data length is returned in Ry, and the data buffer contains the CCW OP code in its first byte and data in the remaining buffer space.

PRESENT: DIAGNOSE CODE X'7C' SUBCODE X'0003'

The PRESENT function passes input data to CP. The location of the data is described by an address or a complemented address in Rx+1. If the register contains an address, it is the address of a data buffer 4096 bytes or less in length. In this case, Ry+1 contains the length of that data buffer. If Rx+1 contains a complemented address, it is the address of a list that describes a data stream occupying multiple data buffers and/or greater than 4096 bytes in length. In this case, Ry+1 is not used to describe the data length. However, in either case, a high-order bit of 1 in Ry+1 indicates that the response is to a READ BUFFER command. Data format is the same as that produced by a local display control unit in response to a READ MODIFIED channel command.

If a list is used to describe the data, the list must be in the format:

Length SEG1	Address SEG1
Length SEG2	Address SEG2
=	= =
Length SEGn*	Address SEGn

*Last entry indicated by a 1 in bit zero of its length field.

The list must start on a fullword boundary. Each entry consists of two fullword fields that describe the length and location of sequential segments of a data stream. A single entry list may be used to describe a single data buffer greater than 4096 bytes in length. Neither the list nor the data may be modified before transfer of the data has completed. An external interrupt signals completion of data transfer.

TERMINATE: DIAGNOSE CODE X'7C' SUBCODE X'0004'

The TERMINATE function notifies CP to drop a specific logical device. If the logical device is the console of a virtual machine, the virtual machine is placed in FORCE DISCONNECT state. If the logical device is DIALed to a virtual machine, it is detached from that virtual machine. If an input or output operation is being processed, it is terminated with a unit check and intervention required.

TERMINATE (ALL): DIAGNOSE CODE X'7C' SUBCODE X'0005'

The TERMINATE (all) function notifies CP to terminate all logical devices created for the issuing virtual machine.

STATUS: DIAGNOSE CODE X'7C' SUBCODE X'0006'

The STATUS function allows status to be returned to CP after an ACCEPT function is performed. It must be used with a logical 328x printer to indicate when the printer has completed the printout and the ending status of the printer.

External Interrupt Code X'2402'

The logical device support uses a service signal interrupt, (class 24 external interrupt) to notify the virtual machine of a change in status for a specific logical device. The external interrupt code is X'2402'. This interrupt causes a full word of data to be stored at location 128 (decimal) in the virtual machine. The interrupt is masked on and off by bit 22 of control register 0.

The format of the stored fullword is:

128-129 Logical device address

130 Flag byte
bit zero - PRESENT function purged
bit one - error in transmission or list

131 Interrupt reason code

The logical device address is returned to the user after an INITIATE, and must be specified by the user for an ACCEPT, PRESENT, or TERMINATE.

Flag byte, bit zero is set to 1 if the data from the last PRESENT has been discarded by the system (subsequent I/O to the logical device was a WRITE instead of a READ). Flag byte, bit one is set to 1 if an error was encountered in the address list describing multiple buffers of a data stream, or one of the specified addresses in the list was not accessible. Otherwise, the flag byte remains zero.

The reason codes are:

01 CP is terminating the connection.

02 A WRITE has been issued, so an ACCEPT must be done. (External interrupt flag byte, bit zero also indicates whether previous PRESENT data has been discarded.)

03 A previous PRESENT is now finished (user received CC=2 and RC=2 after a PRESENT).

A PRESENT has been suspended because of a transmission error. (External interrupt flag byte, bit one indicates this.)

04 A READ BUFFER has been issued.

05 A READ MODIFIED has been issued.

Reason code 1 indicates that the logical device no longer exists. The user receives a condition code 3 if he tries to perform another function with this device.

Logical Device Interrupt Code X'2402'

The logical device address is returned to the user after an INITIATE and must be specified by the user for an ACCEPT, PRESENT, STATUS, or TERMINATE.

Logical Device Restrictions

The devices supported are:

local 3277 Model 2
local 3278 Models 2, 3, 4, and 5
3279 Models 2 and 3
3284, 6, 7, 8, and 9 printers

Migration/Coexistence

For the INITIATE function of DIAGNOSE code X'7C', the high order byte of register Rx+1 indicates the following optional features:

Bit 0 - 3270 extended features to be supported
Bit 1 - ACCEPT function must be followed by STATUS function
Bit 2 - Specific device address requested

Existing applications that use the high-order byte of Rx+1 will experience migration and coexistence problems because this byte was not checked prior to this support.

DIAGNOSE Code X'80' -- MSSFCALL

All privilege classes (except ANY)

DIAGNOSE code X'80' is the VM interface for communicating between CP and the Monitoring and Service Support Facility (MSSF). MSSF is a hardware component of the processor controller of the 3081 processor complex; it provides system configuration and storage information for the 3081 processor complex.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'80':

Rx
Is a user-specified register that contains the address of the MSSF data block (MSFBLOK). MSFBLOK is defined in increments of 8 bytes to a maximum of 2048 bytes. It must be aligned on a 2K boundary. MSFBLOK is locked in storage during the MSSFCALL request.

Ry
Is a user-specified register that contains the MSSF command word representing the function that MSSF is to perform. (See MSSF command words below.)

Use: The CP module DMKMHC issues a real DIAGNOSE code X'80' and services all MSSF external interruptions. DMKMHC issues a real DIAGNOSE code X'80' when:

1. The operator issues a VARY ONLINE PROCESSOR nn command or a VARY OFFLINE PROCESSOR nn VPHY command. (These commands modify the real processor configuration to bring the processor physically on-line or off-line.)

2. A V=R virtual machine running in EC mode issues the MSSF command word SCPINFO.[6] (Operating systems running in a virtual machine use the MSSF SCPINFO command to get information about a system configuration and storage allocation.

3. A user with privilege class C or E runs the IOCP program to read from or write to the Input/Output Configuration Data Set (IOCDS).

When CP issues DIAGNOSE code X'80', the hardware call block that is created (HCBLOK) contains the MSFBLOK address, the MSSF command word, and the address to return to after the MSSF has processed the request. See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or

---

6 When a V=V virtual machine issues SCPINFO, DMKMHV does not pass control to DMKMHC. CP does not issue the DIAGNOSE code X'80' but simulates the MSSF response and returns predefined data and status codes to the user. See *VM/SP System Logic and Problem Determination Guide Volume 1 (CP)* or *VM/SP HPO System Logic and Problem Determination Guide - CP* for a description of the pre-defined data.

VM/SP HPO Data Areas and Control Block Logic - CP for the format of the MSFBLOK and HCBLOK.

MSSF Command Words

X'0011nn01' VARY ONLINE PROCESSOR nn where nn is the ID of the processor to be varied on-line. CP use only.

X'0010nn01' VARY OFFLINE PROCESSOR nn VPHY where nn is the ID of the processor to be physically varied off-line. CP use only.

X'00020001' SCPINFO command. Virtual machine use only.

X'00400002' IOCP WRITE to Level 0 IOCDS Side A

X'00400102' IOCP WRITE to Level 1 IOCDS Side A

X'00400202' IOCP WRITE to Level 2 IOCDS Side A

X'00400302' IOCP WRITE to Level 3 IOCDS Side A

X'00401002' IOCP WRITE to Level 0 IOCDS Side B

X'00401102' IOCP WRITE to Level 1 IOCDS Side B

X'00401202' IOCP WRITE to Level 2 IOCDS Side B

X'00401302' IOCP WRITE to Level 3 IOCDS Side B

X'00410002' IOCP READ from Level 0 IOCDS Side A

X'00410102' IOCP READ from Level 1 IOCDS Side A

X'00410202' IOCP READ from Level 2 IOCDS Side A

X'00410302' IOCP READ from Level 3 IOCDS Side A

X'00411002' IOCP READ from Level 0 IOCDS Side B

X'00411102' IOCP READ from Level 1 IOCDS Side B

X'00411202' IOCP READ from Level 2 IOCDS Side B

X'00411302' IOCP READ from Level 3 IOCDS Side B

Condition and Return Codes: Two possible condition codes returned for DIAGNOSE code X'80' are:

CC=0 MSSF is processing the MSSFCALL request.

CC=2 MSSF is busy.

Note: If CP issued the MSSFCALL request, and MSSF was already processing a previous MSSFCALL request, abend MHC001 occurs. If a V=R user issued the request, CP reflects the condition code (2) to the virtual machine's PSW.

At the completion of an MSSFCALL, the following actions occur:

- MSSF generates a service signal interrupt, external interrupt X'2401' (class 24 external interrupt). This interrupt stores the absolute address of the MSSF data block (MSFBLOK) at decimal locations 128-131 in the virtual machine's PSA. Bit 22 of control register 0 controls masking of the service signal.
- MSSF passes a completion status code back in the MSFBLOK.
- CP returns control to the address specified in HCBLOK.
- If tracing is on, trace table entry X'17' traces all MSSFCALL requests. In addition, trace table entry X'01' reflects external interrupt X'2401' when the MSSF generates the service signal interrupt to CP.

  Note: Please refer to the *VM Diagnosis Guide* for the format and content of the entries of the CP Internal Trace Table. This information is also available on the *VM CP Internal Trace Table (Poster)*, LX24-5202.

Successful MSSF completion status codes are:

0010 SCPINFO complete.

0020 The processor is varied on-line/off-line.

0120 The processor is already varied on-line/off-line. IOCP operation complete.

8020 An IOCP READ is invalid because the file is open for writing.

4020 The IOCP read or write operation was performed on the active IOCDS.

2020 The active IOCDS has been written to.

Reject status codes are:

01F0 Invalid command code or identification byte.

41F0 Attempt to read closed IOCDS.

0100 Data block not aligned on a 2K boundary.

0200 Data block length not a multiple of 8.

0300 The data block length is not adequate for the amount of information requested.

DIAGNOSE Code X'84' -- Directory Update-In-Place

Privilege class B

DIAGNOSE code X'84' enables a class B user to replace certain data in any entry of the VM directory. The user must specify the directory entry and may replace the following data:

- Logon password
- Virtual machine storage size
- Maximum virtual machine storage size
- Privilege classes
- Dispatching priority
- Logical editing symbols
- Initial program load (IPL) system
- IPL parameter data
- Account number
- Distribution word
- User options
- Minidisk access mode
- Minidisk read, write, or multiple password
- Options of the SCREEN directory control statement

With the exception of the account number, all changes to the entry take effect the next time the USERID associated with the entry logs onto VM. The account number may be updated such that the change

- Takes effect immediately,

- Takes effect immediately but is temporary lasting only until the USERID is logged off, or

- Takes effect the next time the USERID associated with the entry is logged on.

DIAGNOSE code X'84' cannot add new entries to the directory, cannot delete existing entries, nor can it alter directory user-description statements. It can only replace existing directory data. Data is replaced in the form of the directory created by the directory service program, that is, in VM control blocks.

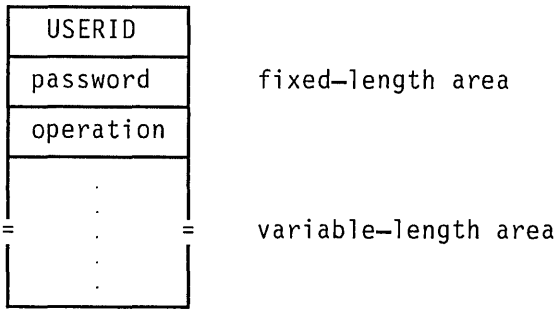
For a detailed description of the directory data, see the *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference.*

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'84':

Rx
Must point to a variable length parameter list.

Ry
Must specify, in bytes, the length of the list.

The list cannot be greater than 112 bytes long or less than zero bytes. The parameter list contains an area of fixed length followed by an area of variable length. Data in the fixed-length area identifies the directory entry to be updated, the password of the USERID associated with the entry, and the data field to be replaced in the directory entry. The variable-length area contains replacement data for the directory entry. All entries in the parameter list must contain unpacked, EBCDIC data. The parameter list is organized as follows:



Fixed-length area

USERID
The USERID of the user whose directory entry is updated. This is an eight-character, left-justified value and must be padded with blanks.

password
The current CP password of the USERID whose CP directory entry is updated. This is an eight-character, left-justified value and must be padded with blanks.

If this field is blank, the update-in-place function is processed in 'testmode'. When the DIAGNOSE is issued in this fashion, the directory is not updated. 'Testmode' lets you check the syntax of the directory statements without accessing the directory disk.

operation
An eight-byte, left-justified character string that identifies the data in the directory entry that is to be replaced. Valid values and the data that each identifies for replacement are defined in the description of the variable-length area which follows.

Variable-length area

The following diagram shows for each value of the operation field, the data that must be in the variable-length area of the parameter list, and the format and characteristics of the data.

Operation Field Value	Data	Characteristics/Format
LOGPASS	logon password	An eight-byte, left-justified value padded with blanks.
STORAGE	virtual machine storage size	An eight-byte, left-justified decimal value followed by the letter K. Pad with blanks following the letter K.
MAXSTOR	maximum virtual machine storage size	An eight-byte, left-justified decimal value followed by the letter K. Pad with blanks following the letter K.
PRIVLEGE	privilege classes	A 32-byte value where each byte represents a privilege class. Valid values for each byte are A through Z, and 1 through 6. All existing classes in the directory entry are replaced. Therefore, specify existing classes that are to be retained as well as classes that are to be changed. The data must be left-justified and padded with blanks. This field can be all blanks in which case the specified virtual machine will always have the default classes (defined via SYSFCN) each time the user logs on.
PRIORITY	dispatching priority	An eight-byte, left-justified value where the first two bytes, counting from the left, specify the dispatching priority. Valid values for these bytes are 1 - 99. Values 1 through 9 must be padded with a blank. The other six bytes are reserved for IBM use.
EDITCHAR	logical editing symbols	An eight-byte value where the first four bytes, counting from the left, are line edit symbols. The first or high-order byte is the "line-end" symbol, the second byte is the "line-delete" symbol, the third byte is the "character-delete" symbol, and the fourth byte is the "escape-character" symbol. All existing symbols in the directory are replaced. Therefore, specify existing symbols that are to be retained as well as symbols that are to be changed. Unspecified symbols must contain blanks. The last four bytes of the eight-byte value are reserved for IBM use.

Figure 4 (Part 1 of 4). DIAGNOSE Code X'84' -- Parameter List Operation Field

Operation Field Value	Data	Characteristics/Format
IPL	system name or virtual device address and variable data	A one-to-eight character value, left-justified and padded with blanks, followed by the keyword PARM and up to 48 characters of variable data. All existing values are replaced in the directory entry; therefore, specify values that are to be retained as well as values that are to be changed. Trailing blanks are not truncated but passed.
ACCOUNT	account number	A one-to-eight character value, left justified and padded with blanks. (This change takes effect the next time the USERID is logged on.)
IACCOUNT	account number	A one-to-eight character value left-justified and padded with blanks. (This change takes effect immediately.)
TACCOUNT	account number	A one-to-eight character value, left-justified and padded with blanks. (This change takes effect immediately but is temporary, lasting only until the USERID is logged off.) *Note:* DIAGNOSE code X'84' with TACCOUNT or IACCOUNT does not update the account number in the user accounting block (ACCTBLOK). DIAGNOSE code X'84' is for the directory update which updates the account number in the user's VMBLOK if change is to take effect immediately. The user accounting block (ACCTBLOK) is created by DIAGNOSE code X'4C' and the 'charge to' information in the ACCTBLOK can only be changed via DIAGNOSE code X'4C'. DIAGNOSE code X'84' options TACCOUNT and IACCOUNT have no effect on the accounting information in the ACCTBLOK.
DISTRIB	distribution identification word	A one-to-eight character value, left-justified and padded with blanks.

Figure 4 (Part 2 of 4). DIAGNOSE Code X'84' -- Parameter List Operation Field

Operation Field Value	Data	Characteristics/Format
OPTIONS	user options	An eighty-byte, left-justified value padded with blanks. Specify each option as a character string with a blank character between options. The following virtual machine OPTIONS may be updated using DIAGNOSE code X'84': • Realtimer • Ecmode • Isam • Virt=real • Acct • Svcoff • BMX • CPUID • AFFinity • LANG For a description of each option and a list of valid values, see the *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference.* All existing options except for 'CPUID' and 'AFFINITY', are replaced in the directory entry. Therefore, specify existing options (except for 'CPUID' and 'AFFINITY') that are to be retained, as well as options that are to be changed. The options field must be followed by the value X'FFFFFFFF' or by at least two blanks (X'4040').

Figure 4 (Part 3 of 4). DIAGNOSE Code X'84' -- Parameter List Operation Field

Operation Field Value	Data	Characteristics/Format
MDISK	minidisk address, access mode, read password, write password, and multiple password	A thirty-byte field defined as follows. All values must be left justified and padded with blanks. Valid values for the access mode and for passwords are defined in the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference.* Bytes 1-3, counting from the left, specify a minidisk address. This is the minidisk whose mode and passwords will be changed. The address must already exist in the directory entry. Bytes 4-6 specify the access mode. Bytes 7-14 specify the read password. Bytes 15-22 specify the write password. Bytes 23-30 specify the multiple password. The access mode, the read password, the write password, and the multiple password are replaced in the directory entry. Therefore, specify existing values that are to be retained as well as values that are to be changed.
SCREEN	display screen options	An eighty-byte area composed of ten doubleword fields. The ten fields are paired into five sets corresponding to the five display areas of the screen. You must specify these areas in the following order: 1. CP output 2. VM output 3. input redisplay 4. input area 5. status area. Each of the five doubleword sets has a color field and an extended highlight field. (See the SCREEN option description in the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference* for the valid color and extended highlight values.) Within each doubleword set you must specify the color first followed by the extended highlight value. You must specify all fields, including those you don't want to change. All of the options you specify must also be left justified in their eight-byte field.

Figure 4 (Part 4 of 4). DIAGNOSE Code X'84' -- Parameter List Operation Field

Condition and Return Codes: Before control is returned to the virtual machine, DIAGNOSE code X'84' sets a condition code and, if errors were detected, a return code in Ry. The condition codes and return codes are defined as follows:

Condition Code	Meaning
0	The directory was successfully updated.
1	DIAGNOSE code X'84' detected an error. The directory is unchanged. The return code defines the error.

Return Code	Meaning
10, 11	An error occurred writing the directory to a direct access device. To update the directory, use the directory service program described in the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference.*
20 - 25, 90, 112, 113	DIAGNOSE code X'84' encountered a processing error. To update the directory, use the directory service program described in the *VM/SP Planning Guide and Reference.* or the *VM/SP HPO Planning Guide and Reference.*
26	Specified minidisk address does not exist in directory entry, or specified userid does not have any devices defined in the directory entry.
27	No directory volume is linked in read/write.
28	The value in the OPERATION field of the parameter list is invalid.
30	The specified USERID could not be found.
31	The password specified in the fixed-length area of the parameter list does not match the current password of the USERID being updated.
40, 41	The value specified for the virtual machine storage size or for the maximum virtual machine storage size is too large. The maximum allowable size is 16 megabytes.
42, 43	The value specified for the virtual machine storage size or for the maximum virtual machine storage size contains a syntax error or an invalid character.
50, 51	The specified privilege classes are invalid.

52, 53	The specified privilege classes contain a syntax error or an invalid character.
60, 61, 62	The specified priority contains a syntax error or an invalid character.
63	The priority value is too large. The maximum allowable value is 99.
65, 66	Parameter list size error; if return code = 65, the list exceeds 112 bytes; if return code = 66, the list size is less than zero bytes long.
70	A specified option is invalid. Refer to the description of OPTIONS in Figure 4 on 76 for a list of OPTIONS that can be updated with DIAGNOSE code X'84'.
71	The value X'FFFFFFFF' was not coded after the list of options.
72	The option value contains a syntax error or an invalid character.
80	The parameter list contains an invalid minidisk address.
81	The parameter list specifies an invalid access mode for a minidisk.
82, 83	The minidisk read, write, or multiple password specified in the parameter list requires a change in the size of the directory entry.
91	No attributes were found on the SCREEN command.
92	Invalid attributes were found on the SCREEN command.
101	The parameter list is too large.
102	The parameter list is less than 1.
110	No parameter data currently exists in the directory entry.
111	The parameter length is invalid.

DIAGNOSE code X'8C' -- Access Certain Device Dependent Information

All privilege classes (except ANY)

DIAGNOSE code X'8C' allows a virtual machine to obtain certain device-dependent information without issuing a WRITE STRUCTURED FIELD QUERY (WSF QUERY). DIAGNOSE code X'8C' retrieves this information from the RDEVBLOK or NICBLOK and a pageable buffer and creates a DIAGNOSE interface to enable the virtual machine to access it.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'8C':

Rx
Is the address of user-provided data buffer.

Rx+1
Is the virtual device address, or the value negative 1 (-1). Specify -1 for Rx+1 when the device is the virtual console of the user issuing the DIAGNOSE code X'8C'.

Ry
Is the length of user-provided data buffer.

Exit Values: The data returned by DIAGNOSE code X'8C' is in the following format:

Byte 0	Byte 1	Byte 2 and 3	Byte 4 and 5	Byte 6-n (n < 502)
Flags	Number of partitions	Screen width in cells	Screen height in cells	WSF Query Reply data

Flags
80 = extended color
40 = extended highlight
20 = Programmable Symbol Sets (PSS) available
02 = 3270 Emulation Feature
01 = 14-bit addressing

Return Codes: Upon completion, Rx+1 contains the following return codes:

0 If the DIAGNOSE completes successfully.

4 If an I/O error occurs.

Residual Count: If the user receives less data than he requested, the difference between the amount of data requested and the amount received is returned in Ry.

Program Exceptions: The user receives the following exceptions:

Addressing
If an invalid buffer address is specified.

Protection
The user receives a storage protection exception of the address in Rx is a protected area. For example,

- A CMS module
- The nucleus area.

Specification
If the length specified is negative.

If the virtual device address specified is invalid. If the virtual device is present, but the real device is not, it is considered an invalid virtual device address.

If the buffer address is not on a doubleword boundary.

Notes:

1. *Devices for which the write structured field is not applicable return zeroes in bytes 0 and 1, and screen width and height in the remaining bytes.*

2. *For the 3290 Information Panel, dynamic changes in the characteristics of the terminal are obtained by the virtual machine when the user disconnects or logs off from the logical terminal. This change, therefore, is reflected in the data returned by DIAGNOSE code X'8C'.*

3. *If a paging error occurs when paging in the pageable buffer, six bytes of information are returned in the user-provided data buffer, and the residual count is returned in the user's Ry.*

DIAGNOSE code X'94' -- VMDUMP Function

All privilege classes (except ANY)

DIAGNOSE code X'94' allows a virtual machine to request dumping of its virtual storage to a spool file for use with the Interactive Problem Control System (IPCS). Upon completion of DIAGNOSE code X'94' execution, control is returned to the invoker with a condition code set indicating the status of the DIAGNOSE function. DIAGNOSE code X'94' uses the VMDUMP command dump processor to produce the dump.

Only storage that the user is authorized to access will be dumped. Any storage which is fetch protected (via storage keys and the fetch protection bit) will:

- Be replaced by the mask value of X'EE' if only 2K of the 4K page violated fetch protection
- Not be dumped if the page is a single-key 4K block.

Only second level storage (storage that is created for the guest virtual machine) is dumped. Certain operating systems running in a guest virtual machine such as OS/VS and DOS/VSE have virtual storage (third level) of their own. CP cannot dump this third level storage directly.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'94':

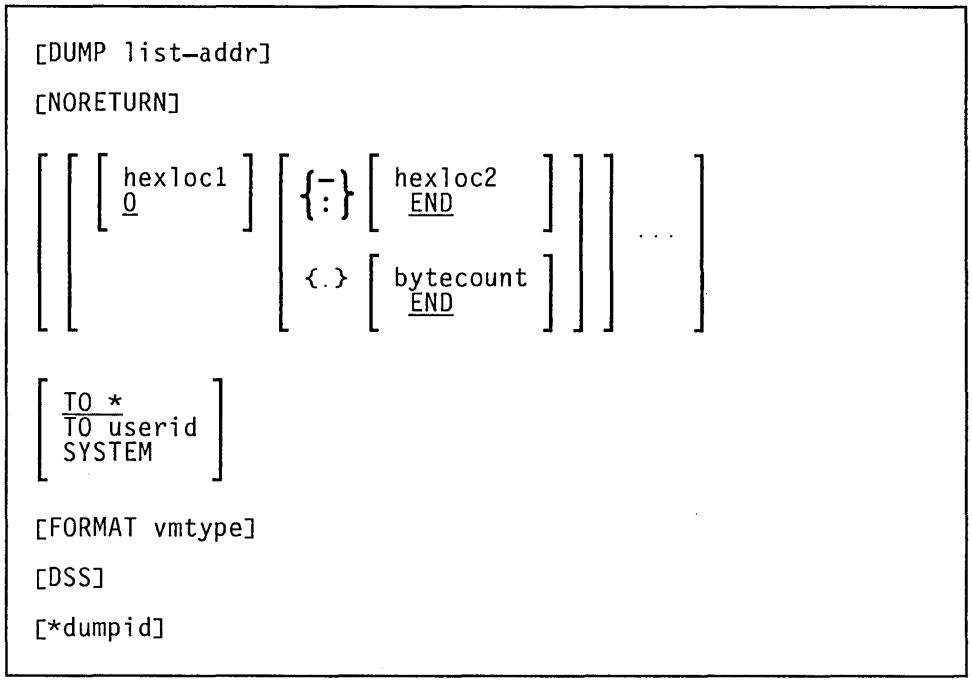
Rx
Must contain the pointer to the parameter list in the virtual machine storage.

Ry
Must contain the length of the parameter list.

The parameter list cannot span a 4K page boundary, cannot exceed 240 bytes (30 doublewords), and must not reside in fetch protected storage.

Supported Parameters

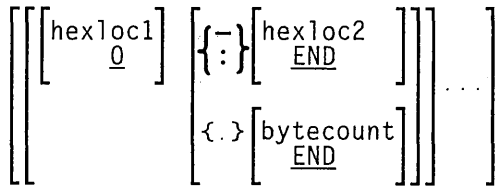
The parameters supported by DIAGNOSE code X'94' are:



DUMP list-addr
indicates that the dump address ranges are in hexadecimal within a separate parameter list.

list-addr is the address of the dump address list. It is expressed as a hexadecimal fullword immediately following the DUMP keyword with a blank separating list-addr from the DUMP keyword. Additional information about the dump address parameter list can be found in "Dump Address Parameter List."

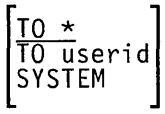
NORETURN
indicates that the invoker may not transfer the file back to himself. This parameter must be used with the "TO userid" or "SYSTEM" operand where "userid" is not the invoker of the DIAGNOSE instruction.



hexloc1
is the first or only hexadecimal virtual storage address dumped. If you omit the hexloc1 operand, the default is zero, the beginning of virtual machine storage.

hexloc2
is the last hexadecimal virtual storage address dumped. If you do not specify the hexloc2 operand, the default is END, and CP dumps the contents of virtual machine storage starting from hexloc1 to the end of virtual storage.

bytecount
is the hexadecimal number of bytes dumped. If you do not specify bytecount, the default is END, and CP dumps the contents of virtual machine storage from the first byte at hexloc1 to the end of virtual storage.



defines where the dump is to be transferred.

If you enter an asterisk (*) with the TO, CP transfers the dump to your virtual card reader.

If you enter a userid with the TO, CP transfers the dump to that user's virtual card reader.

If you enter SYSTEM, CP transfers the dump to the virtual card reader of the userid specified on the SYSDUMP operand of the SYSOPR system generation macro instruction. You must not specify TO preceding the keyword SYSTEM.

FORMAT vmtype
provides VM/SP IPCS with the virtual machine type (vmtype) which IPCS uses to format the dump.

vmtype is a one-to-eight byte name of the operating system running in a virtual machine (for example, CMS). CP also uses the specified vmtype as the virtual card reader filetype. CP does not validity check the vmtype. Any vmtype longer than eight bytes generates an error message and halts further VMDUMP processing. The dump header record includes your specific vmtype and IPCS uses the vmtype information to format the dump. If you enter FORMAT, you must also specify a vmtype. If you do not specify FORMAT, the default vmtype is FILE.

DSS
specifies that CP take a dump of all discontiguous saved segments in use by your virtual machine. When "DSS" is specified, only the discontiguous saved segments will be dumped unless you explicitly specify other locations. If "DSS" is specified, it overrides the hexloc1-hexloc2 defaults of 0 and end.

***dumpid**
is a line of user input up to 100 characters long including imbedded blanks and asterisks which you can enter for your own benefit (that is, for descriptive purposes, such as the time and date of the dump, or what was being processed at the time of the dump). If you specify this operand, it becomes the DMPDMPID field in the dump file information record (DMPINREC) data area. If specified, you must enter *dumpid as the last operand in the parameter area.

If you use the FORMAT CP operand, the dumpid information is not saved. While using IPCS, the dumpid information is not available when requested. CP dumps do not contain dumpid information.

Notes:

1. *The DUMP operand and the inline hexadecimal ranges may not be specified together on the same invocation (e.g. DUMP xxxx 4-5).*

2. *Absence of the DSS (Discontiguous Saved Segments) when DSS is specified causes an error only if DSS is the only dump address range specified.*

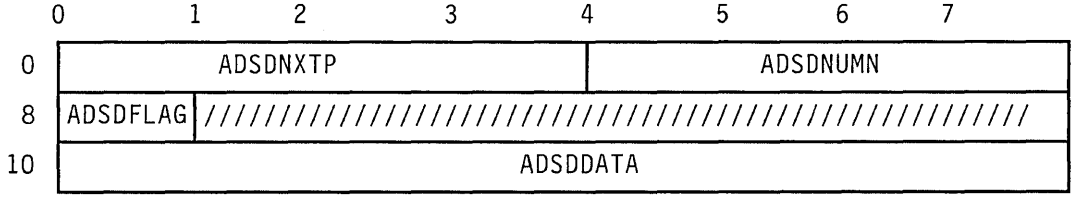
3. *Except for the *dumpid operand, you can specify the operands of the DIAGNOSE code X'94' in any order. However, if you specify the *dumpid operand, it must be the last operand on the command line.*

4. *The first asterisk of the dumpid is not included in the 100 characters for the user input.*

5. *The DUMP address list must contain a minimum of one range.*

Dump Address Parameter List

You must create your own dump address parameter list. An example of a dump address parameter list showing the format and content follow. In the example, field names are given for reference purposes only. As you create your dump address parameter list, you can use names of your own choice.



where:

ADSDNXTP
(Displacement 0, Length 4) Address of the next list. This field is zero if there is no additional list needed. The condition under which such a list would be used is where the invoker has built multiple lists of storage ranges to be dumped.

ADSDNUMN
(Displacement 4, Length 4) Number of address ranges in this list.

ADSDFLAG
(Displacement 8, Length 1) Flag field for dump list data.

Bit 0 = 0 - the dump list has the starting address and length of the area to be dumped.

Bit 0 = 1 - the dump list has the starting and ending address of the area to be dumped.

All other bits are reserved and should be set to zero.

ADSDDATA
(Displacement 10, Length is variable) Start of variable number of entries. Beginning at ADSDDATA, the following structure is repeated for the number of times equal to the value given in ADSDNUMN.

ADSDSTRT
(Displacement 0, Length 4) Starting address of storage to be dumped.

ADSDSTOP
(Displacement 4, Length 4) Length of storage area to be dumped or address of last storage byte to be dumped, as determined by ADSDFLAG setting.

Condition and Return Codes: Upon completion of DIAGNOSE code X'94' execution, control is returned to the invoker with a condition code set to indicate the status of the DIAGNOSE. The condition codes returned to the invoker are:

Condition Code	Result
0	Function completed successfully. All requested ranges have been dumped.
1	Function completed unsuccessfully. Portions of the requested ranges have been dumped. Ry contains a numeric value (the return code) which indicates the reason for the failure.

Condition Code	Result
2	Unsuccessful completion. No dump has been created. Ry contains a numeric value (the return code) which indicates the reason for the failure.

The return codes returned to the invoker are:

Return Code	Condition Code	Description
0	0,1	Successful completion.
4	2	Parameter list exceeds 240 bytes (30 doublewords)
8	1,2	System I/O error
C	1,2	Violation of fetch protection
10	2	Invalid range
14	2	Conflicting option
18	2	Userid missing or invalid
1C	2	Hexloc missing or invalid
20	2	Parameter missing
24	2	Userid not in directory
28	2	Spooling error
2C	2	Hexloc exceeds storage
30	2	List spans page boundary
34	2	Invalid address pointer

Detailed Description of Return Codes:

Return Code **Meaning**

0 Successful completion.

CC=0 indicates that a dump of all the requested area has been created.

CC=1 indicates that a dump of only a portion of the requested area has been created. This can occur when a valid address range and DSS (discontiguous saved segments) are requested but no discontiguous saved segments are loaded for the user.

4 The parameter list exceeds 240 bytes (30 doublewords). No dump has been created.

8 System I/O error.

CC=1 indicates that the system failed to bring in a 4K page within and address range. This causes a partial dump to be created.

Note: No dump is created if the 4K page is the only page requested.

CC=2 indicates that the system failed to bring in the parameter list or the dump address list. These conditions do not create a dump.

C Violation of fetch protection; the storage keys do not match.

CC=1 indicates that the system found a 4K page within an address range. A partial dump is created.

Note: No dump is created if the 4K page is the only page requested.

CC=2 indicates that a parameter list or dump address list is fetch protected. No dump is created.

10 Invalid range. The ending address is less than the starting address or the maximum number of ranges (2,049) was exceeded. No dump has been created.

14 Conflicting option. No dump has been created.

18 Userid is missing or invalid. Userid consists of more than eight characters. No dump has been created.

1C Hexloc missing or can not be converted into hexadecimal, or the value specified for the number of address ranges in the address list is zero. No dump has been created.

20 Required parameter missing. No dump has been created.

24 Userid not in directory. No dump has been created.

28 Spooling error. No dump has been created.

2C Hexloc exceeds storage size. No dump has been created.

30 Parameter list or dump list spans the page boundary. No dump has been created.

34 Invalid address pointer. Pointer points to storage outside of the user defined storage area. No dump has been created.

DIAGNOSE Code X'98' -- Real Channel Program Support

All privilege classes (except ANY)

Using DIAGNOSE code X'98', a virtual machine can lock and unlock virtual pages, and it can execute its own real channel programs.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'98':

Rx
Contains a hexadecimal code indicating the operation to be performed. The possible codes are:

Code	Function
0000	Lock a virtual page
0004	Unlock a virtual page
0008	Perform I/O on a real CCW string

Ry
Contains either the virtual address of a user's page to be locked or unlocked, or else, if Rx contains X'0008', a virtual device address.

The page address in Ry must fall within the virtual machine's storage size. DIAGNOSE code X'98' does not support operations on pages in saved segments which lie outside of the virtual machine's storage size.

Exit Values: Ry+1 contains a return code if the operation was unsuccessful (indicated by condition code 3). If a lock operation was successful, Ry+1 contains the real address of the successfully locked page.

Program Exceptions: The user receives the following exceptions:

Operation
If a user tries to use DIAGNOSE code X'98' without being authorized by the DIAG98 directory option. DIAGNOSE code X'98' is provided to only those users defined as being authorized in the system directory by the DIAG98 option.

Specification
If Ry is equal to 15, and Rx is equal to Ry or Ry+1.

Note: A virtual machine should only use the real addresses returned by DIAGNOSE code X'98' in its real channel programs. The virtual machine is responsible for any security violations it may cause from using any other real addresses.

Subcode X'0000' -- Lock a Virtual Page

Rx = function subcode
Ry = virtual address of a page to be obtained and locked
Ry+1 = real address of the user's page if the lock attempt was successful; a return code if the locking operation was unsuccessful (indicated by condition code three).

Return Code	Error
1	User is running virtual=real
2	Invalid virtual address
3	Page unavailable in dynamic page area
4	Page already locked

DIAGNOSE code X'98' subcode X'0000' locks in real storage a selected page of a user's virtual storage, thus excluding the page from future paging activity. Locking pages can enhance the efficiency of a particular virtual machine by keeping frequently-used pages in real storage.

If too many pages of real storage are locked, other virtual machines may not have enough available remaining pages to operate efficiently. This can severely degrade the throughput in all virtual machines because of excessive contention for the remaining available page frames. So, if the amount of page frames available for paging is limited, DIAGNOSE code X'98' subcode X'0000' should not be used without the system programmer's approval.

Once a page is locked, it remains locked until the user either logs off the system or issues the UNLOCK command for that page. If a user with the locked pages option in effect re-IPLs the system by device address and specifies the clear option, the locked pages are unlocked and available to the system being loaded. If a user with the locked pages option in effect re-IPLs the system by device address and doesn't specify the clear option, all locked pages remain locked except the page given to DMKVMI for IPL. If a user with the locked pages option in effect re-IPLs the system by name (shared system), the locked pages are unlocked only if the locked pages are not in the shared segment or if the page is in the shared segment and the user who is re-IPLing is the last user of the shared segment. In addition, issuing DIAGNOSE codes X'14', X'30', X'34', or X'38' against a locked page causes the page to become unlocked. Shared pages cannot be locked in a system generated for AP or MP operation.

The virtual pages locked in processor storage are blocks of 4K (4096) bytes. This block of storage need not represent all of the user's virtual storage. DIAGNOSE code X'98' subcode X'0000' may be issued as many times as required for one virtual machine to lock noncontiguous pages of storage. The remaining virtual machine storage blocks may remain pageable.

Note: For a user's virtual storage, DIAGNOSE code X'98' subcode X'0000' operates exactly like the CP LOCK command.

Subcode X'0004' -- Unlock a Virtual Page

Rx = function subcode
Ry = virtual address of page to be unlocked
Ry+1 = return code if the operation was unsuccessful (indicated by condition code three).

Return Code	Error
1	User is running virtual=real
2	Invalid virtual address
3	Page already unlocked

DIAGNOSE code X'98' subcode X'0004' unlocks a page of a virtual machine that was previously locked by a DIAGNOSE code X'98' subcode X'0000' or a CP LOCK command. Once pages are unlocked, they are available to CP for other virtual machine paging operations.

Note: For a user's virtual storage, DIAGNOSE code X'98' subcode X'0004' operates exactly like the CP UNLOCK command.

Subcode X'0008' -- Perform I/O on a Real CCW String

Rx = function subcode
Ry = virtual device address (bits 16-31)
Ry+1 = return code if the operation was unsuccessful (indicated by condition code three).

Return Code	Error
1	Device not operational
2	Device not dedicated
3	Virtual CAW key is zero

Condition codes 0, 1, and 2 are compatible with normal virtual I/O so that conditions can be consistently reflected to the virtual machine.

DIAGNOSE code X'98' subcode X'0008' executes the real channel program built by the virtual machine. To do this, it interfaces with the current virtual I/O support to perform actual I/O.

Notes:

1. *The I/O for this function performs identically to the virtual SIOF instruction, except that all I/O addresses are real addresses and channel program translation can be bypassed.*

2. *The user must coordinate locking of pages containing CCWs and data areas by using the lock and unlock subfunctions.*

3. *The I/O initiated by DIAGNOSE code X'98' cannot be traced by the CP TRACE command.*

DIAGNOSE Code X'A0' -- Retrieve a Group Name

All privilege classes (except ANY)

DIAGNOSE code X'A0' retrieves a group name for a given userid.

Note: This group is not related to groups defined for the Group Control System (GCS) component of VM.

This DIAGNOSE, along with access verification routines, can increase security on your VM system. For more information on access verification routines, see *VM/SP CP for System Programming* or *VM/SP HPO CP for System Programming.*

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'A0':

Rx
Contains the address of a field consisting of two doublewords. The first doubleword is a userid. The second doubleword is empty.

Ry
Contains the subcode X'00'. This requests CP to retrieve a group name.

Exit Values: If a group name exists for the given userid, CP returns the group name in the second doubleword of the two doublewords pointed to by Rx.

Condition Codes: DIAGNOSE code X'A0' sets the following condition codes:

Condition Code	Meaning
0	Request completed successfully
1	Request failed

DIAGNOSE Code X'B0' -- Access Diagnostic Information Saved For Protected Application Facility Users

All privilege classes (except ANY)

DIAGNOSE code X'B0' lets a virtual machine access diagnostic information saved on behalf of a user running with the protected application facility invoked, for whom a re-IPL has been attempted. This information consists of the information normally displayed for one of the following errors:

- Shared page altered
- Virtual machine disabled wait
- Paging error

- Invalid PSW
- External interrupt loop
- Program interrupt loop
- Translation exception.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'B0':

Rx
Contains the virtual address of the user's buffer.

Ry
Contains the buffer length. Any non-negative buffer length is allowed, as long as the user's buffer does not cross a page boundary.

Exit Values: If the DIAGNOSE completed successfully, the first byte of the user's buffer contains one of the following codes:

X'01' Disabled Wait PSW
X'02' External Interrupt Loop
X'03' Paging Error
X'04' Program Interrupt Loop
X'05' Shared Page Altered
X'06' Translation Exception

If the code indicates a Disabled Wait PSW condition, the remainder of the user's buffer contains:

Bytes 2-9 8-byte binary Disabled Wait PSW

If the code indicates an External Interrupt Loop, the remainder of the user's buffer contains:

Bytes 2-9 8-byte External Old PSW

If the code indicates a Program Interrupt Loop, the remainder of the user's buffer contains:

Bytes 2-9 8-byte Program Old PSW

If the code indicates a Shared Page Altered condition, the remainder of the user's buffer contains:

Bytes 2-9 8-byte character representation of the shared system name
Bytes 10-13 4-byte binary page address

Rx still contains the virtual address of the user's buffer. Ry contains the completion code.

Completion Codes: The completion codes returned in Ry are:

Code Meaning
0 Operation successful

4 No re-IPL information found
8 I/O error paging in the user's buffer

Programming Exceptions: The user of DIAGNOSE code X'B0' receives the following exceptions:

Addressing
If the buffer address specified is invalid.

Protection
If a storage protection violation has occurred.

Specification
If a negative buffer length is specified.

If the buffer crosses a page boundary.

DIAGNOSE Code X'B4' -- Virtual Printer External Attribute Buffer Manipulation

All privilege classes (except ANY)

DIAGNOSE code X'B4' allows a user to associate an external attribute buffer (XAB)[7] he provides with his virtual printer device. A user is able to:

- Read the existing XAB into the storage of his virtual machine
- Write or rewrite an XAB
- Determine the size of an existing XAB
- Determine if an XAB has been defined
- Erase the XAB.

Each time a print file is CLOSED for the virtual printer, CP adds the XAB information to the spool file. Both the character data actually written to virtual printer and the XAB data for a print file are kept on SPOOL in blocks called SPLINKs.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'B4':

Rx
Contains a 4-byte (unsigned) buffer address.

---

[7] The XAB is a collection of data that defines how a print spool file is to be processed. Each print file may have its own XAB, but the XAB is optional. CP has the facilities to maintain the XABs.

Ry

Is divided up as follows. Bytes 0 and 1 of Ry contain the 2-byte (unsigned) length of the XAB buffer. The maximum valid length is 32K-1 bytes. Bytes 2 and 3 of Ry contain the 2-byte virtual printer address.

Ry+1

Is divided up as follows. Bytes 2 and 3 of Ry+1 contain a 2-byte hexadecimal code indicating the function to be done. The function subcodes are:

Code	Function
0000	Read external attribute buffer
0004	Write to or erase the external attribute buffer

Condition Codes: On return, byte 0 of Ry+1 may contain error codes that further define a returned condition code of 2. Bytes 0 and 1 of Ry contain the length of the XAB that overlays the original input value for the buffer length if CC=0, and if CC=2 with the error code in Ry+1 set to '08'. If an XAB does not exist for the specified virtual printer, then a value of 0 is returned as the length.

Notes:

1. *The XAB is erased if the length specified for the XAB is zero and the function is '0004' (WRITE). The virtual printer is marked to indicate that there is no XAB associated with it.*

2. *If an invalid length is specified, then the the virtual storage address field of the XAB is not checked for validity.*

3. *The total storage that would be changed by reading the XAB must be addressable using the PSW key of the current PSW at the time the DIAGNOSE code is issued. This storage must not include any portion of a protected shared segment.*

Condition Code	Ry+1	Error
0	X'00'	Read, write, or erase function successful
2	X'04'	CUU not a valid virtual printer device
2	X'08'	Length of XAB buffer greater than the user buffer
2	X'0C'	Length of user buffer invalid
2	X'14'	CP I/O error during paging operation
2	X'1C'	Invalid subcode specified in bytes 2 and 3 of Ry+1
2	X'20'	User buffer address invalid
2	X'24'	User buffer in protected storage
2	X'28'	Erase functions unsuccessful (no existing XAB)

Note: If register 15 is specified for Ry, then no Ry+1 return code is given; but, PSW condition code=2 is set.

Considerations

- The maximum length of the user buffer is 32K-1 bytes (32,767 bytes).
- The length of the XAB overlays the input value for the user buffer length. If an XAB does not exist for the specified virtual printer, then a value of 0 is returned as the length.
- The total storage that would be changed by reading an XAB must be addressable using the PSW key of the current PSW at the time the DIAGNOSE code is issued. This storage must not include any portions of a protected shared segment. For a WRITE operation, the storage location that contains the XAB data must not violate fetch protection. For a READ operation, the storage location to receive the XAB data must not violate storage protection.

External Attribute Buffer (XAB)

The External Attribute Buffer (XAB) is a control block that contains data you create to specify additional information about a print file. Each print file has its own XAB and CP has the facilities to maintain XABs.

Suggested Format for an External Attribute Buffer

The following shows the suggested format for data contained in an External Attribute Buffer. The basic design of the format is to create separate blocks within the XAB. Using this format, you can create or locate a specific block without affecting other blocks within the XAB.

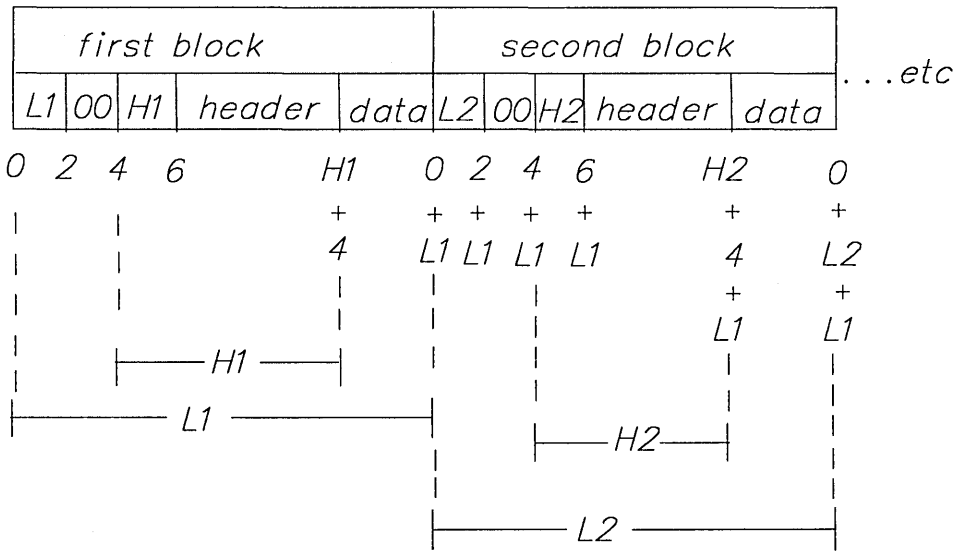


Figure 5. Suggested Format of an External Attribute Buffer

Lx
is the length of a block. This is a 2-byte field which specifies the total number of bytes for this block.

00
is reserved.

Hx
is the length of the header. This is a 2-byte field which specifies the total number of bytes used for header information. The value is the size of the header + 2 (for the size of the Hx field itself).

header
is the header data. This is a variable size field which identifies the block.

For multiple independent blocks to be contained in the XAB, it is necessary for each block to have a unique *header*.

- The first part of the header should be a name or character string that uniquely identifies who or what is defining the block. For example, a company name or trademark.
- The second part of the header should be the name of the product associated with the company or trademark.
- The third part of the header should be the name of the block.
- The fourth part of the header should be format level for the block. If a change is made to a defined block, it is reflected in the format level for that block.

The following are examples of headers.

Blocks defined for products from IBM might use a header like:

```
IBM - VM/SP - BLOCK XYZ - LEVEL 0.0.0
```

And if IBM changed this block, the header might look like:

```
IBM - VM/SP - BLOCK XYZ - LEVEL 1.0.0
```

Blocks defined for products from company JJKKLL might use a header like:

```
JJKKLL - PROD1 - BLOCK OPQ - LEVEL 0.0.0
```

data
is the actual data for the block. This is a variable size field.

Note: CP has no restrictions on the content of an XAB except that the total size of the XAB cannot exceed 32K-1 bytes (32,767 bytes). Although CP does not check to see that the standard format has been followed, we

recommended that each user of an XAB use the suggested format for multiple uses of the XAB.

DIAGNOSE Code X'B8' -- Spool File External Attribute Buffer Manipulation

All privilege classes (except ANY)

DIAGNOSE code X'B8' allows an application virtual machine to read, write, or erase an external attribute buffer (XAB)[8]. A user is able to:

- Read the existing XAB associated with the file
- Write or rewrite an XAB
- Determine the size of an existing XAB
- Determine if an XAB has been defined
- Prevent a file from being SELECTed while the XAB is being changed
- Erase the XAB.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'B8':

Rx
Contains a 4-byte (unsigned) buffer address.

Ry
Is divided as follows. Bytes 0 and 1 of Ry contain the 2-byte (unsigned) length of the XAB. The maximum valid length is 32K-1 bytes. Bytes 2 and 3 of Ry contain the 2-byte hexadecimal spoolid of the spool file that is the target of the function.

Ry+1
Is divided as follows. Byte 1 of Ry+1 contains a flag indicating the special processing of the file. Bytes 2 and 3 of Ry+1 contain a 2-byte hexadecimal code indicating the function to be done. The 1-byte flag in Ry+1 can be any one of the following:

X'00' No special processing
X'01' Spool file placed in USER HOLD status
X'02' Spool file placed in USER NOHOLD status
X'04' Spool file to be read is on the RDR chain rather than the PRT chain

The function subcodes are:

---

[8] The XAB is a collection of data that defines how a print spool file is to be processed. Each print file may have its own XAB, but the XAB is optional. CP has the facilities to maintain the XABs.

Code	Function
0000	Read external attribute buffer
0004	Write to or erase the external attribute buffer

Condition Codes: On return, byte 0 of Ry+1 may contain error codes that further define a returned condition code of 2. Bytes 0 and 1 of Ry contain the length of the XAB that overlays the original input value for the buffer length if CC=0, and if CC=2 with the error code in Ry+1 set to '08'. If an XAB does not exist for the spool file, then a value of 0 is returned as the length.

Notes:

1. *If the length of 0 is specified for the XAB, then the erase function is done. The spool file is marked to indicate that there is no XAB associated with it.*

2. *If an invalid length is specified, then the virtual storage address field of the XAB is not checked for validity.*

3. *The total storage that would be changed by reading the XAB must be addressable using the PSW key of the current PSW at the time the DIAGNOSE code is issued. This storage must not include any portion of a protected shared segment.*

Condition Code	Ry+1	Error
0	X'00'	Read, write, or erase function successful[9]
2	X'04'	Invalid spoolid
2	X'08'	Length of XAB buffer greater than the user buffer[9]
2	X'0C'	Length of user buffer invalid.
2	X'14'	CP I/O error during paging operation
2	X'18'	Invalid special processing flag
2	X'1C'	Invalid subcode specified in bytes 2 and 3 of Ry+1
2	X'20'	User buffer address invalid
2	X'24'	User buffer in protected storage
2	X'28'	Erase functions unsuccessful (no existing XAB)
2	X'2C'	Spool file not available

Note: If register 15 is specified for Ry, then no Ry+1 return code is given; but, PSW condition code=2 is set.

[9] Any special processing requested by the flag byte is completed.

DIAGNOSE Code X'BC' -- Open A Spool File

All privilege classes (except ANY)

DIAGNOSE code X'BC' opens[10] a spool file for a spooled reader device and returns spool file identification into a user buffer. If a file is already open on the reader device, DIAGNOSE code X'BC' returns spool file identification for the open file regardless of the reader class.

DIAGNOSE code X'BC' allows a program running in a virtual machine to open a file with the appropriate class for a spooled reader device. The appropriate class is the current class of the spooled reader device. The program receives the same information received from issuing the following commands:

- QUERY READER spoolid
- QUERY READER spoolid ALL
- QUERY READER spoolid TBL.

Entry Values: Set the input registers up as follows when invoking DIAGNOSE code X'BC':

Rx
Contains the virtual address of a buffer.

Rx+1
Contains the length of the buffer.

Ry
Contains the virtual address of a spooled reader device.

Exit Values: Upon return, Ry+1 contains a return code.

Depending on the specified buffer length, the user's buffer contains as much of the following information as possible.

Character Length (in bytes)	Description
4	Spool file id (EBCDIC)
8	Userid
1	Class
3	Type
8	REC -- Number of records (EBCDIC)
3	Copies (EBCDIC)
12	Filename
12	Filetype
8	Date
8	Time

[10] A file is opened when an SIO to the spooled device is issued.

8	Distribution
4	Status -- 'NONE'
8	FORM -- User forms
8	Destination
4	Flash name
3	Flash count (EBCDIC)
4	FCB -- Forms control buffer
4	CMOD -- Character modification
1	Character modification count (EBCDIC)
3	Load 3800 -- 'ANY' 'BEG' 'NO '
16	CHARS -- Character Arrangement Tables
8	SIZE -- Number of SPLINKs (EBCDIC)

Condition and Return Codes: Upon completion, DIAGNOSE code X'BC' returns the following condition and return codes:

Condition Code	Ry+1	Meaning
0		Data transfer is successful
2		No file is found
3	04	The device address is invalid
3	08	The device type is invalid
3	12	Device busy, not ready, or a real reader
3	16	A paging I/O error is received

Program Exceptions: If DIAGNOSE code X'BC' is specified incorrectly, the following exception is received:

Addressing
If the user's buffer address is invalid.

Protection
If the user's buffer address storage key is invalid.

Specification
If the buffer length is less than or equal to 0.

If the buffer address is equal to 0.

If either Rx or Ry is specified as R15.

If Ry is specified as Rx.

DIAGNOSE Code X'C8' -- Set Language

All privilege classes (except ANY)

DIAGNOSE code X'C8' initiates the SET function required to make a national language available.

When the SET function executes, CP finds or builds the control block associated with the CP message repository for that language. The address

of that control block is saved in the user's VMBLOK. CP uses this language to issue most CP system messages.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'C8':

Rx and Rx+1
Contain the identifier (langid) of the language to be set. This langid must correspond to a LANGID parameter that is on the NAMELANG macro in the DMKSNT entry for this language. Specify the language identifier using up to 5 characters. Place the langid in the register pair so that the first four characters of the langid are in Rx and the fifth character is in the high-order byte of Rx+1. Set the three unused bytes of Rx+1 to binary zeroes. If the langid is less than five characters, it should be padded on the right with blanks as required.

Ry
Contains the function code for SET (X'00') in the high-order byte.

Exit Values: When processing of the SET function completes, the first five bytes of the register pair Rx, Rx+1 contain the langid for the language currently set for CP messages. If the langid is less than five characters, it will be padded on the right with blanks as required.

Return Codes: The low-order byte of Ry contains one of the following return codes:

Code	Meaning and Action
X'00'	The language requested has been set.

If the return code is anything besides X'00', the requested language has not been set. Also, the language used to issue CP messages does not change.

X'04'	The DMKSNT entry for the language specified does not exist. Specify the appropriate NAMELANG macro in DMKSNT for this language.
X'08'	The volid specified in the DMKSNT entry for the language is not a CP-owned volume. Ensure a CP-owned volume is specified in the DMKSNT entry generated by the NAMELANG macro for this language.
X'0C'	The volid specified in the DMKSNT entry for the language is not mounted. Ensure that the appropriate volume is mounted.
X'14'	A paging error occurred during the SET operation.
X'1C'	The DMKSNT entry for the language was found; however, the langid in this DMKSNT entry does not match the langid in the saved message repository. Ensure that the NAMELANG entries do not specify overlapping areas on DASD.

X'20' The "MSGREP" identifier was not found on the first page of the requested message repository. CP looks for this identifier to determine if a valid message repository is saved. Save the appropriate CP message repository.

X'24' No more virtual page buffers are available.

Program Exceptions: If DIAGNOSE code X'C8' is specified incorrectly, the following exception is received:

Specification
If the input registers are set up incorrectly for the DIAGNOSE.

DIAGNOSE Code X'CC' -- Saving the CP Message Repository

Privilege Class E

Use DIAGNOSE code X'CC' to initiate the SAVE function for the CP message repository.

When the SAVE function executes, CP finds the DMKSNT entry for the language to be saved. This entry specifies the DASD location where CP saves its message repository. If the SAVE operation completes successfully, then DIAGNOSE code X'C8' can be used to set that language. The CMS SET LANGUAGE command can also be used provided the CMS language files have previously been saved.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'CC':

Rx and Rx+1
Contain the identifier (langid) of the language to be saved. This langid must correspond to a LANGID parameter that is on the NAMELANG macro in the DMKSNT entry for this language. Specify the language identifier using up to 5 characters. Place the langid in the register pair so that the first four characters of the langid are in Rx and the fifth character is in the high-order byte of Rx+1. Set the three unused bytes of Rx+1 to binary zeroes. If the langid is less than five characters, it should be padded on the right with blanks as required.

Ry
Contains the function code for SAVE (X'00') in the high-order byte.

Ry+1
Contains the virtual address where you have loaded the CP message repository to be saved. The virtual address specified must start on a 4K page boundary.

Return Codes: When processing of the SAVE function completes, the low-order byte of Ry contains one of the following return codes:

Code Meaning and Action

X'00' The entire CP message repository is successfully saved.

If the return code is anything besides X'00', the CP message repository was not saved.

X'04' The DMKSNT entry for the language specified does not exist. Specify the appropriate NAMELANG macro in DMKSNT for this language.

X'08' The volid specified in the DMKSNT entry for the language is not a CP-owned volume. Ensure a CP-owned volume is specified in the DMKSNT entry generated by the NAMELANG macro for this language.

X'0C' The volid specified in the DMKSNT entry for the language is not mounted. The operator must mount this volume.

X'10' The repository is too large to be saved in the area reserved on DASD. The compiled listing gives the number of pages for the repository; the NLSPGCT parameter in NAMELANG must specify a page count greater than or equal to that number.

X'14' A paging error occurred during the save operation. The previous repository may be invalid.

X'18' An error occurred while attempting to write a page of the repository to DASD. The previous repository may be invalid.

X'1C' The langid specified with the DIAGNOSE does not match the langid in the repository you want to save. Either the wrong text deck was loaded into virtual storage, or the wrong langid was specified on the DIAGNOSE instruction.

X'20' The message repository is invalid. The text loaded into virtual storage to be saved is not the message repository.

Program Exceptions: If DIAGNOSE code X'CC' is specified incorrectly, the following exceptions are received:

Addressing
If the area to be saved extends beyond the user's virtual storage.

Privileged Operation
If the user does not have the privilege class required to issue the DIAGNOSE instruction.

Protection
If the area to be saved is fetch protected.

Specification
If the input registers are set up incorrectly for the DIAGNOSE or if the message repository to be saved is not found on a 4K page boundary.

DIAGNOSE Code X'D0' -- Provide 3480 Tape Volume Serial Number

All privilege classes (except ANY)

DIAGNOSE code X'D0' allows any virtual machine to provide CP with the virtual device address and the volume serial (VOLSER) of a 3480 volume. The VOLSER is then recorded in the OBR or MDR when an OBR or MDR is logged for the tape device.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'D0':

Rx
Contains the address of the tape volume serial. This number must be 6 bytes long and not cross a page boundary.

Ry
Contains the virtual device address.

Condition Codes: A condition code is returned for DIAGNOSE X'D0' as follows:

Condition Code	Meaning
0	DIAGNOSE complete.
1	Error condition. CP will not save the VOLSER for the specific tape volume. Ry contains one of the following return codes:

Ry=1	Invalid VOLSER address pointer. The pointer points to storage outside of the user defined storage area.
Ry=2	VOLSER crosses the page boundary
Ry=3	I/O error during paging
Ry=4	VOLSER is fetch protected
Ry=5	Invalid virtual device address or device is not dedicated
Ry=6	Dedicated device address not a 3480 tape device

Example: The following example shows how to invoke DIAGNOSE code X'D0':

```
DMKXXX          CSECT
                .               .
                .               .
                .               .
                LA              R5,VOLSER
                L               R6,X182
                DC              X'835600D0'
                .               .
                .               .
                .               .
VOLSER          DC              CL6'V00002'
                DS              0F
X182            DC              XL4'182'
                .               .
                .               .
                .               .
                END
```

DIAGNOSE Code X'D4' -- Specify An Alternate Userid

Privilege class B

DIAGNOSE code X'D4' lets a class B "master" virtual machine tell CP the userid of a "worker" machine that is performing the work and the userid of the "end-user" for which it is authorized to work. The end-user's userid is considered to be the "alternate userid."

CP uses the alternate userid in the following ways:

- Placed in the IPVMID field of the APPC/VM connection pending interrupt data when the "worker" issues an APPC/VM CONNECT. (See the *VM/SP Transparent Services Access Facility Reference* for more information on APPC/VM.)
- Used as the spool file origin id for spool files created by the worker.

Essentially, this lets the "worker" imitate or act on behalf of the "end-user" in the ways specified above.

The class B virtual machine must

- Be on the same system as the worker machine, but not necessarily on the same system as the end-user.
- Guarantee the identity of a remote user.
- Provide the userid of the end-user. CP cannot verify the end-user's id.
- Use this DIAGNOSE to set and reset the identity of the end-user for whom the worker machine is performing. When the worker machine is finished, the master machine can reset the alternate userid by issuing DIAGNOSE code X'D4' with the alternate userid set to zero.

Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'D4':

Rx
Contains binary zero.

Ry
Contains the virtual address of a 16-byte parameter list. The parameter list must not cross a page boundary. The 16-byte parameter list is divided as follows:

Bytes	Contain:
0 - 7	The userid of the worker machine to be authorized to access resources on behalf of an end-user. It must be left-justified and padded with blanks.
8 - 15	The userid of the end-user (alternate userid). It must be left-justified and padded with blanks. If the entire field is set to binary zeros, then the alternate userid is cleared; thus there is no alternate userid.

Return Codes: When DIAGNOSE code X'D4' processing completes, one of the following return codes is placed into Rx by CP:

Return Code	Meaning
0	The operation is successful.
4	There is an I/O error while paging in the parameter list.
8	The VMBLOK of the worker machine is not found.
12	The operation does not have RACF authorization.

Programming Exceptions: The user of DIAGNOSE code X'D4' receives the following exceptions:

Addressing
If the parameter list address specified is invalid.

Protection
If the parameter list is fetch protected against the user.

Specification
If the parameter list crosses a page boundary or Rx is not zero on entry.

Note: Nothing prevents the "master" and the "worker" from being the same virtual machine, but in this case, a security exposure may exist.

DIAGNOSE Code X'D8'—System Spool Information

Privilege Class D

Applying to VM/SP HPO only, DIAGNOSE code X'D8' allows the spooling operator to access SFBLOKs regardless of which system queue they are on (reader, printer, or punch). Either the system or a given user's spool file chains can be used.

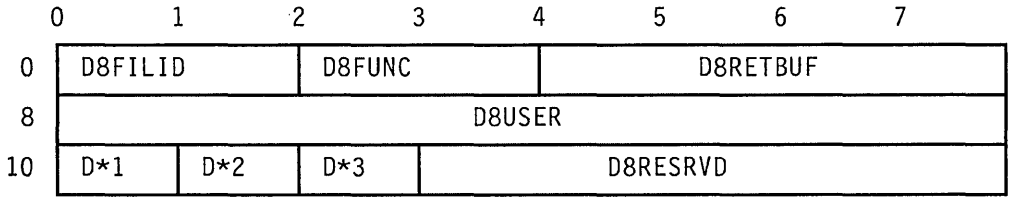
Entry Values: Set up the input registers as follows when invoking DIAGNOSE code X'CC':

Rx
Contains the doubleword-aligned virtual storage address of a parameter list (D8PARMS). The entire parameter list must be within a 4K page. The subfunction entry in this parameter list specifies the request being initiated. The subfunctions and their codes are:

Code	Subfunction
0000	Read next SFBLOK

Ry
Not used.

The parameter list pointed to by Rx has the following format:



where:

D8FILID
The spool file ID of the previous file. If D8FILID is 0, the first SFBLOK on the specified queue is returned.

D8FUNC
The subfunction code.

D8RETBUF
The virtual address of a buffer where the requested information is to be returned. The size of the buffer is defined by D8BUFSZ in doublewords. If the buffer is larger than the requested information, it is padded with zeroes.

D8USER
The virtual machine ID that owns the previous spool file (specified in D8FILID). If D8USER is 0 or blank, and D*FILID is non-zero, a specification exception is returned.

D*1
D8QUEUE contains one of the following flags (only one can be specified, otherwise a specification exception is returned):

D8PRINT (X'80') - Return the next print SFBLOK
D8PUNCH (X'40') - Return the next punch SFBLOK
D8READ (X'20') - Return the next reader SFBLOK

D*2
D8BUFSZ contains the size of the buffer in doublewords whose address is specified in D8RETBUF. If 0 is specified, 20 doublewords are returned. The specified amount of the SFBLOK is copied into the virtual machine's response buffer if a value other than 0 is used.

D*3
D8FLAG contains the following flag:

D8ANYVM (X'80') - Return next SFBLOK regardless of owning virtual machine. If this bit is off, the next file of the specified type (printer, punch, or reader) that belongs to the virtual machine specified in D8USER is returned.

D8RESRVD
Reserved. If this field is not 0, a specification exception is returned.

Exit Values: The contents of the SFBLOK is returned in the buffer specified by D8RETBUF.

Condition Codes:

Condition Code	Meaning
0	Data transfer successful
1	End of chain
2	Input file not found (or D8QUEUE is empty)

Program Exceptions: If DIAGNOSE code X'D8' is specified incorrectly, the following exceptions are received:

Specification
If one of the following errors was detected:

- The specified buffer or parameter list crosses a page boundary
- D8USER, D8FILID, and D8ANYVM are all zero
- D8USER is zero and D8FILID and D8ANYVM are not zero
- D8USER and D8ANYVM are both zero and D8FILID is not zero
- The subfunction (D8FUNC) is not defined

- The parameter list is not on a doubleword boundary
- More than one type of SFBLOK is specified in D*1
- D8RESRVD is not zero.

Subcode X'0000' -- Read next SFBLOK

If D8USER is 0 or blank and D8FILID is 0, the first SFBLOK on the specified input chain (system or user) is copied into the issuer's buffer. If a user ID and spool ID are specified, the queue specified by D8QUEUE is scanned for the specified file. If the SFBLOK was found, the next SFBLOK meeting the input criteria is copied into the issuer's buffer. If the file specified by D8USER and D8QUEUE is not found, the condition code is set to 2. If the file specified by D8USER and D8QUEUE is the last file meeting the input criteria, the condition code is set to 1 and no data is transferred.

Chapter 2. Inter-User Communications Vehicle

The Inter-User Communications Vehicle (IUCV) is a communications facility that allows a program running in a virtual machine to communicate with other virtual machines, with a CP system service, and with itself.

An IUCV communication takes place between a source communicator and a target communicator. The communication takes place over a predefined linkage called a path. Each communicator can have multiple paths, and can receive or send multiple messages on the same path simultaneously.

IUCV provides functions, through the IUCV macro instruction, to:

- Create and dismantle paths
- Send and reply to messages
- Receive or reject messages
- Control the sequence of IUCV events.

Communicators receive information about IUCV events by handling IUCV external interrupts.

Note: Advanced Program-to-Program Communication/VM (APPC/VM) is based on the IUCV support described in this chapter. It lets users connect to resource managers by specifying resource ids, instead of virtual machine userids and nodeids. Unlike IUCV, the resource, controlled by the target virtual machine, can be at the local system or, if the TSAF virtual machine is running, at any other system within a TSAF collection (a defined group of VM systems). For a complete description of how to use the APPC/VM facility refer to the *VM/SP Transparent Services Access Facility Reference.*

IUCV Paths

The IUCV directory control statement authorizes the establishment of paths between virtual machines, or between a virtual machine and a CP system service. If the maximum number of paths is not specified by the MAXCONN keyword of the OPTION statement in the user's directory, a communicator can establish a maximum of four paths.

Once authorized, users establish a path when the source communicator invokes the CONNECT function and the target communicator invokes the ACCEPT function. Either communicator can terminate an established path via the SEVER function. The target communicator can also prevent the establishment of a path by invoking the SEVER function instead of the ACCEPT function. In addition, communication over a path can be temporarily suspended when a communicator invokes the QUIESCE

function. The quiesced path can be reactivated when a communicator invokes the RESUME function.

A single communicator can have multiple paths defined, and virtual machines may have multiple paths between them. The communicator could be a source communicator on some of its defined paths, a target communicator on other paths, and both a source and a target communicator on still other paths. Communication over any and all paths can occur simultaneously.

Every path has two ends: the source communicator's end and the target communicator's end. The source communicator has a description of the path from the source's perspective and the target communicator has a description of the same path from the target's perspective.

Each path description has a path id that is unique for each communicator. IUCV assigns path ids when communicators invoke the CONNECT and ACCEPT functions. When invoking IUCV functions, the source communicator identifies the path by using the source's path id. The target communicator identifies the same path to IUCV by using the target's path id. A pathid is IUCV's method of distinguishing among the paths available to a communicator.

IUCV Messages

An IUCV communication is called a message. The source communicator invoking the SEND function initiates communication and creates a message. The target communicator obtains the message by invoking the RECEIVE function.

The target communicator can optionally request information about messages sent to it by invoking the DESCRIBE function, and can refuse a message sent to it by invoking the REJECT function. The target communicator can respond to a message via the REPLY function.

Communication is terminated and the message is destroyed when the source communicator issues the TEST COMPLETION function or handles an IUCV message complete external interrupt.

Message Data Transfer

When the target communicator issues the RECEIVE function, IUCV moves the message data from the source communicator's SEND virtual address space to the target communicator's RECEIVE virtual address space. When the target communicator issues the REPLY function during a two-way communication, IUCV moves data from the target communicator's REPLY virtual address space to the source communicator's ANSWER virtual address space.

Figure 6 on page 113 illustrates the movement of message data during an IUCV two-way communication.

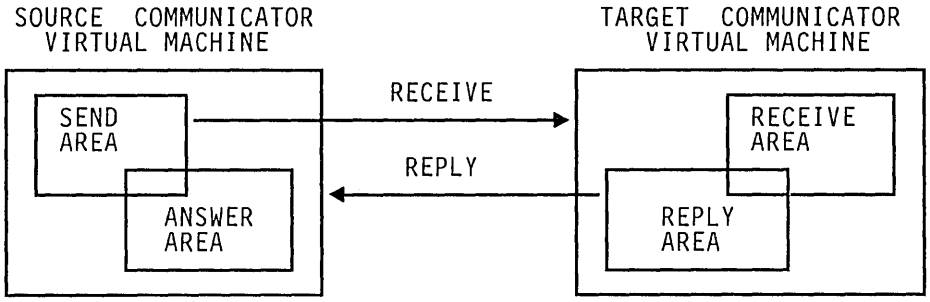


Figure 6. IUCV Two-Way Data Transfer

The source communicator's SEND and ANSWER areas may overlap. Similarly, the target communicator's RECEIVE and REPLY areas may overlap.

CP performs storage protection checking for all data moved during an IUCV communication.

Message Identification

A message is fully identified to a virtual machine by three values. IUCV functions allow one or more of these values to be specified to selectively process messages.

- Message id

  IUCV assigns a message id when the source communicator invokes the SEND function. The message id is generated by a sequential counter value and is unique for the system IPL.

- Message class

  The source communicator identifies a message by using the source message class and target communicator identifies a message by using the target message class. The message classes are arbitrary values that the source communicator specifies when invoking the SEND function. The meaning of the message classes is agreed to in advance by the two communicators. IUCV places no restrictions on the values specified for message class. The communicators can use the message class to handle messages selectively.

- Path id

  IUCV assigns the path id when a path is established with the CONNECT function.

There is no defined relationship between the values of the source and target path ids IUCV assigns, or between the message classes the source and the

target communicators use. None of these values need to be the same although they refer to the same message.

The message id always has the same value for both target and source communicators.

When invoking IUCV functions, the source communicator may refer to a message by a combination of its source path id, source message class, and message id. The target communicator may refer to the same message by a combination of its target path id, target message class, and message id.

The message tag information may optionally be used by the source communicator to further identify a message. Since IUCV presents the tag to the source communicator when the message completes, the tag may be used to tie the completed message to the original SEND request.

Since a message can be identified as a priority message, the source communicator may also use this as an indication to the target communicator that special handling is required. IUCV queues a priority message ahead of any nonpriority messages and behind any earlier priority messages. A communicator must be authorized to handle priority messages in the IUCV directory control statement.

IUCV External Interrupts

The IUCV external interrupt notifies a virtual machine about IUCV events.

To enable IUCV external interruptions, communicators must:

- Invoke the DECLARE BUFFER function to indicate to IUCV where to store data associated with an external interruption.
- Set bit 7 in the virtual machine's PSW to one.
- Set submask bit 30 of control register 0 to one.

IUCV functions generate a type X'4000' external interruption. When a virtual machine in EC mode receives an IUCV external interruption, IUCV places the interruption code in locations X'86' and X'87' of the virtual machine's storage. For a virtual machine in BC mode, IUCV places the code in the external old PSW. In addition, IUCV stores an external interrupt buffer containing information about the message or IUCV function at the address specified when the communicator invoked the DECLARE BUFFER function. One field of this buffer is an external interrupt subtype that indicates why the external interrupt occurred. The possible values of this field are:

- 01 - Connection pending
- 02 - Connection complete
- 03 - Connection severed
- 04 - Connection quiesced

- 05 - Connection resumed
- 06 - Priority message completion
- 07 - Nonpriority message completion
- 08 - Priority message pending
- 09 - Nonpriority message pending

The first five types are called "control interrupts", and the last four types are called "message interrupts".

Control interrupts are always reflected to the virtual machine in first-in-first-out (FIFO) order before message interrupts. Message interrupts of the same subtype are reflected in first-in-first-out (FIFO), but message interrupts of different subtypes are reflected in the order shown above.

A virtual machine can use the SET MASK function to enable or disable external interrupts selectively for IUCV communications. The SET MASK function has mask bits that enable or disable external interruptions for:

- Priority message pending
- Nonpriority message pending
- Priority message completion
- Nonpriority message completion
- IUCV control functions

To divide and handle the control type interrupts even further, the SET CONTROL MASK function may be used on the IUCV macro. The types of control interrupts may be separately enabled and disabled. These control type interrupts are:

- Connection pending
- Connection complete
- Connection severed
- Connection quiesced
- Connection resumed

The SET MASK function is interrogated before the SET CONTROL MASK function. If you specify that all control interrupts are disabled using the SET MASK function, then the SET CONTROL MASK settings are not interrogated. If you specify that all control interrupts are enabled using the SET MASK function, then the SET CONTROL MASK settings are interrogated to determine how to handle the individual types of control interrupts.

After IUCV initialization and until you issue the SET MASK or SET CONTROL MASK functions, all IUCV submask bits are on, enabling all IUCV external interrupts.

Avoiding IUCV External Interrupts

A virtual machine can only be notified about an IUCV control function by receiving an external interruption. However, a virtual machine can handle pending messages either by an external interrupts or by using the DESCRIBE function. Message completions can be handled either by an external interrupt or with the TEST COMPLETION function.

IUCV also provides the TEST MESSAGE function to determine the presence of any pending messages or message completions. If neither is pending, the virtual machine goes into a wait state until one is pending.

For example, if a source communicator sends a priority message, IUCV queues an external interrupt for the target communicator. If the target virtual machine is enabled for external interrupts, then the target virtual machine receives an external interrupt. However, if the target virtual machine is not enabled, the message remains pending for the virtual machine, and the target virtual machine can issue the DESCRIBE function to obtain information about the message in the parameter list. The message pending external interrupt is cleared. The target virtual machine can continue processing the message with the RECEIVE or REJECT functions.

Note: If a communicator is enabled for external interrupts and issues the DESCRIBE or TEST COMPLETION function, results are unpredictable. It can not be determined whether information about a particular message is received via external interrupt or by the completion of DESCRIBE or TEST COMPLETION. However, IUCV supplies information about a message only once.

Two IUCV functions, QUIESCE and RESUME, let a virtual machine control the arrival of message pending external interrupts. The QUIESCE function suspends incoming messages on one or all IUCV paths. Any communicator trying to send a message over a path that has been quiesced receives a return code indicating a quiesced path. No message is created and thus no external interrupt is reflected. The RESUME function restores normal communications.

Security Considerations

Installations control the use of IUCV through the virtual machine directory entries. If the installation has not authorized a user for IUCV communications in the directory, all requests for IUCV communications to virtual machines other than the user's own are denied. Service virtual machines and CP system services defined with the ALLOW (any virtual machine to connect) option do their own authorization checking, and individual directory entries are not needed.

IUCV moves data from one virtual machine address space to another. A virtual machine never has access to the storage or registers of CP or another virtual machine. When the user invokes the RECEIVE or REPLY

functions, the data to be moved is described by a starting address and a length, or a list of starting addresses and lengths. The length specified in the parameter list is the maximum amount of data moved. No requirements are placed on a virtual machine as to the location of these buffers.

IUCV assigns path ids and records the path id of each communicator. A given communicator can reference only the paths that have been established for his virtual machine.

IUCV assigns the message id for each message. IUCV does not use this identifier as a direct reference, but only as an operand in a comparison. It is conceivable that a virtual machine could generate a valid message identifier and use this to request a message. However, when a message id is used to request a message, a user must also specify a message class and a path id. If the specified message is not associated with the specified path id and message class, the user cannot access the messages. If the message id, path id, and message class do match, the user could legitimately access the message by specifying simply path id and/or message class without the generated message id.

The installation can limit the number of connections for a particular virtual machine by using the MAXCONN parameter of the OPTION control statement in the virtual machine's directory entry.

Virtual Machine-to-Virtual Machine Communication

Using Data in a Buffer

Figure 7 on page 118 illustrates a typical sequence of functions invoked when a virtual machine communicates with another virtual machine. The functions include initializing, connecting to another virtual machine, sending and receiving messages, replying to and waiting for messages, severing communications with the other virtual machine, and terminating communications.

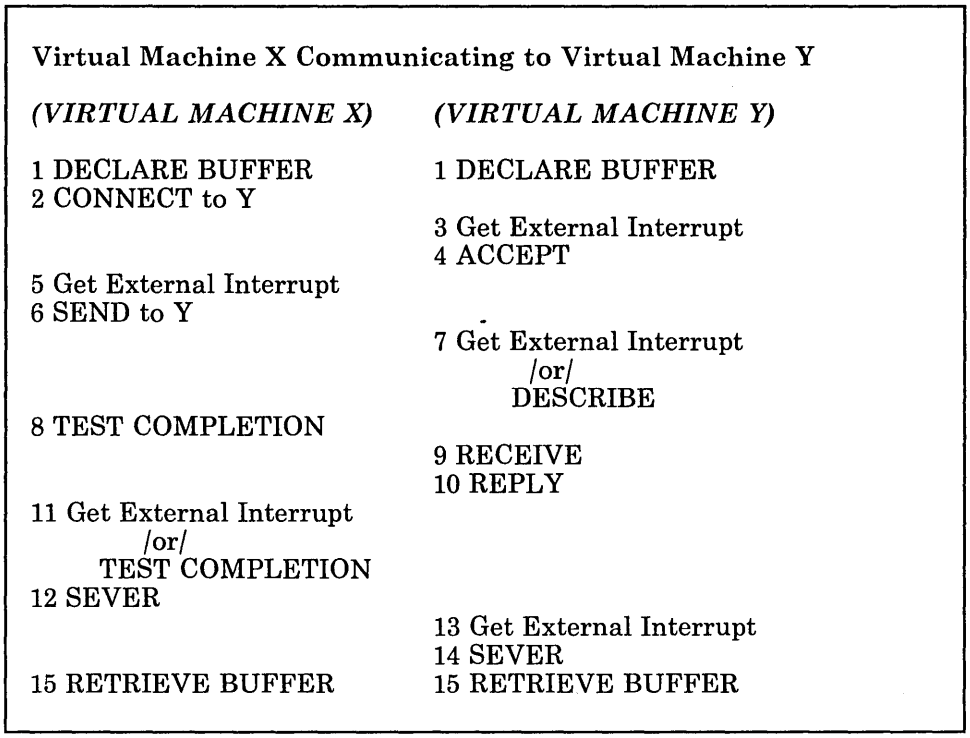


Figure 7. Sequence of Functions

1. Virtual machine X wishes to communicate with virtual machine Y. Both virtual machines must independently invoke the DECLARE BUFFER function. The buffer provides the virtual machine with information about incoming external interrupts concerning IUCV functions.

2. Virtual machine X invokes the CONNECT function, indicating Y as the target. IUCV checks the directory to determine if this connection is authorized. If it is, IUCV queues an external interrupt for Y indicating that there is a connection pending for it. IUCV returns control to X at the next instruction after the CONNECT.

3. The external interrupt queued by step 2 is reflected to Y indicating a connection pending. IUCV places the external interrupt information in the buffer that Y provided in step 1. IUCV passes control to the external interrupt handler of Y.

4. Virtual machine Y interprets the external interrupt and responds with an ACCEPT to complete the connection. IUCV then completes the connection and queues a Connection Complete external interrupt for X. IUCV returns control to Y at the next instruction after the ACCEPT.

5. The external interrupt queued by step 4 is reflected to X, indicating that the connection is complete, and the communication path is available for use. IUCV places the external interrupt information in the buffer that X provided in step 1. IUCV passes control to the external interrupt handler of X.

6. Virtual machine X issues a SEND. The SEND function queues an external interrupt for Y indicating that a message is pending. Control returns in X at the next instruction after the SEND.

7. If virtual machine Y is enabled for external interrupts and for IUCV messages (via SET MASK), the external interrupt queued by step 6 is reflected to Y, indicating that a message is pending. IUCV places external interrupt information in the buffer specified in step 1. IUCV passes control to the external interrupt handler of Y. If virtual machine Y is disabled for external interrupts or IUCV messages and invokes the DESCRIBE function, IUCV places the message information in the DESCRIBE parameter list, and the Message Pending external interrupt for this message is cleared. IUCV passes control to the next instruction after the DESCRIBE.

8. While virtual machine Y is processing the message, virtual machine X can decide to check if the communication has been completed by issuing the TEST COMPLETION function. The condition code indicates that (in this example) the communication is not complete.

9. With the message description from step 7, virtual machine Y starts processing the message and issues a RECEIVE. The parameter list associated with RECEIVE specifies where the message data is stored in virtual machine Y.

   If the message was one-way, the RECEIVE function queues an external interrupt for X indicating that the message had completed. REPLY processing in step 10 would not be required for one-way messages. Control returns to Y at the next instruction after the RECEIVE.

10. When processing the message is complete, virtual machine Y responds to X by invoking the REPLY function. The REPLY function queues an external interrupt for X indicating that the message has completed. Control returns to Y at the next instruction after the REPLY.

11. If virtual machine X is both enabled for external interrupts and enabled for IUCV replies, the external interrupt queued by step 10 is reflected to X, indicating a reply pending. To identify the reply, the external interrupt information is placed in the buffer specified in step 1. IUCV passes control to the external interrupt handler of X. If virtual machine X is disabled for external interrupts and issues a TEST COMPLETION, IUCV places the message information in the TEST COMPLETION parameter list, and the Message Completion external interrupt is cleared. IUCV passes control to the next instruction after the TEST COMPLETION.

12. Virtual machine X has now completed its communications with virtual machine Y and issues a SEVER to break the communications path. The SEVER function queues an external interrupt for Y indicating that the communication link has been broken. Control returns in X at the next instruction after the SEVER.

13. The external interrupt queued by step 12 is reflected to Y indicating that the path has been broken by virtual machine X. Virtual machine Y can now do any clean up needed in its storage.

14. After virtual machine Y has completed processing, the virtual machine issues a SEVER notifying IUCV that it also is finished with the communication path. IUCV can then clean up its control blocks.

15. When all communications are complete and all communication paths have been severed, both virtual machines independently invoke the RETRIEVE BUFFER function.

Using Data in a Parameter List

Most IUCV functions require a **parameter list** which contains information necessary for IUCV to perform the requested function. The IUCV macro assists you in filling in the parameter list properly. The parameters used with each function are described with the individual function descriptions later in this chapter.

The parameters let you specify eight bytes of data in the parameter list. To understand better how data specified in the parameter list is handled, the IUCV functions are covered in a typical scenario.

1. The IUCV DECLARE BUFFER, CONNECT, and ACCEPT sequence must be invoked to establish the user's external interrupt buffer and a path to the target virtual machine (or CP). If you expect to receive data in the parameter list, you must authorize such communication on the CONNECT or ACCEPT by specifying PRMDATA=YES. The external interrupt information to the target communicator includes a bit indicating if PRMDATA=YES was chosen.

2. Issue an IUCV SEND request. When the data is to be passed in the parameter list, the DATA=PRMMSG option is used on the IUCV macro, and the PRMMSG= option is used to move the data into the parameter list. The sender of the message should be prepared to handle a return code indicating that DATA=PRMMSG is not allowed if the target communicator has not specified PRMDATA=YES at connection time. IUCV saves the message data until it is to be presented to the target.

3. If the target is enabled for IUCV Message Pending external interrupts, the target virtual machine receives an IUCV Message Pending external interrupt because of the SEND request in the previous step. The message data is stored in the external interrupt buffer. A flag is set in the IPFLAGS1 field of the buffer indicating that the data is in the parameter list. Since the message data has been presented to the target, the target does not have to issue an IUCV RECEIVE for this message. If the message was a one-way message, communication is complete. There is no asynchronous return of message completion given to the source (sending) virtual machine on a one-way message.

4. If the target is disabled for IUCV Message Pending external interrupts and issues the IUCV DESCRIBE or RECEIVE functions, the message data is stored in the parameter list. A flag is set in the IPFLAGS1 field of the parameter list indicating that the data is in the parameter list. Since the message data is presented to the target on a DESCRIBE, the target does not have to issue an IUCV RECEIVE for this message. If the message was a one-way message, the communication is complete. There is no asynchronous return of message completion given to the source (sending) virtual machine on a one-way message.

5. If the communication in the previous steps was a two-way message, a REPLY is issued by the target virtual machine. When the REPLY data is to be passed in the parameter list, the DATA=PRMMSG option is used on the IUCV macro, and the PRMMSG= option is used to move the data into the parameter list. The REPLYer of the message should be prepared to handle a return code indicating that DATA=PRMMSG is not allowed if the source communicator has not specified PRMDATA=YES at connection time. IUCV saves the message data until it is to be presented to the source communicator.

6. If the source communicator is enabled for IUCV Message Completion external interrupts, the source virtual machine receives an IUCV Message Completion external interrupt because of the REPLY in the previous step. The message data is stored in the external interrupt buffer. A flag is set in the IPFLAGS1 field of the buffer indicating that the data is in the parameter list. The communication is complete.

7. If the target is disabled for IUCV Message Completion external interrupts, and issues the IUCV TEST COMPLETE function, the message data is stored in the parameter list. A flag is set in the IPFLAGS1 field of the parameter list indicating that the data is in the parameter list. The communication is complete.

8. SEVER and RETRIEVE BUFFER cause any messages pending to be destroyed for that virtual machine. Since no asynchronous Message Completion external interrupt is returned to the source communicator for one-way messages using the DATA=PRMMSG option, the source communicator must realize upon receiving an IUCV Connection Severed external interrupt from the target communicator that messages may not have been received by the target.

Using Control Paths

IUCV control paths and buffers allow a control program (like CMS) running in a virtual machine to use the IUCV functions without interfering with a user application that is also using IUCV. Applications would not be coded using the CONTROL parameter on the IUCV DECLARE BUFFER and CONNECT functions.

To understand better how control paths would be handled, the IUCV functions are covered in a typical user scenario. In the scenario, CMS is

used as the control program running in a virtual machine executing a normal IUCV application.

1. When CMS is IPLed in the virtual machine, CMS issues an IUCV DECLARE BUFFER with the CONTROL=YES parameter. This establishes a control buffer for CMS to use. All IUCV external interrupt information for control paths is presented in this buffer.

2. After CMS has defined a control buffer, CMS may establish control paths to other virtual machines by issuing an IUCV CONNECT with the CONTROL=YES parameter. All paths used by CMS should be specified as control paths.

3. If the target virtual machine accepts the connection request, an IUCV Connection Complete external interrupt is presented to the CMS control program. The IPCNTRL bit in IPFLAGS1 of the external interrupt indicates that a control path was accepted. CMS may now start communications on this path.

4. When CMS allows the application program to run, the application issues an IUCV DECLARE BUFFER with the CONTROL=NO parameter. All application paths are established using IUCV CONNECT with the CONTROL=NO parameter. These are the functions that the application uses today so no changes are required to the application.

5. The application starts communicating over its established paths.

6. Since both CMS and its application have established paths, both are expecting and handling external interrupts.

   If an external interrupt is on a control path, the IUCV information about the interrupt is stored in the control buffer when the interrupt is presented to CMS. CMS interrogates the control buffer, recognizes the path id as belonging to a control path, and handles the IUCV interrupt. The application's buffer remains unchanged.

   If the external interrupt is on an application path, the IUCV information about the interrupt is stored in the application's buffer when the interrupt is presented to CMS. Since CMS only has access to the control buffer, IUCV stores the path id in the control buffer and clears (to zero) the remainder of the buffer. CMS interrogates the control buffer, recognizes the path id as belonging to an application path, and passes the IUCV external interrupt to the application for handling.

7. When the application wishes to terminate a path or all IUCV communications, it uses the IUCV SEVER or RETRIEVE BUFFER functions. Neither of these functions affect the control paths being used by CMS.

Certain IUCV functions result in an operation exception if executed with only a control buffer declared. These functions are:

- ACCEPT
- RETRIEVE BUFFER
- TEST MESSAGE
- DESCRIBE
- TEST COMPLETION
- SET MASK
- SET CONTROL MASK.

The ALL=YES parameter on the IUCV functions of SEVER, QUIESCE, and RESUME does not affect control paths.

When handling IUCV messages with the IUCV functions of RECEIVE, REPLY, REJECT, and PURGE on control paths, the message must be fully qualified. The message id, path id, and class of the message must be specified in the parameter list to reference the message.

The IUCV functions affecting IUCV external interrupts do not operate on interrupts for control paths. These functions are TEST MESSAGE, DESCRIBE, TEST COMPLETION, SET MASK, and SET CONTROL MASK. These functions are never used by a control program since they have no affect on control paths.

Since IUCV cannot tell what part of the virtual machine issued an IUCV function, it is possible for an application to issue an IUCV function on a control path. This reference to a control path by an application, whether intentional or accidental, is considered a user application error. For example, the SEVER function specifying a control path terminates that path even though the function was issued by the application program.

Invoking IUCV Functions

You can invoke all IUCV functions through the IUCV macro. In general, specify the name of the IUCV function you wish to perform, the address of a parameter list to contain input to the function, and keyword parameters. IUCV moves the values specified on the keyword parameters into the specified parameter list.

The parameter list must be defined on a doubleword boundary.

You can specify IUCV parameters in two ways:

- By coding keyword parameters on the IUCV macro. IUCV stores values in the parameter list based on values you specify on the macro.
- By storing required input to the function in the function parameter list before invoking the IUCV macro instruction. To store input in an IUCV parameter list, use labels generated by the IPARML DSECT.

You may use a combination of these methods to supply input to a single IUCV function. If you specify any optional parameters on the IUCV macro, you are responsible for providing the USING for the IPARML DSECT when the macro is invoked. If you do not specify an optional parameter to initialize the parameter list, the macro assumes that you have stored a value in the parameter list before invoking the IUCV macro.

Note: The IUCV macro does NOT clear parameter list fields since values may have been already stored by the user. Therefore, it is the user's responsibility to insure that all unused fields are cleared (set to zero). All reserved fields in the parameter list should always be set to zero.

An advantage of using the IUCV macro instruction is that IUCV provides extensive error checking of parameter combinations when input is supplied on the macro. Many invalid parameter combinations can be detected by IUCV when you assemble the program.

The IUCV macro, after formatting the parameter list with any optional keyword parameters, generates two assembler instructions.

- A load address (LA) instruction to put the IUCV function code into register zero (0).
- An IUCV instruction (B2F0xxxx) which indicates the IUCV parameter list.

IUCV Functional Descriptions

In the functional descriptions the following terms are used:

Address
: A guest real address (real to the virtual machine). It can be specified on the IUCV macro as either the

  1. Label of the storage location, or
  2. Number of a register in parentheses that contains the address, "(reg)".

Address List
: A virtual machine defined area used on an IUCV SEND, RECEIVE, or REPLY that allows data to be moved from discontiguous areas. The address list must be on a doubleword boundary in the following format:

address 1	length 1
address 2	length 2
.	.
address n	length n

Each entry contains two fullwords; the address of the data to be transferred and the number of bytes to be transferred from that address.

Label — An addressable label in the user's program. The IPARML DSECT provides common labels for referencing fields in an IUCV parameter list.

Length — The amount of data to be transferred on an IUCV request. It can be specified on the IUCV macro as either the

1. Label of the storage location containing the length, or
2. Number of a register that contains the length, "(reg)".

The IUCV macro assumes a halfword value for the length at the storage location, or the low-order halfword of the register specified. A length modifier of 2 or 4 may be used, (label,2) or ((reg),2), or (label,4) or ((reg),4). If a length modifier of 4 is used, the macro uses the fullword value for the length at the storage location or in the register specified.

The functional descriptions are presented in the following groupings.

Basic communication functions:

QUERY	- Get IUCV information
DECLARE BUFFER	- Initialize for IUCV communications
CONNECT	- Establish a path
ACCEPT	- Complete a path
SEND	- Transmit a message
RECEIVE	- Receive a message
REPLY	- Respond to a message
REJECT	- Refuse a message
PURGE	- Cancel a message
SEVER	- Terminate a path
RETRIEVE BUFFER	- Terminate all IUCV communications

Controlling IUCV external interrupts:

QUIESCE	- Suspend message pending interrupts
RESUME	- Restore message pending interrupts

TEST MESSAGE	- Check for interrupts or wait
DESCRIBE	- Avoid Message pending interrupt
TEST COMPLETION	- Avoid Message complete interrupt
SET MASK	- Disable all types of IUCV interrupts
SET CONTROL MASK	- Disable all IUCV control interrupts

QUERY Function

The QUERY function determines how large an external interrupt buffer IUCV requires to store information, and determines the maximum number of communication paths you can establish in your virtual machine.

QUERY can be issued before DECLARE BUFFER to determine the buffer size and allocate the buffer before it is declared to IUCV. The maximum number of paths facilitates the allocation of a user-defined path table. This function is useful to virtual machines that have dynamic storage allocations routines. For those programs that must allocate fixed storage, the buffer size is 40 bytes (X'28'), and the maximum number of paths available would be the default of four or the number on the user's OPTION directory control statement, the MAXCONN option.

IUCV Macro Format

label	IUCV	QUERY

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 3 - Errors were encountered reading directory

PROGRAM INTERRUPTIONS
Operation Exception Your virtual machine is not in supervisor state.

Output from QUERY

When you invoke the QUERY function, IUCV returns:

- The size of the IUCV external interrupt buffer in general register 0.
- The maximum number of connections that can be outstanding for this virtual machine in general register 1.

DECLARE BUFFER Function

The DECLARE BUFFER function specifies the address of an external interrupt buffer where IUCV can store information. When a virtual machine receives an IUCV external interruption, IUCV stores in this buffer information about the message, reply, or control function that caused the the interruption.

The DECLARE BUFFER function must be invoked before any other IUCV function can be used (except QUERY).

After invoking the DECLARE BUFFER function and if enabled for IUCV interrupts, the virtual machine can now start receiving IUCV external interrupts.

IUCV Macro Format

label	IUCV	DCLBFR	,PRMLIST=	{address}
			,MF=	L
			,BUFFER=	{address}
			,CONTROL=	{YES|NO}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the DECLARE BUFFER parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

BUFFER=
specifies the address of the external interrupt buffer.

CONTROL=
specifies whether this buffer is to be used with control paths or application paths.

CONTROL=YES indicates that this buffer is to be used with control paths. For a complete discussion on using control buffers and paths, see "Using Control Paths" on page 121.

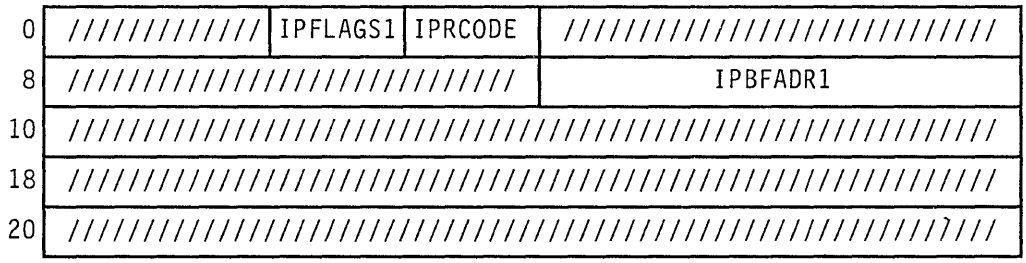
CONTROL=NO indicates that this buffer is to be used with application paths.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE 3 - Errors encountered in reading directory

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Addressing Exception** The parameter list or buffer address that you specified is outside the virtual machine's storage. **Operation Exception** Your virtual machine is not in supervisor state. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

DECLARE BUFFER Parameter List Format



Parameter List Input Fields

- IPFLAGS1 -- contains options for the CONNECT function.

  IPCNTRL (X'04') -- indicates that you want this to be a control buffer.

- IPBFADR1 - The address of your external interrupt buffer.

Output from DECLARE BUFFER

RETURN CODES in IPRCODE
0 - Normal return 19 - A previously declared buffer is still in use

CONNECT Function

The CONNECT function establishes an IUCV path to another virtual machine. Although the CONNECT may complete successfully, you are not able to use the path until you receive an IUCV Connection Complete external interrupt (the target has ACCEPTed your connection) for this path.

If you receive an IUCV Connection Severed external interrupt (the target has SEVERed your connection) for this path, you may not use this path since the connection has been refused by the target virtual machine.

If the CONNECT function completes successfully, the count of active connections is incremented for both virtual machines. If a virtual machine is connecting to itself, the active connection count is incremented by two. The count is not decremented until the virtual machine issues an IUCV SEVER for a particular path.

IUCV Macro Format

label	IUCV	CONNECT	,PRMLIST=	{address}
			,MF=	L
			,CONTROL=	{YES|NO}
			,MSGLIM=	{address}
			,PRMDATA=	{YES|NO}
			,PRTY=	{YES|NO}
			,QUIESCE=	{YES|NO}
			,USERDTA=	{address}
			,USERID=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the CONNECT parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

CONTROL=
specifies whether this connection is to be associated with the external interrupt buffer for control paths or application paths.

CONTROL=YES indicates that this path is to be associated with the external interrupt buffer for control paths. For a complete discussion

on using control paths and buffers, see "Using Control Paths" on page 121.

CONTROL=NO indicates that this path is to be associated with the external interrupt buffer for application paths.

MSGLIM=
specifies the limit of outstanding messages to be allowed on the path established by this CONNECT. The address of the MSGLIM is to a halfword value.

Upon executing the IUCV instruction, the message limit specified is checked to insure that the maximum limit of 255 has not been exceeded. The actual limit assigned to the path established by this connection is the lower of this value in the parameter list or the value on the IUCV directory control statement (if specified). If the message limit is not specified in the parameter list or on the IUCV directory control statement, IUCV assumes a default of 10. After executing the IUCV instruction, the IPMSGLIM field reflects the actual message limit established for the path.

If the SEND function completes successfully, the count of active messages is incremented for the sending virtual machine. The count is not decremented until the message complete external interrupt is returned. For one-way messages using data in the parameter list, the count is decremented when the message pending external interrupt is presented to the target virtual machine.

PRMDATA=
specifies whether your program can handle message data in the parameter list.

PRMDATA=YES indicates that your program can handle message data in the parameter list (those messages sent using the parameter DATA=PRMMSG on an IUCV SEND).

PRMDATA=NO indicates that your program can only handle message data presented in a buffer (sent using the parameter DATA=BUFFER on an IUCV SEND).

PRTY=
specifies if you want to send priority messages on this path. It does not affect the program's ability to receive priority messages.

PRTY=YES indicates that you want to send priority messages. Priority must be authorized on the IUCV directory control statement for this parameter to be effective. After executing the IUCV instruction, the IPPRTY bit in IPFLAGS1 should be checked to insure that priority messages were authorized.

PRTY=NO indicates that you do not want to send priority messages.

If your program is unauthorized or if PRTY=NO is specified, IUCV prevents your program from sending priority messages.

QUIESCE=
specifies whether you want to quiesce the path being established.

QUIESCE=YES prevents messages from coming across this path until your program is ready to process them. You can restore the path to full communication by invoking the IUCV RESUME function.

QUIESCE=NO indicates that the path is to become active as soon as the corresponding IUCV ACCEPT is done by the target communicator.

USERDTA=
specifies the data area containing the 16 bytes of user data that IUCV is to reflect to the target virtual machine. The user data is reflected as part of the IUCV Connection Pending external interrupt.

USERID=
specifies the 8-character userid of the target virtual machine or the IUCV system service to which you want to establish this path.

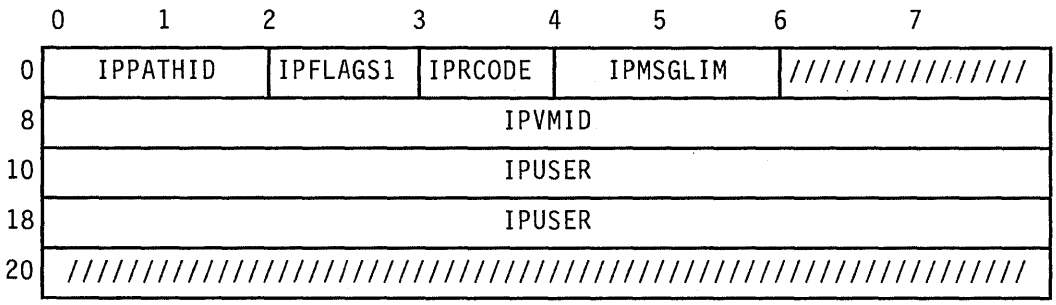
IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored in IPRCODE

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

IUCV CONNECT

CONNECT Parameter List Format



Parameter List Input Fields

- IPFLAGS1 -- contains options for the CONNECT function.

  IPCNTRL (X'04') -- indicates that you want this to be a control path.

  IPAPPC (X'08') -- indicates the protocol to be used on this path. This bit must be set to zero.

  IPPRTY (X'20') -- indicates that you want to send priority messages on this path.

  IPQUSCE (X'40') -- indicates that you do not want to receive messages on this path until an IUCV RESUME is issued.

  IPRMDATA (X'80') -- indicates that your program can handle message data in the parameter list.

- IPMSGLIM -- contains the limit of outstanding messages that IUCV is to allow on the path.
- IPVMID -- contains the userid of the virtual machine or IUCV system service to which you want to establish this path.
- IPUSER -- contains the user data that IUCV reflects to the target virtual machine.

Output from CONNECT

- IPPATHID -- contains the path id that IUCV assigns the new path.
- IPMSGLIM -- contains the message limit for this path.
- IPFLAGS1 -- contains specific information about this connection.

  IPPRTY (X'20') -- indicates that you may send priority messages on this path.

- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

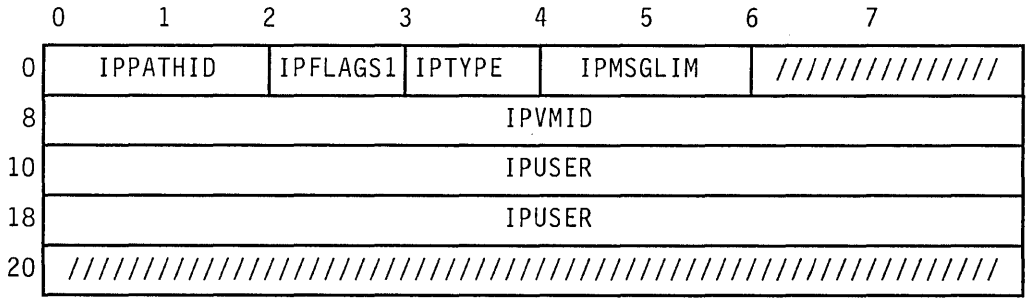
RETURN CODES in IPRCODE
0 - Normal return 11 - Target communicator is not logged on 12 - Target communicator has not invoked the DECLARE BUFFER function 13 - Maximum number of connections for this communicator exceeded 14 - Maximum number of connections for the target exceeded 15 - No authorization found 16 - Invalid IUCV system service name 18 - Value in IPMSGLIM exceeds 255

Connection Pending External Interrupt

To notify the target virtual machine that you wish to establish a new path (via the CONNECT), IUCV reflects an IUCV Connection Pending external interrupt to the target virtual machine.

The target virtual machine receives this external interrupt if it is enabled for IUCV interrupts in Control Register 0 and the PSW. The functions of SET MASK and SET CONTROL MASK also control the presentation of this type of interrupt.

The external interrupt contains the information that the target virtual machine needs to either ACCEPT or SEVER the pending connection.



- IPFLAGS1 -- contains options for this path.

  IPPRTY (X'20') -- indicates that the virtual machine may receive priority messages on this path.

  IPQUSCE (X'40') -- indicates that IUCV will not allow messages to be sent on this path until an IUCV RESUME is issued by the connecting virtual machine.

  IPRMDATA (X'80') -- indicates that the connecting virtual machine can handle message data in the parameter list.

- IPMSGLIM -- contains the maximum number of messages that IUCV allows the connecting virtual machine to send on this path.
- IPPATHID -- contains the path id that IUCV assigns the new path.
- IPTYPE -- indicates a Connection Pending external interrupt with a value of X'01'.
- IPUSER -- contains the user data specified by the virtual machine that wants to establish this path.

ACCEPT Function

The ACCEPT function is issued after the user receives a Connection Pending external interrupt and now wishes to complete the IUCV communication path.

IUCV Macro Format

label	IUCV	ACCEPT	,PRMLIST=	{address}
			,MF=	L
			,MSGLIM=	{address}
			,PATHID=	{address}
			,PRTY=	{YES|NO}
			,PRMDATA=	{YES|NO}
			,QUIESCE=	{YES|NO}
			,USERDTA=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the ACCEPT parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MSGLIM=
specifies the limit of outstanding messages to be allowed on the path completing this ACCEPT. The address of the MSGLIM is to a halfword value.

Upon executing the IUCV instruction, the message limit specified is checked to insure that the maximum limit of 255 has not been exceeded. The actual limit assigned to the path established by this connection is the lower of this value in the parameter list or the value on the IUCV directory control statement (if specified). If the message limit is not specified in the parameter list or on on IUCV directory control statement, IUCV assumes a default of 10. After executing the IUCV instruction, the IPMSGLIM field reflects the actual message limit established for the path.

PATHID=
specifies the path identification number on which you wish to communicate. This path id is presented to the virtual machine in the Connection Pending external interrupt.

PRMDATA=
specifies whether your program can handle message data in the parameter list.

PRMDATA=YES indicates that your program can handle message data in the parameter list (those messages sent using the parameter DATA=PRMMSG on an IUCV SEND).

PRMDATA=NO indicates that your program can only handle message data presented in a buffer (sent using the parameter DATA=BUFFER on an IUCV SEND).

PRTY=
specifies if you want to send priority messages on this path. It does not affect the program's ability to receive priority messages.

PRTY=YES indicates that you want to send priority messages. Priority must be authorized in the IUCV directory control statement for this parameter to be effective. After executing the IUCV instruction, the IPPRTY bit in IPFLAGS1 should be checked to insure that priority messages were authorized.

PRTY=NO indicates that you cannot send priority messages.

If your program is unauthorized or if PRTY=NO is specified, IUCV prevents your program from sending priority messages.

QUIESCE=
specifies whether you want to quiesce the path being established.

QUIESCE=YES prevents messages from coming across the path until your program is ready to process them. You can restore the path to full communication by invoking the IUCV RESUME function.

QUIESCE=NO indicates that the path will become active as soon as the IUCV ACCEPT completes.

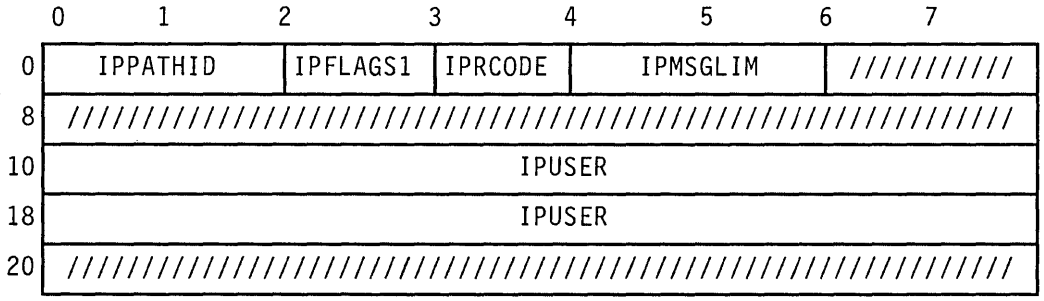
USERDTA=
specifies the data area containing the 16 bytes of user data that IUCV is to reflect to the source virtual machine. The user data is reflected as part of the IUCV Connection Complete external interrupt.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored in IPRCODE

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary.
Operation Exception The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state.
Addressing Exception The parameter list address that you specified is outside the virtual machine's storage.
Protection Exception The storage key of the specified parameter list address does not match the key of the user.

ACCEPT Parameter List Format



Parameter List Input Fields

- IPFLAGS1 -- contains options for the ACCEPT function.

  IPPRTY (X'20') -- indicates that you want to send priority messages on this path.

  IPQUSCE (X'40') -- indicates that you do not want to receive messages on this path until an IUCV RESUME is issued.

  IPRMDATA (X'80') -- indicates that you are prepared to handle message data in the parameter list.

- IPMSGLIM -- contains the limit of outstanding messages that IUCV is to allow on the path.

- IPPATHID -- contains the path identification number of the path you are completing.

- IPUSER -- contains the user data that IUCV reflects to the target virtual machine.

Output from ACCEPT

- IPMSGLIM -- contains the message limit for this path.
- IPFLAGS1 -- contains specific information about this connection.

  IPPRTY (X'20') -- indicates that you may send priority messages.

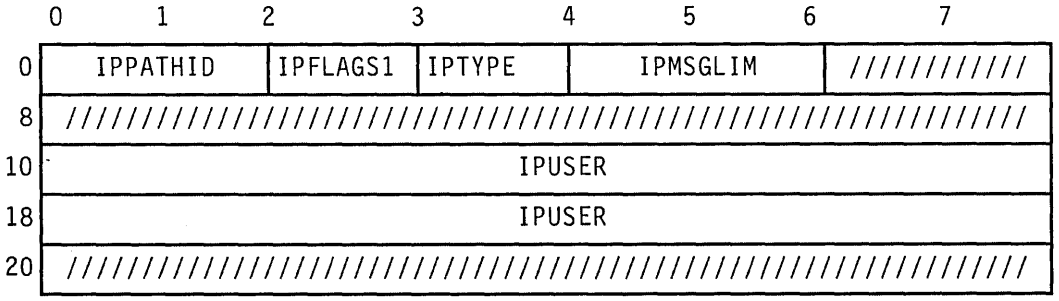
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Connection is not pending on this path 18 - Value in IPMSGLIM exceeds 255 20 - Originator has severed this path 30 - IPAPPC flag in IPFLAGS1 not zero

Connection Complete External Interrupt

To notify the source virtual machine that you have accepted the connection and completed a new IUCV path, IUCV reflects an IUCV Connection Complete external interrupt to the source virtual machine.

The source virtual machine receives this external interrupt if it is enabled for IUCV interrupts in Control Register 0 and the PSW. The functions of SET MASK and SET CONTROL MASK also control the presentation of this type of interrupt.



- IPFLAGS1 -- contains options for this path.

  IPCNTRL (X'04') -- indicates that this is a control path.

  IPPRTY (X'20') -- indicates that the virtual machine may receive priority messages on this path.

IPQUSCE (X'40') -- indicates that IUCV will not allow messages to be sent on this path until an IUCV RESUME is issued by the connecting virtual machine.

IPRMDATA (X'80') -- indicates that the connecting virtual machine can handle message data in the parameter list.

- IPMSGLIM -- contains the maximum number of messages that IUCV allows the connecting virtual machine to send on this path.
- IPPATHID -- contains the path id of the path which has now been established.
- IPTYPE -- indicates a Connection Complete external interrupt with a value of X'02'.
- IPUSER -- contains the user data specified by the target virtual machine when it ACCEPTed this connection.

SEND Function

The SEND function transmits data to another virtual machine. This data is called a "message" and may be specified in the parameter list or in a buffer. The message is sent over a path that has been established by the CONNECT function.

IUCV Macro Format

label	IUCV	SEND	,PRMLIST=	{address}
			,MF=	L
			,PATHID=	{address}
			,TRGCLS=	{address}
			,DATA=	{BUFFER|PRMMSG}
			,BUFLIST=	{YES|NO}
			,BUFFER=	{address|list}
			,BUFLEN=	{length}
			,PRMMSG=	{address}
			,TYPE=	{1WAY|2WAY}
			,ANSLIST=	{YES|NO}
			,ANSBUF=	{address|list}
			,ANSLEN=	{length}
			,MSGTAG=	{address}
			,PRTY=	{YES|NO}
			,SRCCLS=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the SEND parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

PATHID=
specifies the path over which you wish to send the message. The address of the PATHID is to a halfword value.

TRGCLS=
specifies the target message class associated with this message. This value, which is user defined, is considered part of the message identification (along with the PATHID and the MSGID field returned

by IUCV). It can be used by the target virtual machine to receive particular messages.

With the SEND function you have an option of sending the message in a buffer or in the parameter list. The size of a parameter list message is very limited (only 8 bytes), but the overhead of an IUCV RECEIVE is avoided which simplifies the communication process. The protocol that you chose on SEND does not affect the protocol chosen if the target must REPLY.

DATA=
specifies the location of your message data for this IUCV communication.

DATA=BUFFER indicates that your message is in a buffer. You can use the BUFFER, BUFLEN, or BUFLIST macro options to help you fill in the parameter list.

DATA=PRMMSG indicates that your message is in the parameter list. Use the PRMMSG option if you want the macro to fill in the parameter list.

BUFLIST=
specifies if the list format is being used.

BUFLIST=NO indicates that the list format is not being used. The BUFFER parameter is the address of the complete message.

BUFLIST=YES indicates that the address on the BUFFER parameter identifies a list of addresses and lengths of discontiguous buffers that hold the message text.

BUFFER=
specifies the address or the list of addresses from which IUCV moves the message. Any message buffers should not be reused until you receive a Message Complete external interrupt for this message.

BUFLEN=
specifies the total length of the message. If BUFFER specifies an address list (BUFLIST=YES), the value specified with BUFLEN is the total of the individual buffer lengths in the list.

PRMMSG=
specifies the eight bytes of message data that are moved into the parameter list.

With the SEND function you have an option of sending the message with and without a reply. If you depend on the target virtual machine processing the message (for example, updating a data base), you should use the 2-WAY protocol with a REPLY.

TYPE=
specifies whether a reply is expected to this message.

TYPE=1WAY indicates that this is a one-way message and that the receiver will not reply to the message.

TYPE=2WAY indicates that this is a two-way messages and that the receiver is expected to reply to the message. You can use the ANSBUF, ANSLEN, or ANSLIST macro options to help you fill in the parameter list.

ANSLIST=
specifies whether the list format is being used.

ANSLIST=NO indicates that the list format is not being used. The ANSBUF parameter is the address to contain the complete reply.

ANSLIST=YES indicates that the address on the ANSBUF parameter identifies a list of addresses and lengths of discontiguous buffers that contains the reply.

ANSBUF=
specifies the address or the list addresses into which IUCV moves the reply to this message. You do not know that the reply has been stored into this area until you receive a Message Complete external interrupt.

ANSLEN=
specifies the total length of the expected reply. If ANSBUF specifies an address list (ANSLIST=YES), the value specified with ANSLEN is the total of the individual buffer lengths in the list.

Additional options for SEND include tagging the messages, sending priority messages, and defining a source class.

MSGTAG=
specifies a tag to be associated with this message. This tag is returned to you on the IUCV Message Complete external interrupt. IUCV does not reference this tag so it may be used for any purpose that you desire. The tag information is not presented to the target user.

PRTY=
specifies whether this is a priority messages.

PRTY=YES indicates that this is a priority message.

PRTY=NO indicates that this is not a priority messages.

SRCCLS=
specifies the source message class associated with this message.

The tag information is not presented to the target user. The source

class is returned on the Message Complete external interrupt. You can use this field to identify different types of messages.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored in IPRCODE

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list or buffer address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

SEND Parameter List Format

	0	1	2	3	4	5	6	7
0	IPPATHID		IPFLAGS1	IPRCODE	IPMSGID			
8	IPTRGCLS				IPBFADR1 / IPRMMSG1			
10	IPBFLN1F / IPRMMSG2				IPSRCCLS			
18	IPMSGTAG				IPBFADR2			
20	IPBFLN2F				//////////////////////////////			

Parameter List Input Fields

- IPPATHID -- contains the path id on which to send the message.
- IPFLAGS1 -- contains options for the SEND function.

  IPAPPCSN (X'02') -- indicates the protocol to be used on this path. This bit must be set to zero.

IPANSLST (X'08') -- indicates that you are using an address list for the reply data.

IPNORPY (X'10') -- indicates that this is a one-way message. No reply expected.

IPPRTY (X'20') -- indicates that you are sending a priority message.

IPBUFLST (X'40') -- indicates that you are using an address list for the message data.

IPRMDATA (X'80') -- indicates that the message is in the parameter list.

- IPTRGCLS -- contains the target message class.
- IPBFADR1 -- contains the address of the message.
- IPBFLN1F -- contains the length of the message buffer. Use this label with a fullword value. Use IPBFLN1 with a halfword value.
- IPRMMSG1/IPRMMSG2 -- contains the message when it is stored in the parameter list rather than a buffer. The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.
- IPSRCCLS -- contains the source message class.
- IPMSGTAG -- contains the tag data of the message.
- IPBFADR2 -- contains the address to hold the reply.
- IPBFLN2F -- contains the length of the reply area. Use this label with a fullword value. Use IPBFLN2 with a halfword value.

Output from SEND

- IPMSGID -- contains a message id that IUCV assigns the message.
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

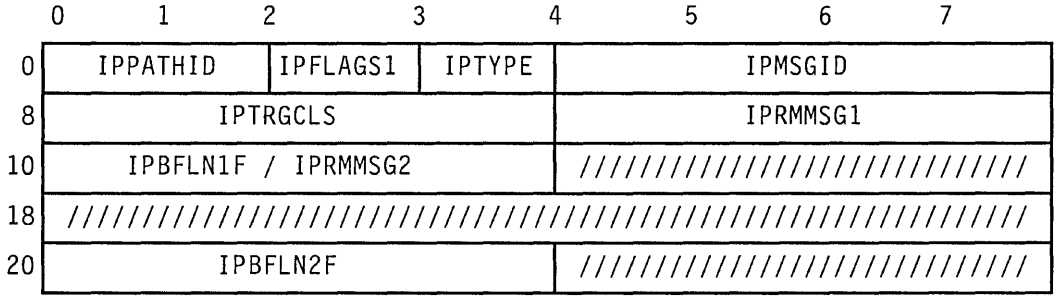
RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id 2 - Path quiesced - SENDs not allowed 3 - Message limit exceeded 4 - Priority messages not allowed on this path 10 - Message length is negative 21 - Message in parameter list not allowed on this path 25 - PRMMSG invalid with BUFLIST option 26 - Buffer list not on a doubleword boundary 27 - Answer list not on a doubleword boundary 30 - IPAPPCSN flag in IPFLAGS1 not zero

Message Pending External Interrupt

To notify the target virtual machine that you have sent a message, IUCV reflects an IUCV Message Pending external interrupt to the target virtual machine.

The target virtual machine receives this external interrupt if it is enabled for IUCV interrupts in Control Register 0 and the PSW. The SET MASK function also controls the presentation of this type of interrupt.

The external interrupt contains the information that the target virtual machine needs to continue processing the message. If the message is being sent in a buffer, the target continues processing by issuing a RECEIVE or a REJECT. If the message is contained in the interrupt information (IPRMDATA in IPFLAGS1 is on), a RECEIVE is not needed. If the interrupt indicates a one-way message with the data in the parameter list, no further IUCV processing is necessary.



- IPPATHID -- contains the path on which the message was sent.
- IPFLAGS1 -- contains options for this message.

  IPFGMCL (X'01') -- is always set to one indicating that the target message class has been stored at IPTRGCLS.

  IPFGPID (X'02') -- is always set to one indicating that the path id has been stored at IPPATHID.

IPFGMID (X'04') -- is always set to one indicating that the message id has been stored at IPMSGID.

IPNORPY (X'10') -- indicates that this is a one-way message and no REPLY is expected.

IPPRTY (X'20') -- indicates that this is a priority message.

IPRMDATA (X'80') -- indicates that the 8-byte message is in the interrupt information.

- IPTYPE -- indicates an Incoming Message external interrupt. If this is an incoming priority message, the interrupt type is X'08'. If this is an incoming nonpriority message, the interrupt type is X'09'.
- IPMSGID -- contains the message id.
- IPTRGCLS -- contains the target message class.
- IPBFLN1F -- contains the length of the message.
- IPRMMSG1/IPRMMSG2 -- contains the message when it is stored with the interrupt information (indicated by IPRMDATA in IPFLAGS1). The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.
- IPBFLN2F -- contains the length of the maximum expected reply.

RECEIVE Function

The RECEIVE function receives messages that are being sent to you over established paths. The RECEIVE function moves the message data from the source virtual machine to your virtual machine.

When receiving a message, you can completely identify the message by specifying the message id, path id, and target class. You can also identify the message by either the path id or the target class, or both. If you do not specify any identifiers when invoking the RECEIVE function, you receive the first message that has not been partially received.

If your receive area cannot contain the complete message, you can issue another RECEIVE to obtain the remainder of the message. If you use the same parameter list on the subsequent RECEIVE, the message id, path id, and message class that are already stored in the parameter list completely identify the message. These fields are required on any subsequent RECEIVEs for the same message. You must initialize the receive area and length for the subsequent RECEIVEs.

The RECEIVE function completes a one-way communication when all the data has been received.

IUCV Macro Format

label	IUCV	RECEIVE	,PRMLIST=	{address}
			,MF=	L
			,MSGID=	{address}
			,PATHID=	{address}
			,TRGCLS=	{address}
			,BUFLIST=	{YES|NO}
			,BUFFER=	{address|list}
			,BUFLEN=	{length}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the RECEIVE parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MSGID =
specifies the message id of the message to be received. If the message id is used to locate the message, the path id and the target class must also be correctly specified in the parameter list.

PATHID =
specifies the path over which you wish to receive the message. The address of the PATHID is to a halfword value.

TRGCLS =
specifies the target message class associated with this message.

BUFLIST =
specifies that the list format is being used.

BUFLIST = NO indicates that the list format is not being used. The BUFFER parameter is the address of the complete message.

BUFLIST = YES indicates that the address on the BUFFER parameter identifies a list of addresses and lengths of discontiguous buffers that hold the message text.

BUFFER =
specifies the address or the list of addresses into which IUCV moves the message.

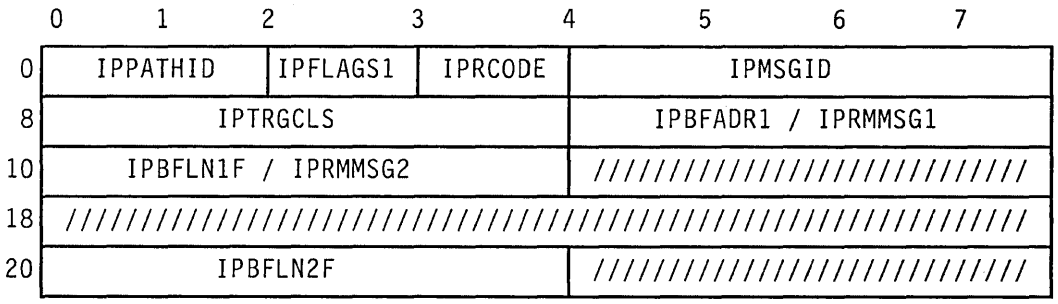
BUFLEN =
specifies the total length of the message to RECEIVE. If BUFFER specifies an address list (BUFLIST = YES), the value specified with BUFLEN is the total of the individual buffer lengths in the list.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE 2 - No message found

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary, or the message id was specified without the path id and the message class. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list, the buffer address, or the buffer list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address, buffer list address, or buffer address in the parameter list or the buffer list does not match the key of the user.

RECEIVE Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the path to receive the message.
- IPFLAGS1 -- contains options for the RECEIVE function.

  IPFGMCL (X'01') -- indicates that you have specified a target message class (IPTRGCLS) to identify the message you are trying to receive.

  IPFGPID (X'02') -- indicates that you have specified a path id (IPPATHID) for the message you are trying to receive.

  IPFGMID (X'04') -- indicates that you have specified a message id (IPMSGID) for the messages you are trying to receive.

IPAPPC (X'08') -- indicates the protocol to be used on this path. This bit must be set to zero.

IPBUFLST (X'40') -- indicates that you are using an address list for the message data.

- IPMSGID -- contains the message id of the message you are trying to receive.
- IPTRGCLS -- contains the target message class of the message you are trying to receive.
- IPBFADR1 -- contains the address of the receive buffer.
- IPBFLN1F -- contains the length of the receive buffer. Use this label with a fullword value. Use IPBFLN1 with a halfword value.

Output from RECEIVE

- IPPATHID -- contains the path id of the message you received.
- IPFLAGS1 -- contains specific information about the message received.

  IPFGMCL (X'01') -- is always set to one indicating that the target message class has been stored at IPTRGCLS.

  IPFGPID (X'02') -- is always set to one indicating that the path id has been stored at IPPATHID.

  IPFGMID (X'04') -- is always set to one indicating that the message id has been stored at IPMSGID.

  IPNORPY (X'10') -- indicates that this is a one-way message and no reply is expected.

  IPPRTY (X'20') -- indicates that this is a priority message.

  IPRMDATA (X'80') -- indicates that the 8-byte message is contained in the parameter list at IPRMMSG.

- IPMSGID -- contains the message id.
- IPTRGCLS -- contains the target message class.
- IPBFADR1 -- if BUFLIST = NO, contains the address of the buffer updated by the number of bytes you have received. If BUFLIST = YES, the address points to the current list entry IUCV is working on.
- IPRMMSG1/IPRMMSG2 -- contains the message when it is stored in the parameter list (indicated by IPRMDATA in IPFLAGS1). The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.

- IPBFLN1F -- contains one of the following values.

  If the receive buffer is the same length as the message, this field contains zero.

  If the receive buffer is longer than the message, this field contains the number of bytes remaining in the buffer.

  If the receive buffer shorter than the message, this field contains a residual count (that is, the number of bytes remaining in the message that does not fit into the buffer).

- IPBFLN2F -- contains the length of the expected reply. Use this label with a fullword value. Use IPBFLN2 with a halfword value.

- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return
1 - Invalid path id
5 - Receive buffer too short to contain message
6 - Fetch protection exception on send buffer
7 - Addressing exception on the send buffer
8 - Message id found but message class or path id invalid
9 - Message has been purged
10 - Message length is negative
22 - Send buffer list invalid
23 - Negative length in buffer list
24 - Incorrect total length of buffer list lengths
26 - Buffer list not on a doubleword boundary
30 - IPAPPC flag in IPFLAGS1 not zero

REPLY Function

The REPLY function responds to the two-way messages that you receive. The previous IUCV functions will have identified the the message to which you are replying by providing the message id, the path id, and the target class. You must identify completely the message to which you wish to reply.

The REPLY function moves the reply data from your virtual machine to the source virtual machine.

IUCV Macro Format

label	IUCV	REPLY	,PRMLIST=	{address}
			,MF=	L
			,MSGID=	{address}
			,PATHID=	{address}
			,TRGCLS=	{address}
			,DATA=	{BUFFER|PRMMSG}
			,ANSLIST=	{YES|NO}
			,ANSBUF=	{address|list}
			,ANSLEN=	{length}
			,PRMMSG=	{address}
			,PRTY=	{YES|NO}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the REPLY parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MSGID=
specifies the message id of the message to which you are replying.

PATHID=
specifies the path associated with the message.

TRGCLS=
specifies the target message class associated with the message.

With the reply function you have an option of replying to the message in a buffer or in the parameter list. The size of a parameter list message is very limited, but then the originator of the message (sender) does not have to maintain an answer buffer (ANSBUF parameter on SEND). The protocol you chose may be different from that used to send you the message.

DATA=
specifies the location of your message data for this IUCV communication.

DATA=BUFFER indicates that your reply data is in a buffer. You can use the ANSBUF, ANSLEN, or BUFLIST macro options to help you fill in the parameter list.

DATA=PRMMSG indicates that your reply data is in the parameter list. Use the PRMMSG parameter if you want the macro to fill in the parameter list.

ANSLIST=
specifies whether the list format is being used.

ANSLIST=NO indicates that the list format is not being used. The ANSBUF parameter is the address of the complete reply.

ANSLIST=YES indicates that the address on the ANSBUF parameter identifies a list of addresses and lengths of discontiguous buffers that contains the reply data.

ANSBUF=
specifies the address or a list of addresses from which IUCV moves the reply data.

Since the data is moved as part of the REPLY function, all buffer areas may be reused when the REPLY function completes.

ANSLEN=
specifies the total length of the reply data. If ANSBUF specifies an address list (ANSLIST=YES), the value specified with ANSLEN is the total of the individual buffer lengths in the list.

PRMMSG=
specifies the eight bytes of message data that are moved into the parameter list.

An additional option on REPLY lets you define priority messages.

PRTY=
specifies whether this is a priority message.

PRTY=YES indicates that this is a priority message.

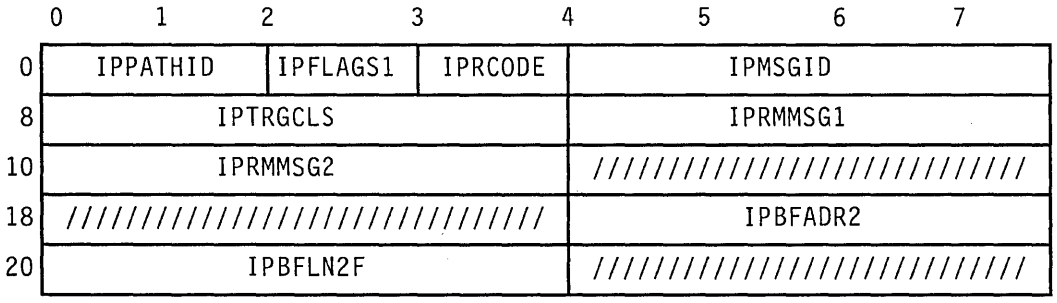
PRTY=NO indicates that this is not a priority message.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored in IPRCODE 2 - No message found

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address, the answer list address, the answer buffer address, or an answer buffer address in the answer list is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address, answer list address, answer buffer address, or an answer buffer address in the answer list does not match the key of the user.

REPLY Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the message to which you are replying.
- IPFLAGS1 -- contains the options for the REPLY function.

  IPANSLST (X'08') -- indicates that you are using an address list for the reply data.

  IPPRTY (X'20') -- indicates that this is a priority response.

  IPRMDATA (X'80') -- indicates that the reply is in the parameter list.
- IPMSGID -- contains the message id of the message to which you are replying.
- IPTRGCLS -- contains the message class of the message to which you are replying.
- IPRMMSG1/IPRMMSG2 -- contain the reply data when it is stored in the parameter list rather than a buffer. The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.
- IPBFADR2 -- contains the address of the reply data.
- IPBFLN2F -- contains the length of the reply data. Use this label with a fullword value. Use IPBFLN2 with a halfword value.

Output from REPLY

- IPBFADR2 -- contains the address of the reply data, when ANSLIST=NO, updated by the number of bytes of data that IUCV moved. If ANSLIST=YES is specified, the address points to the current list entry IUCV is working on.
- IPBFLN2F -- contains one of the following values:

  If the answer buffer is the same length as the reply, this field contains zero.

  If the answer buffer is longer than the reply, this field contains the number of bytes remaining in the buffer.

  If the answer buffer shorter than the reply, this field contains a residual count (that is, the number of bytes remaining in the reply that does not fit into the buffer).
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id 5 - Answer buffer too short to contain message 6 - Storage protection exception on answer buffer 7 - Addressing exception on answer buffer 8 - Message id found but message class or path id invalid 9 - Message has been purged 10 - Message length is negative 21 - Parameter list data not allowed on this path 22 - Send buffer list invalid 23 - Negative length in buffer list 24 - Incorrect total length of buffer list lengths 25 - PRMMSG option invalid with ANSLIST option 26 - Buffer list not on a doubleword boundary

Message Complete External Interrupt

To notify the originator of the message (the sender) that you have replied to the message, IUCV reflects an IUCV Message Complete external interrupt to the originator's virtual machine. If the sender had reserved buffers or address lists for use with this message, they can now be reused.

Note: Even though the user may not be dependent on the information contained in this interrupt, it should be processed as the message is not considered complete until the interrupt is reflected to the virtual machine.

	0	1	2	3	4	5	6	7
0	IPPATHID		IPFLAGS1	IPTYPE	IPMSGID			
8	IPAUDIT			///////	IPRMMSG1			
10	IPRMMSG2				IPSRCCLS			
18	IPMSGTAG				////////////////////////////			
20	IPBFLN2F				////////////////////////////			

- IPPATHID -- contains the path on which the message was sent.
- IPFLAGS1 -- contains specific information about the message.

  IPPRTY (X'20') -- indicates that this is a priority reply.

  IPRMDATA (X'80') -- indicates that the 8-byte reply is in the interrupt information.
- IPTYPE -- indicates a Message Complete external interrupt. If this is an incoming priority message completion, the interrupt type is X'06'. If

this is an incoming nonpriority message completion, the interrupt type is X'07'.

- IPMSGID -- contains the message id.

- IPAUDIT -- contains information about possible asynchronous error conditions which may have affected the normal completion of this message. If this field is zero, the message has completed successfully.

The meanings of the bits in the audit trail are:

IPADRPLE	X'800000'	Reply too long for buffer
IPADSNPX	X'400000'	Protection exception on send buffer
IPADSNAX	X'200000'	Addressing exception on send buffer
IPADANPX	X'100000'	Protection exception on answer buffer
IPADANAX	X'080000'	Addressing exception on answer buffer
IPADRJCT	X'040000'	Message was rejected
IPADPRMD	X'020000'	Reply specified DATA=PRMMSG, but this path cannot handle data in the parameter list.
	X'010000'	Reserved
IPADRCPX	X'008000'	Protection exception on receive buffer
IPADRCAX	X'004000'	Addressing exception on receive buffer
IPADRPPX	X'002000'	Protection exception on reply buffer
IPADRPAX	X'001000'	Addressing exception on reply buffer
IPADSVRD	X'000800'	Path was severed
IPADRLST	X'000400'	Invalid RECEIVE or REPLY address list
	X'000200'	Reserved
	X'000100'	Reserved
IPADBLEN	X'000080'	Bad length in SEND buffer list
IPADALEN	X'000040'	Bad length in SEND answer list
IPADBTOT	X'000020'	Invalid total SEND buffer length
IPADATOT	X'000010'	Invalid total SEND answer length
	X'000008'	Reserved
	X'000004'	Reserved
	X'000002'	Reserved
	X'000001'	Reserved

- IPRMMSG1/IPRMMSG2 -- contains the message when it is stored with the interrupt information (indicated by IPRMDATA in IPFLAGS1). The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.

- IPSRCCLS -- contains the source message class.

- IPMSGTAG -- contains the tag data of the message.

- IPBFLN2F -- contains one of the following values:

  If the answer buffer is the same length as the reply, this field contains zero.

  If the answer buffer is longer than the reply, this field contains the number of bytes remaining in the buffer.

If the answer buffer shorter than the reply, this field contains a residual count (that is, the number of bytes remaining in the reply that does not fit into the buffer). The IPADRPLE bit is set in the audit trail on this condition.

REJECT Function

The REJECT function refuses a specified message. Between the time that you are notified of a message and the time that you complete the message, the message may be rejected.

When a message is rejected, an IUCV Message Complete external interrupt is reflected to the source virtual machine with an indication in the audit trail (IPAUDIT) that the messages was rejected. Depending on when the message was rejected and the type of message, message data may or may not have been moved. When a message is rejected, the sender has no way to determine if any data has been transmitted.

When rejecting a message, you can completely identify the message by specifying the message id, path id, and target message class. You can also identify the message by either the path id or the target message class, or both.

IUCV Macro Format

label	IUCV	REJECT	,PRMLIST=	{address}
			,MF=	L
			,MSGID=	{address}
			,PATHID=	{address}
			,TRGCLS=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the REJECT parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MSGID=
specifies the message id of the message to be rejected. If the messages id is used to locate the message, the path id and the target class must also be correctly specified in the parameter list.

PATHID=
specifies the path id of the message to be rejected.

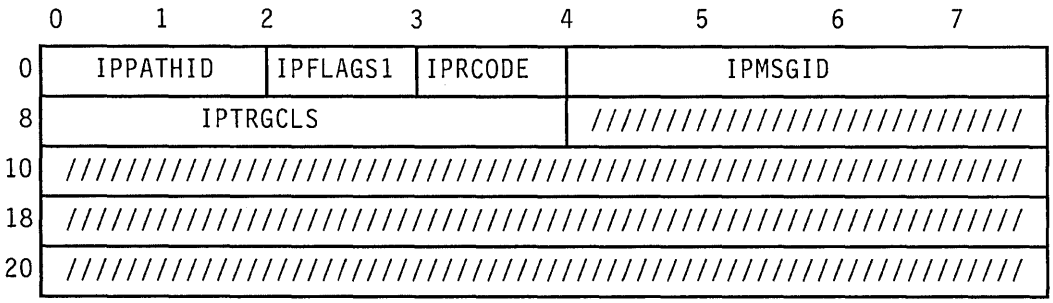
TRGCLS=
specifies the target message class of the message to be rejected.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored in IPRCODE 2 - No message found

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary, or the message id was specified with the path id and the message class. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

REJECT Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the message you are rejecting.
- IPFLAGS1 -- contains options for the REJECT function.

  IPFGMCL (X'01') -- indicates that you have specified a target message class (IPTRGCLS) for the message you are rejecting.

  IPFGPID (X'02') -- indicates, that you have specified a path id (IPPATHID) for the message you are rejecting.

  IPFGMID (X'04') -- indicates that you have specified a message id (IPMSGID) for the message you are rejecting.
- IPMSGID -- contains the message id of the message you are rejecting.
- IPTRGCLS -- contains the target message class of the message you are rejecting.

Output of REJECT

- IPPATHID -- contains the path id of the message you rejected.
- IPMSGID -- contains the message id of the message you rejected.
- IPTRGCLS -- contains the target message class of the message you rejected.
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id 8 - Message id found but message class or path id invalid

PURGE Function

The PURGE function cancels a message that you have sent. When you purge a message, one of the following actions takes place:

- If you purge a message before the target virtual machine has received an IUCV Message Pending external interrupt for this message, the target virtual machine is never aware that you sent the message.
- If you purge a message after the target virtual machine has received the IUCV Message Pending external interrupt, but before the target has completed handling the message, the target receives a return code indicating that the message has been purged the next time it references the message (normally, on a RECEIVE or a REPLY). The target is given only one such indication about the purged message. Any future references to the purged message results in a "no message found" condition.
- If you purge a message on which the target virtual has already completed its processing, the IUCV Message Complete external interrupt is avoided, but the target virtual machine is never aware that the message was purged.

When purging a message, you can completely identify the message by specifying the message id, path id, and source message class. You can also identify the message by the path id with or without the message class.

IUCV Macro Format

label	IUCV	PURGE	,PRMLIST=	{address}
			,MF=	L
			,MSGID=	{address}
			,PATHID=	{address}
			,SRCCLS=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the PURGE parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MSGID=
specifies the message id of the message to be purged. If the message id is used to identify the message, the pathid and the source class must also be correctly specified in the parameter list.

PATHID=
specifies the path id on which the message was sent. The address of the PATHID is a halfword value.

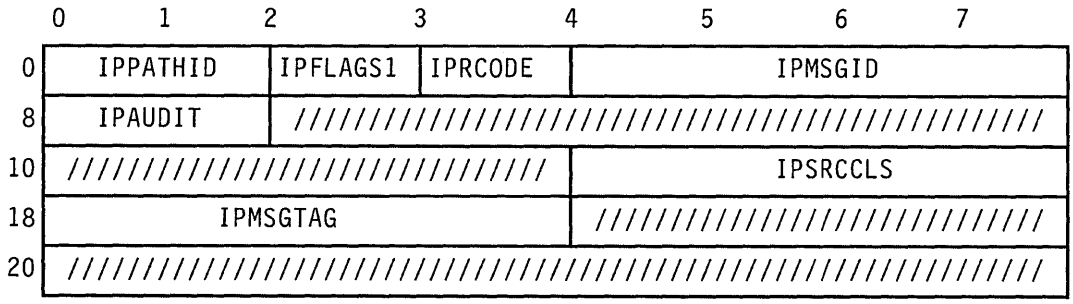
SRCCLS=
specifies the source message class associated with a message.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE 2 - No message found

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary, or the message id was specified without the path id or the message class. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

PURGE Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the message you are purging.
- IPFLAGS1 -- specifies options for the PURGE function.

  IPFGMCL (X'01') -- indicates that you have specified a source message class (IPSRCCLS) to identify the message you are trying to purge.

  IPFGPID (X'02') -- indicates that you have specified a path id (IPPATHID) for the message you are purging.

  IPFGMID (X'04') -- indicates that you have specified a message id (IPMSGID) for the message you are purging.
- IPMSGID -- contains the message id of the message you are purging.
- IPSRCCLS -- contains the source message class of the message you are purging.

Output for PURGE

- IPPATHID -- contains the path id of the message you purged.
- IPFLAGS1 -- contains specific information about the message purged.

  IPNORPY (X'10') -- indicates that the message purged is a one-way message.

  IPPRTY (X'20') -- indicates that the message purged is a priority message.
- IPMSGID -- contains the message id of the message you purged.

The meanings of the bits in the audit trail are:

IPADRPLE	X'800000'	Reply too long for buffer
IPADSNPX	X'400000'	Protection exception on send buffer
IPADSNAX	X'200000'	Addressing exception on send buffer
IPADANPX	X'100000'	Protection exception on answer buffer
IPADANAX	X'080000'	Addressing exception on answer buffer
IPADRJCT	X'040000'	Message was rejected
IPADPRMD	X'020000'	Reply specified DATA=PRMMSG, but this path cannot handle data in the parameter list.
	X'010000'	Reserved
IPADRCPX	X'008000'	Protection exception on receive buffer
IPADRCAX	X'004000'	Addressing exception on receive buffer
IPADRPPX	X'002000'	Protection exception on reply buffer
IPADRPAX	X'001000'	Addressing exception on reply buffer

IPADSVRD	X'000800'	Path was severed
IPADRLST	X'000400'	Invalid RECEIVE or REPLY address list
	X'000200'	Reserved
	X'000100'	Reserved
IPADBLEN	X'000080'	Bad length in SEND buffer list
IPADALEN	X'000040'	Bad length in SEND answer list
IPADBTOT	X'000020'	Invalid total SEND buffer length
IPADATOT	X'000010'	Invalid total SEND answer length
	X'000008'	Reserved
	X'000004'	Reserved
	X'000002'	Reserved
	X'000001'	Reserved

- IPSRCCLS -- contains the message class of the message you purged.
- IPMSGTAG -- contains the message tag of the message you purged.
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id 8 - Message found but message class invalid

SEVER Function

The SEVER function terminates an IUCV path to another virtual machine. You can terminate an established path or a pending connection. When you SEVER an established path, all of your outstanding messages on that path are purged, and any incoming messages on that path are rejected. When you have severed a path, communications on that path are no longer allowed and no communication, in either direction, can occur. The path id may be reused by IUCV when new paths are established.

When you receive an IUCV Connection Severed external interrupt, you can no longer send on that path, but you can process outstanding messages. You should always issue a SEVER in response to a Connection Severed interrupt to terminate your use of the path.

IUCV Macro Format

label	IUCV	SEVER	,PRMLIST=	{address}
			,MF=	L
			,ALL=	{YES|NO}
			,PATHID=	{address}
			,USERDTA=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the SEVER parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

ALL=
specifies whether all paths for this virtual machine are to be severed.

ALL=YES indicates that all of your paths are to be severed.

ALL=NO indicates that you do not want all of your paths severed, only the one specified by PATHID.

PATHID=
specifies the path id to be severed.

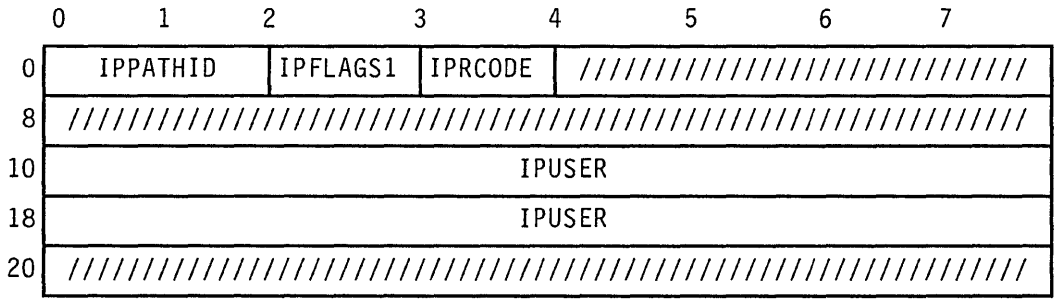
USERDTA=
specifies the data area containing the 16 bytes of user data that IUCV is to reflect across the path. The user data is reflected as part of the IUCV Connection Severed external interrupt.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

SEVER Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the path you want to sever.
- IPFLAGS1 -- contains options for the SEVER function.

  IPAPPC (X'08') -- indicates the protocol to be used on this path. This bit must be set to zero.

  IPALL (X'80') -- indicates that you want to sever all paths for this virtual machine.
- IPUSER -- contains the user data that IUCV reflects across the path.

Output from SEVER

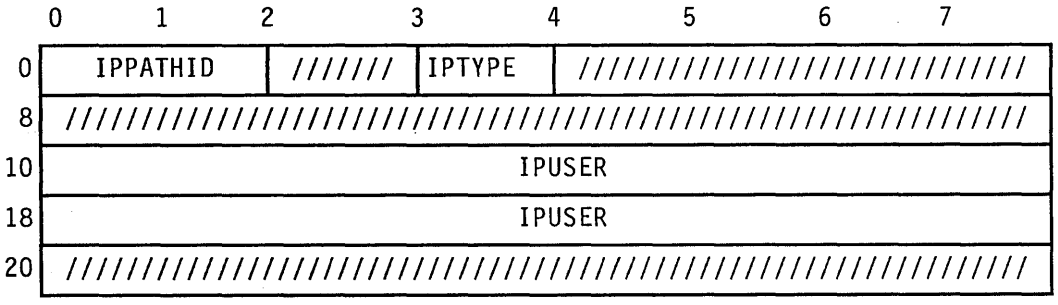
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id specified 30 - IPAPPC flag in IPFLAGS1 not zero

Connection Severed External Interrupt

To notify the other side of the path that you wish to terminate communication on a path, IUCV reflects an IUCV Connection Severed external interrupt.

The target virtual machine receives this external interrupt if it is enabled for IUCV interrupts in Control Register 0 and the PSW. The functions of SET MASK and SET CONTROL MASK also control the presentation of this type of interrupt.



- IPPATHID -- contains the path id of the path being severed.
- IPTYPE -- indicates a Connection Severed external interrupt with a value of X'03'.

- IPUSER -- contains the user data specified by the virtual machine that severed this path.

RETRIEVE BUFFER Function

The RETRIEVE BUFFER function terminates all use of IUCV. After the RETRIEVE BUFFER function completes, you may reuse the storage allocated for the IUCV external interrupt buffer since you will no longer receive IUCV external interrupts.

Since this function results in a SEVER of all IUCV paths, all outstanding IUCV communications are terminated as if each path has been individually SEVERed.

IUCV Macro Format

label	IUCV	RTRVBFR

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion

PROGRAM INTERRUPTIONS
Operation Exception The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state.

QUIESCE Function

The QUIESCE function temporarily suspends incoming messages on an IUCV path. You can later reactivate the path by invoking the RESUME function or you may leave the path quiesced, making it a one-way path. You are still allowed to send messages as the path is only quiesced for incoming messages.

If a message is sent to you on a quiesced path, a return code is returned to the sender, and the message is not sent. Each end of a path can be quiesced independently.

IUCV Macro Format

label	IUCV	QUIESCE	,PRMLIST=	{address}
			,MF=	L
			,PATHID=	{address}
			,ALL=	{YES|NO}
			,USERDTA=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the QUIESCE parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

PATHID=
specifies the path id of the path you want to quiesce.

ALL=
specifies whether all paths for this virtual machine are to be quiesced.

ALL=YES indicates that all of your paths are to be quiesced.

ALL=NO indicates that you do not want all of your paths quiesced, only the one specified by PATHID.

USERDTA=
specifies the data area containing the 16 bytes of user data to be reflected across the path. The user data is reflected as part of the IUCV Connection Quiesced external interrupt.

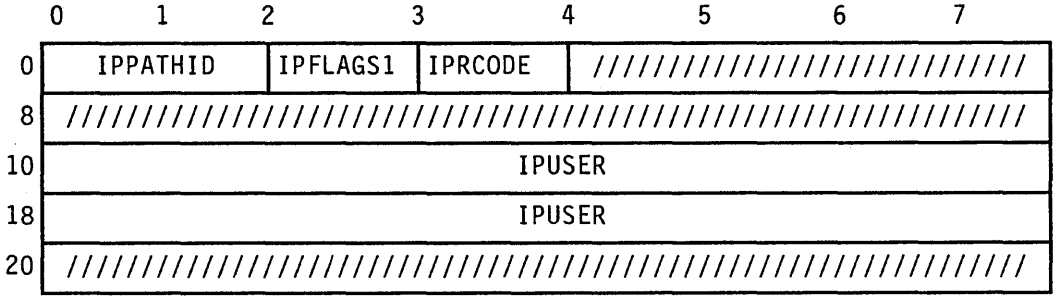
IUCV QUIESCE

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

QUIESCE Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the path you are quiescing.
- IPFLAGS1 -- contains options for the QUIESCE function.

  IPALL (X'80') -- indicates that you want to quiesce all paths for this virtual machine.
- IPUSER -- contains the user data that is reflected across the path.

Output of QUIESCE

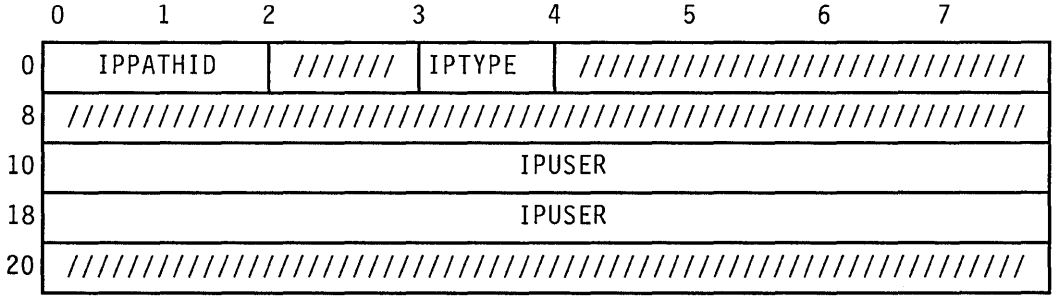
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id specified

Connection Quiesced External Interrupt

To notify the other side of the path that the path has been quiesced, IUCV reflects an IUCV Connection Quiesced external interrupt.

The target virtual machine receives this external interrupt if it is enabled for IUCV interrupts in Control Register 0 and the PSW. The functions of SET MASK and SET CONTROL MASK also control the presentation of this type of interrupt.



- IPPATHID -- contains the path id of the path quiesced.
- IPTYPE -- indicates a Connection Quiesced external interrupt with a value of X'04'.
- IPUSER -- contains the user data specified by the virtual machine that quiesced the path.

RESUME Function

The RESUME function restores communications over a quiesced path.

IUCV Macro Format

label	IUCV	RESUME	,PRMLIST=	{address}
			,MF=	L
			,PATHID=	{address}
			,ALL=	{YES|NO}
			,USERDTA=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the RESUME parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

PATHID=
specifies the path id of the path on which you want to resume getting messages.

ALL=
specifies whether communications should be restored for all paths for this virtual machine.

ALL=YES indicates that communications should be restored on all paths.

ALL=NO indicates that communications should be restored only on the path specified by PATHID.

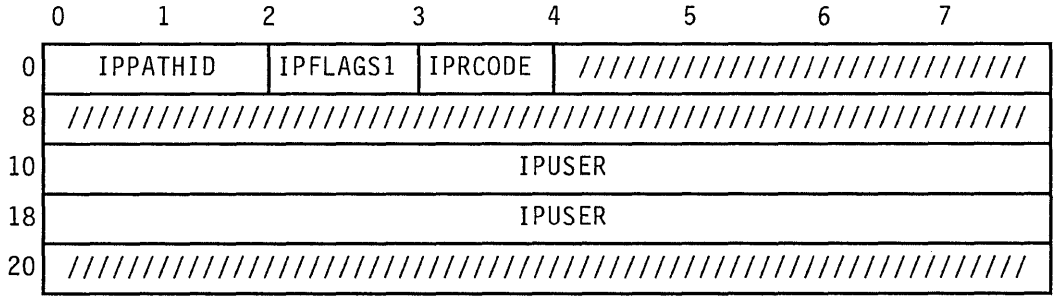
USERDTA=
specifies the data area containing the 16 bytes of user data that is to be reflected across the path. The user data is reflected as part of the IUCV Connection Resumed external interrupt.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

RESUME Parameter List Format



Parameter List Input Fields

- IPPATHID -- contains the path id of the path you are resuming.
- IPFLAGS1 -- contains options for the RESUME function.

  IPALL (X'80') -- indicates that you want to resume all paths for this virtual machine.
- IPUSER -- contains the user data that is reflected across the path.

Output of RESUME

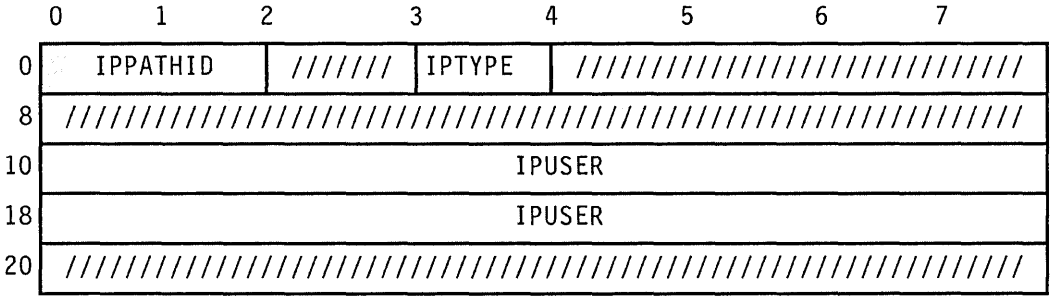
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id specified

Connection Resumed External Interrupt

To notify the other side of the path that the path has been resumed, IUCV reflects an IUCV Connection Resumed external interrupt.

The target virtual machine receives this external interrupt if it is enabled for IUCV interrupts in Control Register 0 and the PSW. The functions of SET MASK and SET CONTROL MASK also control the presentation of this type of interrupt.



- IPPATHID -- contains the path id of the path quiesced.
- IPTYPE -- indicates a Connection Quiesced external interrupt with a value of X'05'.
- IPUSER -- contains the user data specified by the virtual machine that quiesced the path.

TEST MESSAGE Function

The TEST MESSAGE function determines whether any IUCV Message Pending or IUCV Message Complete external interrupts are queued for your virtual machine.

This function, when used with the DESCRIBE and TEST COMPLETION functions, lets the virtual machine avoid the external interrupt handling associated with messages. In some applications, as when only one type of message is ever handled, the DESCRIBE may also be avoided and a RECEIVE issued directly.

When you receive message or message completions, IUCV informs you by reflecting a Message Pending or Message Complete external interrupt to your virtual machine. Therefore, unless you disable your virtual machine for Message Pending and Message Complete external interrupts, you should not use the TEST MESSAGE function.

If, when your virtual machine invokes the TEST MESSAGE function, it finds that there are no messages or message completions pending, your virtual machine enters a wait state. Your virtual machine remains enabled for all interrupts that were enabled when the TEST MESSAGE function was issued.

If, while your virtual machine is in a wait state, you receive an IUCV message or message completion, your virtual machine resumes execution by re-executing the TEST MESSAGE function (which returns a condition code).

IUCV Macro Format

label	IUCV	TESTMSG

IUCV Macro Completion Status

CONDITION CODES
1 - Message pending 2 - Message completion pending 3 - Both message and message completion pending

PROGRAM INTERRUPTIONS
Operation Exception The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state.

DESCRIBE Function

The DESCRIBE function determines whether you have a message pending for your virtual machine. If there is a message pending, information about the message is returned in the parameter list. Since you now have the message information, there is no need for IUCV to reflect an IUCV Message Pending external interrupt and you do not receive one for this message.

Since IUCV normally informs you of the message by reflecting a Message Pending external interrupt, you should not use the DESCRIBE function unless you have disabled for this type of interrupt. The IUCV SET MASK function can be used to disable your virtual machine for Message Pending external interrupts.

A message is described only once; either in the parameter list of a DESCRIBE or by a Message Pending external interrupt.

IUCV Macro Format

label	IUCV	DESCRIBE	,PRMLIST=	{address}
			,MF=	L

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the DESCRIBE parameter list. The IUCV instruction is generated to reference the address specified.

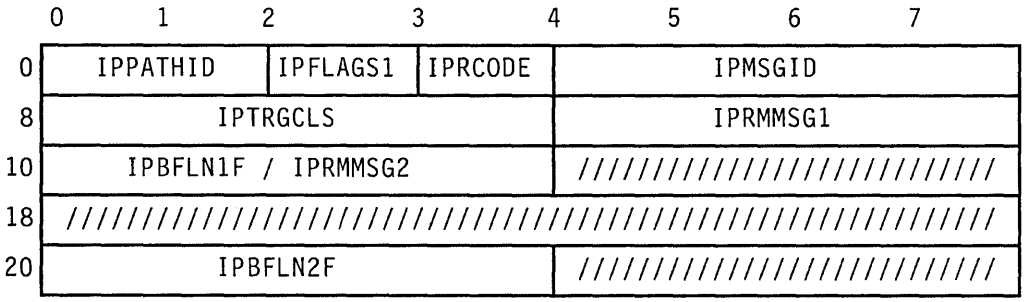
MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 2 - No message found

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

DESCRIBE Parameter List Format



Output of DESCRIBE

- IPPATHID -- contains the path on which the message was sent.
- IPFLAGS1 -- contains specific information about the message.

  IPFGMCL (X'01') -- is always set to one indicating that the target message class has been stored at IPTRGCLS.

  IPFGPID (X'02') -- is always set to one indicating that the path id has been stored at IPPATHID.

  IPFGMID (X'04') -- is always set to one indicating that the message id has been stored at IPMSGID.

  IPNORPY (X'10') -- indicates that this is a one-way message and no REPLY is expected.

  IPPRTY (X'20') -- indicates that this is a priority message.

IPRMDATA (X'80') -- indicates that the 8-byte message is in the parameter list at IPRMMSG.

- IPMSGID -- contains the message id.
- IPTRGCLS -- contains the target message class.
- IPRMMSG1/IPRMMSG2 -- contains the message when it is stored in the parameter list (indicated by IPRMDATA in IPFLAGS1). The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.
- IPBFLN1F -- contains the length of the message.
- IPBFLN2F -- contains the length of the maximum expected reply.
- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return

TEST COMPLETION Function

The TEST COMPLETION function determines whether you have a message completion pending for your virtual machine. If a message completion is pending, information about the message is returned in the parameter list. Since you now have the message completion information, there is no need for IUCV to reflect an IUCV Message Completion external interrupt and you will not receive one for this message.

Since IUCV normally informs you of a message completion by reflecting a Message Completion external interrupt, you should not use the TEST COMPLETION function unless you have disabled for this type of interrupt. The IUCV SET MASK function can be used to disable your virtual machine for Message Completion external interrupts.

When invoking the TEST COMPLETION function, you can completely identify the message by specifying the message id, path id, and source message class. You can also identify the message by either the path id or the source message class, or both. If you do not specify any identifiers when invoking the TEST COMPLETION function, any available message completion satisfies the function.

IUCV Macro Format

label	IUCV	TESTCMPL	,PRMLIST=	{address}
			,MF=	L
			,MSGID=	{address}
			,PATHID=	{address}
			,SRCCLS=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the TEST COMPLETION parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MSGID=
specifies the message id of the message. If the message id is used to locate the message, the path id and the source class must also be correctly specified in the parameter list.

PATHID =
specifies the unique path identification number associated with a message.

SRCCLS =
specifies the source message class associated with a message.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion 1 - Nonzero value stored at IPRCODE 2 - No message found 3 - Nonzero audit trail stored

PROGRAM INTERRUPTIONS
Specification Exception The parameter list is not on a doubleword boundary, or the message id was specified without the path id and the message class. **Operation Exception** The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Addressing Exception** The parameter list address that you specified is outside the virtual machine's storage. **Protection Exception** The storage key of the specified parameter list address does not match the key of the user.

TEST COMPLETION Parameter List Format

	0	1	2	3	4	5	6	7
0	IPPATHID		IPFLAGS1	IPRCODE	IPMSGID			
8	IPAUDIT		////////////////		IPRMMSG1			
10	IPRMMSG2				IPSRCCLS			
18	IPMSGTAG				////////////////////////////			
20	IPBFLN2F				////////////////////////////			

IUCV TEST COMPLETION

Parameter List Input Fields

- IPPATHID -- contains the path id for the message completion.
- IPFLAGS1 -- contains options for the TEST COMPLETION function.

  IPFGMCL (X'01') -- indicates that you have specified a source message class (IPSRCCL) to identify the message completion.

  IPFGPID (X'02') -- indicates that you have specified a path id (IPPATHID) to identify the message completion.

  IPFGMID (X'04') -- indicates that you have specified a message id (IPMSGID) to identify the message completion.
- IPMSGID -- contains the message id for the message completion.
- IPSRCCLS -- contains the source message class for the message completion.

Output of TEST COMPLETION

- IPPATHID -- contains the path id on which the message was sent.
- IPFLAGS1 -- contains specific information about the message.

  IPPRTY (X'20') -- indicates that this is a priority message.

  IPRMDATA (X'80') -- indicates that the 8-byte reply is in the parameter list.
- IPMSGID -- contains the message id.
- IPAUDIT -- contains information about possible asynchronous error conditions which may have affected the normal completion of this message. It this field is zero, the message has completed successfully.

The meanings of the bits in the audit trail are:

IPADRPLE	X'800000'	Reply too long for buffer
IPADSNPX	X'400000'	Protection exception on send buffer
IPADSNAX	X'200000'	Addressing exception on send buffer
IPADANPX	X'100000'	Protection exception on answer buffer
IPADANAX	X'080000'	Addressing exception on answer buffer
IPADRJCT	X'040000'	Message was rejected
IPADPRMD	X'020000'	Reply specified DATA=PRMMSG, but this path cannot handle data in the parameter list.
	X'010000'	Reserved
IPADRCPX	X'008000'	Protection exception on receive buffer
IPADRCAX	X'004000'	Addressing exception on receive buffer

IPADRPPX	X'002000'	Protection exception on reply buffer
IPADRPAX	X'001000'	Addressing exception on reply buffer
IPADSVRD	X'000800'	Path was severed
IPADRLST	X'000400'	Invalid RECEIVE or REPLY address list
	X'000200'	Reserved
	X'000100'	Reserved
IPADBLEN	X'000080'	Bad length in SEND buffer list
IPADALEN	X'000040'	Bad length in SEND answer list
IPADBTOT	X'000020'	Invalid total SEND buffer length
IPADATOT	X'000010'	Invalid total SEND answer length
	X'000008'	Reserved
	X'000004'	Reserved
	X'000002'	Reserved
	X'000001'	Reserved

- IPRMMSG1/IPRMMSG2 -- contains the message when it is stored in the parameter list (indicated by IPRMDATA in IPFLAGS1). The label IPRMMSG refers to the combined IPRMMSG1 and IPRMMSG2 fields.

- IPSRCCLS -- contains the source message class.

- IPMSGTAG -- contains the tag data of the message.

- IPBFLN2F -- contains one of the following values:

  If the buffer is exactly the correct length, this field contains zero.

  If the buffer is too long, this field contains the number of bytes unused in the buffer.

  If the buffer is too short, this field contains a residual count (that is, the number of bytes remaining of the reply that do not fit into the buffer). The IPADRPLE bit is set in the audit trail on this condition.

- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return 1 - Invalid path id 8 - Message id found but message class or path id invalid

SET MASK Function

The SET MASK function enables or disables the following IUCV external interruptions:

- Nonpriority message interrupts
- Priority message interrupts
- Priority reply interrupts
- Nonpriority reply interrupts
- IUCV control interrupts

Individual IUCV control interrupts can be controlled by using the SET CONTROL MASK function.

IUCV external interrupts are controlled by several masks in the following priority order:

1. Submask bit 30 of Control Register 0
2. Bit 7 of the virtual machine PSW
3. Bits defined by the SET MASK function
4. Bits defined by the SET CONTROL MASK function

IUCV Macro Format

label	IUCV	SETMASK	,PRMLIST=	{address}
			,MF=	L
			,MASK=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the SET parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

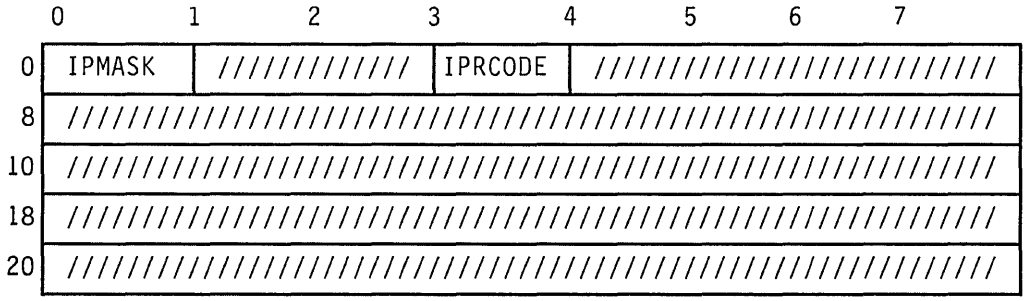
MASK=
specifies the mask byte to determine for which, if any, IUCV external interrupts a virtual machine is enabled.

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion

PROGRAM INTERRUPTIONS
Operation Exception The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Protection Exception** The storage key of the specified parameter list does not match the key of the user.

SET MASK Parameter List Format



Parameter List Input Fields

- IPMASK -- contains the mask that specifies for which, if any, IUCV interrupts your virtual machine is enabled. The meanings of the bits in the mask are:

IPSNDN	X'80'	Enable for nonpriority message interrupts
IPSNDP	X'40'	Enable for priority message interrupts
IPRPYN	X'20'	Enable for nonpriority message completion interrupts
IPRPYP	X'10'	Enable for priority message completion interrupts
IPCTRL	X'08'	Enable for IUCV control interrupts

Output of SET MASK

- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return

SET CONTROL MASK Function

The SET CONTROL MASK function enables or disables the following IUCV external interruptions:

- Pending Connection
- Connection Complete
- Connection Severed
- Connection Quiesced
- Connection Resumed

IUCV external interrupts are controlled by several masks in the following priority order:

1. Submask bit 30 of Control Register 0
2. Bit 7 of the virtual machine PSW
3. Bits defined by the SET MASK function
4. Bits defined by the SET CONTROL MASK function

IUCV Macro Format

label	IUCV	SETCMASK	,PRMLIST=	{address}
			,MF=	L
			,MASK=	{address}

Only the PRMLIST parameter is required. If you do not specify the other parameters, the macro assumes that you have stored the desired values into the parameter list before invoking the IUCV macro.

PRMLIST=
specifies the address of the SET parameter list. The IUCV instruction is generated to reference the address specified.

MF=L
lets you build an IUCV parameter list without initializing any registers or executing the IUCV instruction.

MASK=
specifies the mask byte to determine for which, if any, IUCV external interrupts a virtual machine is enabled.

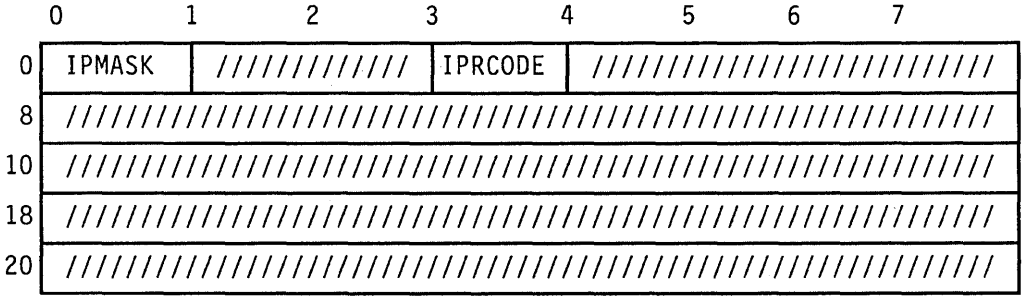
IUCV SET CONTROL MASK

IUCV Macro Completion Status

CONDITION CODES
0 - Normal completion

PROGRAM INTERRUPTIONS
Operation Exception The external interrupt buffer has not been declared using the DECLARE BUFFER function, or your virtual machine is not in supervisor state. **Protection Exception** The storage key of the specified parameter list does not match the key of the user.

SET CONTROL MASK Parameter List Format



Parameter List Input Fields

- IPMASK -- contains the mask that specifies for which, if any, IUCV interrupts your virtual machine is enabled. The meanings of the bits in the mask are:

IPCLPC	X'80'	Enable for pending connections interrupts
IPCLCC	X'40'	Enable for connection complete interrupts
IPCLPS	X'20'	Enable for connection severed interrupts
IPCLPQ	X'10'	Enable for connection quiesced interrupts
IPCLPR	X'08'	Enable for connection resumed interrupts

Output of SET CONTROL MASK

- IPRCODE -- contains the return code describing how this function completed. Possible values are listed below.

RETURN CODES in IPRCODE
0 - Normal return

Trace Table Entries

IUCV support generates a trace table entry for *each* IUCV function. There is one trace table entry type for IUCV entries (X'15'). Each entry contains a subtype field to indicate the exact IUCV function a communicator invoked.

Whether invoked from a virtual machine or from CP system code, *all* uses of IUCV are recorded in the CP trace table. The address portion of the old PSW is recorded as part of the entry. The X'80' bit in the RCODE byte indicates that this address is a real address (when invoked from CP) rather than a virtual address (when invoked from a virtual machine). For virtual machine addresses, the address of the associated VMBLOK can be obtained from preceding trace table entries.

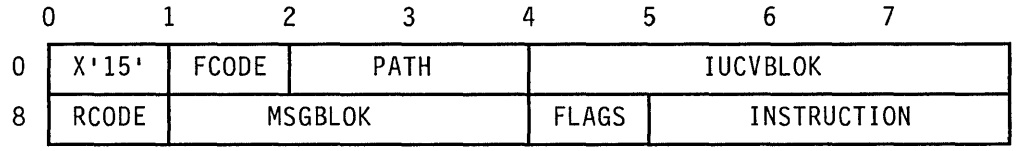
The IUCV trace facilities can be suppressed at assembly time by setting &TRACE(9) to 0 or at execution time by setting the X'80' bit to 0 in TRACFLG3 in PSA.

IUCV functions invoked by other functions are also recorded as if they had been invoked from CP. These secondary functions include:

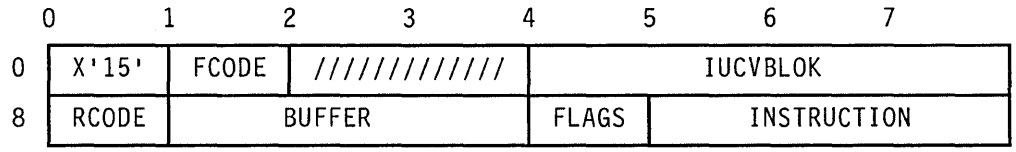
- The RETRIEVE BUFFER function generates a SEVER for all established paths.
- The SEVER function generates a REJECT for each incoming outstanding message and a PURGE for each outgoing outstanding message.
- A CONNECT issued to a CP system service passes control to that service. The selected CP system service usually invokes the ACCEPT function.
- The CP dispatcher invokes the DESCRIBE and TEST COMPLETION functions to dequeue messages intended for the CP system.

IUCV Trace Table Entry Formats

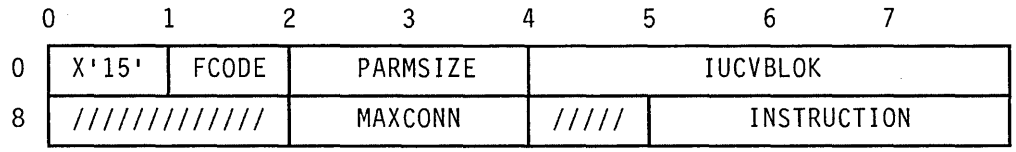
ACCEPT (X'0A'), CONNECT (X'0B'), DESCRIBE (X'03'), PURGE (X'09'), QUIESCE (X'0D'), RECEIVE (X'05'), REJECT (X'08'), REPLY (X'06'), RESUME (X'0E'), SEND (X'04'), SEVER (X'0F'), TEST COMPLETION (X'07')



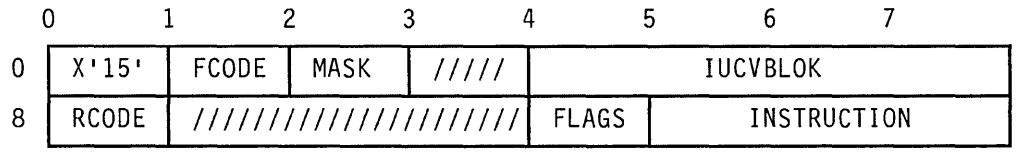
DECLARE BUFFER (X'0C'), RETRIEVE BUFFER (X'02')



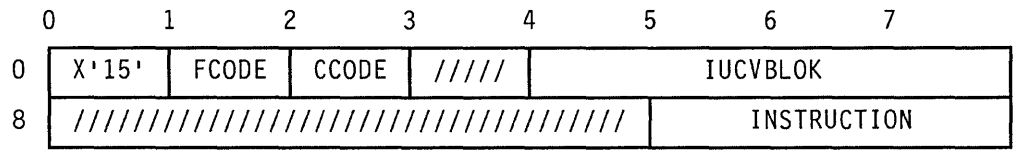
QUERY (X'00')



SET MASK (X'10'), SET CONTROL MASK (X'11')



TEST MESSAGE (X'01')



Trace Table Entry Field Definitions

This section explains, for each IUCV trace table field, the functions for which this field is valid, and the meaning of the field.

BUFFER (Used on DECLARE BUFFER, RETRIEVE BUFFER)

This field contains the virtual buffer address specified by the user for IUCV external interrupt information.

CCODE (Used on TEST MESSAGE)

This field contains the condition code returned to the invoker of the TEST MESSAGE function if a message was pending at the time the TEST MESSAGE function was issued. If no message is pending when TEST MESSAGE is issued, this field contains zero. Bits 6 and 7 of this CCODE field are used for the condition code.

FCODE (Used on all entries)

This field indicates the exact function executed. One of the following function codes is found in this field.

```
X'00' - QUERY
X'01' - TEST MESSAGE
X'02' - RETRIEVE BUFFER
X'03' - DESCRIBE
X'04' - SEND
X'05' - RECEIVE
X'06' - REPLY
X'07' - TEST COMPLETION
X'08' - REJECT
X'09' - PURGE
X'0A' - ACCEPT
X'0B' - CONNECT
X'0C' - DECLARE BUFFER
X'0D' - QUIESCE
X'0E' - RESUME
X'0F' - SEVER
X'10' - SET MASK
X'11' - SET CONTROL MASK
```

FLAGS (Used on ACCEPT, CONNECT, DECLARE BUFFER, DESCRIBE, PURGE, QUIESCE, RECEIVE, REJECT, REPLY, RESUME, SEND, SET CONTROL MASK, SET MASK, SEVER, TEST COMPLETION)

This field is a copy of the input flags specified by the user in the field IPFLAGS1 of the parameter list. Note that the use of these flags varies by function and that the user may have set flags that are not used by the function.

INSTRUCTION (Used on all entries)

This field contains the address of the instruction following where the function was invoked. This address is a real address if the IPCPENTY (X'08') bit in the field FLAGS is set to one. Otherwise, the address is an address in a virtual machine.

IUCVBLOK (Used on all entries)

This field contains the address of the IUCVBLOK associated with the invoker. For the QUERY function, this field may be zero if no IUCVBLOK currently exists for the invoker. For the DECLARE BUFFER function, this field contains the address of the IUCVBLOK created by this function.

MASK (Used on SET MASK, SET CONTROL MASK)

This field contains a copy of the mask field that was specified by the virtual machine.

MAXCONN (Used on QUERY)

This field contains the maximum number of connections allowed by the virtual machine issuing this request.

MSGBLOK (Used on DESCRIBE, PURGE, RECEIVE, REJECT, REPLY, SEND, TEST COMPLETION)

This field contains the real address of the MSGBLOK processed by this request.

PARMSIZE (Used on QUERY)

This field contains the size of IUCV parameter list returned to the invoker of the QUERY function.

PATH (Used on ACCEPT, CONNECT, DESCRIBE, PURGE, QUIESCE, RECEIVE, REJECT, REPLY, RESUME, SEND, SEVER, TEST COMPLETION)

This field contains the path id of the path associated with this request. For the CONNECT function, this is the path id associated with the path being created. For the other functions, this is the path id used to process the request.

RCODE (Used on ACCEPT, CONNECT, DECLARE BUFFER, DESCRIBE, PURGE, QUIESCE, RECEIVE, REJECT, REPLY, RESUME, SEND, SET CONTROL MASK, SET MASK, SEVER, TEST COMPLETION)

This field contains the code returned in the field IPRCODE of the parameter list. If this return code field is non-zero, only the TYPE, FCODE, INSTRUCTION, FLAGS, and IUCVBLOK fields are valid. The other fields may be invalid because of the return code. Invalid fields always contain zeroes.

IUCV System Services

IUCV treats communications with CP as if CP were a single virtual machine. IUCV gathers information about a message and routes it to the proper module in CP for processing.

IUCV provides:

- Routing of connections to IUCV system services
- Routing of messages to IUCV system services
- Routing of message completions to the IUCV system service that issued the SEND
- Severing of connections to IUCV system services.

Each IUCV system service that interfaces with virtual machines is uniquely defined to IUCV. The following table shows the corresponding userid for each of the IUCV system services. This userid must be specified on the USERID= parameter when invoking the IUCV CONNECT function.

System Service Userid	System Service
*BLOCKIO	DASD Block I/O System Service
*CCS	Console Communication Services
*IDENT	Identify System Service[11]
*LOGREC	Error Logging System Service
*MSG	Message System Service
*MSGALL	Message All System Service
*RPI	Access Verification System Service
*SIGNAL	Signal System Service
*SPL	SPOOL System Service
*CRM	Collection Resource Management System Service[11]

Figure 8. CP System Services and Their Userids

The IUCV System Services use the IUCV macro to invoke IUCV functions just as a virtual machine does. However, to generate the proper linkage, two parameters are provided on the macro for use by CP system code.

---

11 This system service is documented in the *VM/SP Transparent Services Access Facility Reference.*

These macro parameters are CP= and VMBLOK=. These parameters are used to indicate CP system code is invoking the macro and to indicate if the function is for a system service or for a virtual machine.

CP does not, and can not, use all of the functions provided by the IUCV macro. Only those currently used by CP are supported for use by CP.

Chapter 3. CMS IUCV

The Inter-User Communications Vehicle (IUCV) and the Advanced Program-to-Program Communication/VM (APPC/VM) are communication facilities. IUCV allows a program running in a virtual machine to communicate with other virtual machines, with a CP system service, and with itself. See "The Inter-User Communications Vehicle" for more detail.

APPC/VM enables a program running in a virtual machine to communicate with resources in virtual machines in the same system, or if TSAF is running, on different systems within a specified group, or collection. See the *VM/SP Transparent Services Access Facility Reference* for more detail. CMS support of IUCV and APPC/VM makes it easier for multiple programs, operating within one virtual machine, to use IUCV and APPC/VM functions.

You can invoke IUCV and APPC/VM functions through the CMS macros, HNDIUCV and CMSIUCV. These macros enable you to:

- Initialize and terminate a program's IUCV or APPC/VM environment
- Begin or terminate communications with another virtual machine (in the same system or different systems) or with a CP system service (IUCV only)
- Specify path-specific exits for IUCV and APPC/VM external interrupts.

HNDIUCV Macro

Use the HNDIUCV macro to identify an IUCV or APPC/VM program to CMS. HNDIUCV initializes or terminates the virtual machine's IUCV and APPC/VM environment. No CMS IUCV function is permitted by a particular program unless the program has first issued the HNDIUCV macro and identified itself to CMS.

The four formats of the HNDIUCV macro are:

- Standard
- List (MF=L)
- Complex List (MF=(L,addr[,label]))
- Execute (MF=(E,addr)).

Standard Format

The Standard format of the HNDIUCV macro is:

[*label*]	**HNDIUCV**	**SET,NAME=***addr*,**EXIT=***addr*[,**UWORD=***addr*][,**ERROR=***addr*] **REP,NAME=***addr*[,**ERROR=***addr*] {,**EXIT=***addr* / ,**UWORD=***addr* / ,**EXIT=***addr*,**UWORD=***addr*} **CLR,NAME=***addr*[,**ERROR=***addr*]

where:

addr
is an assembler program label or an address stored in a general purpose register. If a register is used, it must be enclosed in parentheses. Also, the register must contain a non-zero value. A zero value is treated as though the parameter was not specified, and any defaults are used. If the parameter is required by the macro function, a non-zero return code is generated.

label
is an assembler program label.

SET
identifies the program to CMS. It must be issued before invoking any CMS IUCV functions. Upon error free completion, register 0 contains the maximum number of possible connections for the virtual machine. If the user is connecting to the Identify System Service (*IDENT) for resource identification, the NAME field must be equal to the resource name that is being identified.

CLR
removes the program from the list of active CMS IUCV programs. This function should be issued when the program no longer wishes to do any more IUCV or APPC/VM communications. Any paths associated with this program are SEVERed when this function is requested (the IPUSER field of the IUCV SEVER parameter list is set to binary ones to indicate the SEVER was done by CMS).

REP
replaces the currently defined exit address and/or UWORD field for a specified program. Only the parameters specified are replaced.

NAME=

label is an assembler program label that is the address of an 8 character symbolic name.

(Rn) is a general purpose register. Its value is the address of an 8 character symbolic name.

This symbolic name is used as the CMS IUCV program's identity. When this program issues the CMSIUCV macro to perform an IUCV or APPC/VM function, the NAME parameter specified on the CMSIUCV macro must be the same as the one specified here. This parameter is required to execute the HNDIUCV function.

EXIT=

label is an assembler program label that is the address of the exit routine.

(Rn) is a purpose general register. Its value is the address of the exit routine.

The exit routine receives control whenever an IUCV or APPC/VM external interrupt of the type "PENDING CONNECT" occurs for this program. To activate this exit, the connecting virtual machine must specify the same symbolic name in the first 8 bytes of the IPUSER field of its CONNECT parameter list as the NAME parameter here. This exit address is the default address associated with any path owned by this program. If an IUCV or APPC/VM external interrupt occurs on a path where no specific exit has been established (a pending connect external interrupt has previously occurred on this path and no CMSIUCV ACCEPT has been issued yet, or the EXIT parameter was not specified on the CMSIUCV CONNECT or ACCEPT that established the path), this address receives control. This parameter must be specified on the SET function, but it is optional on the REP function.

UWORD=

label is an assembler program label that is the address stored as the UWORD.

(Rn) is a general purpose register. Its contents are stored as the UWORD.

UWORD is an optional fullword that can be specified by the invoker for any purpose desired. When the exit routine receives control, register 0 contains either an address if a label was used or the value of the register if a register was used. If this parameter is not specified, the UWORD is set to zero. (If the UWORD value is not specified when a CMSIUCV ACCEPT or CONNECT is issued, it defaults to the UWORD value specified on this HNDIUCV macro.)

ERROR=

label is an assembler program label that is the address of the error routine.

(Rn) is a general purpose register. Its value is the address of the error routine.

The error routine receives control if an error is found. If this parameter is not specified and an error occurs, control returns to the next sequential instruction in the calling program.

List Format

The List format (MF=L) of the HNDIUCV macro is:

[*label*]	**HNDIUCV MF=L**	[**,NAME=***label*][**,EXIT=***label*][**,UWORD=***label*] **,SET** [**,NAME=***label*][**,EXIT=***label*][**,UWORD=***label*] **,REP** [**,NAME=***label*][**,EXIT=***label*][**,UWORD=***label*] **,CLR** [**,NAME=***label*]

All parameters have the same meaning as the Standard format with the following difference:

MF=L
indicates that the parameter list is created in-line. No executable code is generated. Register notation cannot be used for macro parameter addresses.

Note: When using the MF= parameter, all other parameters are optional. When the function is executed, however, a valid combination of parameters must have been specified by the List and Execute formats of the macro.

Complex List Format

The Complex List format (MF=(L,addr[,label])) of the HNDIUCV macro is:

[*label*]	**HNDIUCV MF=(L,***addr*[,*label*]**)**	[**,NAME=***addr*][**,EXIT=***addr*][**,UWORD=***addr*] **,SET**[**,NAME=***addr*][**,EXIT=***addr*][**,UWORD=***addr*] **,REP**[**,NAME=***addr*][**,EXIT=***addr*][**,UWORD=***addr*] **,CLR**[**,NAME=***addr*]

All parameters have the same meaning as the Standard format with the following difference:

MF=(L,addr[,label])
indicates that the parameter list is created in the area specified by "addr". The address may represent an area within your program or an area of free storage obtained by a system service. You can determine

the size of the parameter list by coding the "label" operand. The macro expansion equates "label" to the size of the parameter list. This format of the macro produces executable code to move the data into the parameter list specified by "addr". However, it does not generate instructions to invoke the function. If this version of the List format is used, it must be executed before any related invocation of the Execute format.

Note: When using the MF= parameter, all other parameters are optional. When the function is executed, however, a valid combination of parameters must have been specified by the List and Execute formats of the macro.

Execute Format

The Execute format (MF=(E,addr)) of the HNDIUCV macro is:

[*label*]	**HNDIUCV MF=(E,** *addr* **)**	[**,NAME=***addr*][**,EXIT=***addr*][**,UWORD=***addr*] [**,ERROR=***addr*] **,SET**[**,NAME=***addr*][**,EXIT=***addr*][**,UWORD=***addr*] [**,ERROR=***addr*] **,REP**[**,NAME=***addr*][**,EXIT=***addr*][**,UWORD=***addr*] [**,ERROR=***addr*] **,CLR**[**,NAME=***addr*][**,ERROR=***addr*]

All parameters have the same meaning as the Standard format with the following difference:

MF=(E,addr)
indicates that instructions are generated to execute the HNDIUCV function. "addr" represents the location of the parameter list. Information in the parameter list may be changed by specifying the appropriate operands on the macro.

Note: When using the MF= parameter, all other parameters are optional. When the function is executed, however, a valid combination of parameters must have been specified by the List and Execute formats of the macro.

Error Conditions

If an error occurs, register 15 contains one of the following return codes:

Code	Meaning
4	A program with this name has previously issued a HNDIUCV SET (SET)
8	No HNDIUCV SET has been issued for this program (REP,CLR)

16 The NAME parameter was not specified or its address is equal to zero (SET,REP,CLR)

20 The EXIT parameter was not specified or its address is equal to zero (SET)

32 An IUCV DECLARE BUFFER has already been issued by a non-CMS IUCV program. CMS IUCV cannot be initialized (SET)

36 Errors were encountered reading the directory for the virtual machine during CMS IUCV initialization (SET)

40 Unrecognized function

48 The IUCV DCLBFR CONTROL=YES failed, as indicated by CP.

60 HNDIUCV CLR or HNDIUCV REP cannot be issued by the CMS supervisor.

72 The HNDIUCV SET cannot be performed because the SET for the CMS supervisor failed during IPL.

2xx An error was encountered in getting CMS free storage. "xx" = the return code from DMSFREE. (SET)

1xxx While trying to SEVER all of the program's paths, an IUCV SEVER error occurred. "xxx" is the IPRCODE field that was returned by IUCV to aid in diagnosing the error. (CLR)

CMSIUCV Macro

Use the CMSIUCV macro to begin or terminate IUCV communications with another IUCV program or with CP. Also, use the CMSIUCV macro to begin or terminate APPC/VM communications with an APPC/VM resource manager.

The four formats of the CMSIUCV macro are:

- Standard
- List (MF=L)
- Complex List (MF=(L,addr[,label]))
- Execute (MF=(E,addr)).

Standard Format

The Standard format of the CMSIUCV macro is:

[*label*]	**CMSIUCV**	**CONNECT,NAME=***addr* **,PRMLIST=***addr* **[,EXIT=***addr***]** **[,UWORD=** *addr***][,ERROR=** *addr***]** **ACCEPT,NAME=***addr* **,PRMLIST=***addr* **[,EXIT=***addr***]** **[,UWORD=** *addr***][,ERROR=** *addr***]** **SEVER ,NAME=***addr* **,PRMLIST=***addr* **[,ERROR=** *addr***]** **[,CODE={ALL|ONE}]**

where:

addr
is an assembler program label or an address stored in a general purpose register. If a register is used, it must be enclosed in parentheses. Also, the register must contain a non-zero value. A zero value is treated as though the parameter was not specified, and defaults are used. If the parameter was required by the macro function, a non-zero return code is generated.

label
is an assembler program label.

CONNECT
requests CMS to perform an IUCV or APPC/VM CONNECT. A CONNECT parameter list must be set up by the program and passed to CMS.

ACCEPT
requests CMS to perform an IUCV or APPC/VM ACCEPT. An ACCEPT parameter list must be set up by the program and passed to CMS.

SEVER
requests CMS to perform an IUCV or APPC/VM SEVER. A SEVER parameter list must be set up by the program and passed to CMS. Any EXIT established for the path being SEVERed is terminated. A SEVER with the IPALL bit turned on, which would cause a SEVER of all paths for the virtual machine, is not permitted.

EXIT=

label — is an assembler program label that is the address of the exit routine.

(Rn) — is a general purpose register. Its value is the address of the exit routine.

The exit routine receives control whenever an IUCV or APPC/VM external interrupt occurs on this IUCV or APPC/VM path. If this parameter is not specified, the exit address defaults to the address specified in the HNDIUCV macro for this program. Any time an IUCV or APPC/VM external interrupt occurs for the specific IUCV or APPC/VM path, the address is given control.

UWORD=

label is an assembler program label that is the address that is stored as the UWORD.

(Rn) is a general purpose register. Its contents are stored as the UWORD.

UWORD is an optional fullword that can be specified by the invoker for any purpose desired. When the exit routine receives control, register 0 contains the value of the UWORD associated with the path on which the IUCV or APPC/VM external interrupt occurred. Register 0 contains the address if a label was used, or the value of the register if a register was used. If this parameter is not specified, the UWORD value defaults to the value specified on the HNDIUCV macro for this program.

PRMLIST=

label is an assembler program label. It is the address of the program's IUCV PRMLIST.

(Rn) is a general purpose register. Its value is the address of the program's IUCV PRMLIST.

This address points to the block of storage that contains the IUCV or APPC/VM parameter list for the function desired. This parameter list must be previously prepared by the program. It is suggested that the program use the List form of the IUCV or APPC/VM macro to prepare the parameter list. By using this form, the program may set up the IUCV or APPC/VM parameter list by using KEYWORD parameters on the IUCV or APPC/VM macro instead of storing information using the IPARML DSECT. This parameter is required.

CODE=

is only valid when the SEVER function is requested. If CODE=ALL, all paths owned by the program are SEVERed. If CODE=ONE, only the one path specified via the pathid is SEVERed. If this parameter is not specified, CODE=ONE is used as the default.

NAME=

label is an assembler program label. It is the address of an 8 character symbolic name.

(Rn) is a general purpose register. Its value is the address of an 8 character symbolic name.

This symbolic name identifies the program associated with this path. A program with this name must have previously issued an HNDIUCV macro to identify itself as a CMS IUCV program to CMS. This parameter must be specified.

ERROR=

label is an assembler program label that is the address of the error routine.

(Rn) is a general purpose register. Its value is the address of the error routine.

The error routine receives control if an error is found. If this parameter is not specified and an error occurs, control returns to the next sequential instruction in the calling program.

List Format

The List format (MF=L) of the CMSIUCV macro is:

[*label*]	**CMSIUCV MF=L**	[[**,NAME=***label*][**,PRMLIST=***label*][**,EXIT=***label*] [**,UWORD=***label*][**,CODE=**{**ALL** | **ONE**}] **,CONNECT** [**,NAME=***label*][**,PRMLIST=***label*][**,EXIT=***label*] [**,UWORD=***label*] **,ACCEPT** [**,NAME=***label*][**,PRMLIST=***label*][**,EXIT=***label*] [**,UWORD=***label*] **,SEVER** [**,NAME=***label*][**,PRMLIST=***label*] [**,CODE=**{**ALL** | **ONE**}]]

All parameters have the same meaning as the Standard format with the following difference:

MF=L
indicates that the parameter list is created in-line. No executable code is generated. Register notation cannot be used for macro parameter addresses.

Note: When using the MF= parameter, all other parameters are optional. When the function is executed, however, a valid combination of parameters must have been specified by the List and Execute formats of the macro.

Complex List Format

The Complex List format (MF=(L,addr[,label])) of the CMSIUCV macro is:

[*label*]	**CMSIUCV MF=(L,** *addr*[,*label*]**)**	[,**NAME=***addr*][,**PRMLIST=***addr*][,**EXIT=***addr*] [,**UWORD=***addr*][,**CODE=**{**ALL**|**ONE**}] ,**CONNECT** [,**NAME=***addr*][,**PRMLIST=***addr*] [,**EXIT=***addr*][,**UWORD=***addr*] ,**ACCEPT** [,**NAME=***addr*][,**PRMLIST=***addr*] [,**EXIT=***addr*][,**UWORD=***addr*] ,**SEVER** [,**NAME=***addr*][,**PRMLIST=***addr*] [,**CODE=**{**ALL**|**ONE**}]

All parameters have the same meaning as the Standard format with the following difference:

MF=(L,addr[,label])
indicates that the parameter list is created in the area specified by "addr". The address may represent an area within your program or an area of free storage obtained by a system service. You can determine the size of the parameter list coding the "label" operand. The macro expansion equates "label" to the size of the parameter list. This format of the macro produces executable code to move the data in the parameter list specified by "addr". However, it does not generate instructions to invoke the function. If this version of the List format is used, it must be executed before any related invocation of the Execute format.

Note: When using the MF= parameter, all other parameters are optional. When the function is executed, however, a valid combination of parameters must have been specified by the List and Execute formats of the macro.

Execute Format

The Execute format (MF=(E,addr)) of the CMSIUCV macro is:

[*label*]	**CMSIUCV MF=(E,***addr***)**	[,**NAME=***addr*][,**PRMLIST=***addr*][,**EXIT=***addr*] [,**UWORD=***addr*][,**ERROR=** *addr*] [,**CODE=**{**ALL**|**ONE**}] ,**CONNECT** [,**NAME=***addr*][,**PRMLIST=***addr*] [,**EXIT=***addr*][,**UWORD=***addr*][,**ERROR=** *addr*] ,**ACCEPT** [,**NAME=***addr*][,**PRMLIST=***addr*] [,**EXIT=***addr*][,**UWORD=***addr*][,**ERROR=** *addr*] ,**SEVER** [,**NAME=***addr*][,**PRMLIST=***addr*] [,**CODE=**{**ALL**|**ONE**}] [,**ERROR=** *addr*]

All parameters have the same meaning as the Standard format with the following difference:

MF=(E,addr)
indicates that instructions are generated to execute the CMSIUCV function. "addr" represents the location of the parameter list. Information in the parameter list may be changed by specifying the appropriate operands on the macro.

Note: When using the MF= parameter, all other parameters are optional. When the function is executed, however, a valid combination of parameters must have been specified by the List and Execute formats of the macro.

Usage Notes:

1. To insure that no program tries to SEVER a path that another program established, each individual IUCV and APPC/VM path has a NAME associated with it. When a program requests a CONNECT or ACCEPT function, the NAME specified becomes the owner of this path. If the program requests a SEVER or an ACCEPT for a specific path and the NAME specified does not correspond with the owner of that path, the SEVER or ACCEPT is not permitted.

2. The HNDIUCV macro must be issued to identify the program to CMS before issuing the CMSIUCV macro.

3. If the program requests a SEVER function with CODE=ALL, all IUCV and APPC/VM paths owned by that program are SEVERed. The IPUSER field of the IUCV SEVER PRMLIST is set to binary ones. The CP parameter list passed as input on the CMSIUCV SEVER, CODE=ALL must be an IUCV parameter list. A non-zero return code is returned to the user program if an APPC/VM parameter list is passed. Any APPC/VM paths that the user owns will be severed as TYPE=ABEND with CODE=X'0620'.

4. IUCV and APPC/VM generate exceptions for some error conditions. If IUCV or APPC/VM generates an operation, specification, or addressing

exception while a HNDIUCV or CMSIUCV macro is executing, control does not directly return to the next sequential instruction. Instead, a program check is generated.

5. The HNDIUCV REP function will only replace the general exit address and/or UWORD set up by your program via the HNDIUCV SET function. If your program had previously issued any CMSIUCV CONNECTs and had the EXIT address or UWORD default to the HNDIUCV SET's EXIT and UWORD, the HNDIUCV REP function does not replace the path specific EXIT or UWORD set up via the CMSIUCV function. The EXIT and UWORD remain as established when the CMSIUCV function was issued.

Error Conditions:

If an error occurs, register 15 contains one of the following return codes:

Code	Meaning
2	An APPC/VM parameter list was passed as input to CMSIUCV, and the requested function completed immediately. The function complete information is in the parameter list. The user's path-specific exit will not be driven because CP does not reflect an interrupt to the virtual machine.
3	An APPC/VM SENDDATA or RECEIVE function was requested, and completed immediately. CP stored error information in the IPAUDIT field of the CP APPC/VM parameter list. The user's path-specific exit will not be driven because CP does not reflect an interrupt to the virtual machine.
8	No HNDIUCV SET has been issued for this program (CONNECT,ACCEPT,SEVER)
12	The program doesn't own the path (ACCEPT,SEVER)
16	The NAME parameter was not specified or its address is equal to zero (CONNECT,ACCEPT,SEVER)
24	The PRMLIST parameter was not specified or its address is equal to zero (CONNECT,ACCEPT,SEVER)
28	An IUCV SEVER with the IPALL bit on is not allowed (SEVER)
40	Unrecognized function
52	The IUCV CONNECT parameter list is invalid. Only the CMS supervisor can specify CONTROL=YES.
68	An APPC/VM parameter list is not allowed as input on a CMSIUCV SEVER, CODE=ALL.

1xxx Indicates that an IUCV error occurred. "xxx" is the IPRCODE field that was returned by IUCV to aid in diagnosing the error. (CONNECT,ACCEPT,SEVER)

Exits

When the program's IUCV or APPC/VM external interruption routine is given control, all interruptions are disabled. The exit routine is responsible for providing proper entry and exit linkage for its IUCV or APPC/VM external interruption handling routine. The exit routine has the following requirements:

- The routine should not enable itself for any type of interrupts.
- The routine should not perform any I/O operations, since all interruptions are disabled.
- The routine must return control to the address in register 14.

When the routine receives control, the significant registers contain:

Register Contents

0 UWORD Field

1 Points to a SAVEAREA in the format:

Label	Displacement Dec	Displacement Hex	Contents
GRS	0	0	Control registers 0-15 at the time of the interrupt.
FRS	64	40	Floating point registers 0-7 at the time of the interrupt.
PSW	96	60	External Old PSW at the time of the interrupt.
UAREA	104	68	Register save area for exit routine's use.
END	176	B0	End of save area.

2 Address of the IUCV External Interrupt Buffer

13 Points to the save area at label UAREA for use by the exit routine

14 Return address

15 Entry point address

Usage Notes

1. If the CMS IUCV support is active, the external interrupt handler recognizes two error conditions.
   - An IUCV or APPC/VM pending-connect external interrupt occurs and the first eight bytes of the IPUSER field does not match any currently active CMS IUCV program's identity.

- Any other type of IUCV or APPC/VM external interrupts occurs and the path that it occurs on is not owned by any active CMS IUCV programs in the virtual machine.

In either condition, CMS issues an IUCV SEVER for the path in error. The 16 bytes in the IPUSER field contain binary ones (X'F').

2. If a CMS abend occurs, the CMS IUCV environment is terminated. An IUCV RETRIEVE BUFFER is issued, and any exits set up by the CMSIUCV or HNDIUCV macros are cancelled.

3. CMS IUCV clean up does not occur at end-of-command processing.

4. A program must be ready to handle any incoming external interrupts as soon as a HNDIUCV or CMSIUCV macro has finished execution. A program may even be interrupted before the next sequential instruction after the macro in the program is executed.

Using CMS IUCV to Communicate Between Two Virtual Machines

Figure 9 on page 213 illustrates the sequence of macro instructions issued when a virtual machine communicates with another virtual machine using CMS IUCV. With APPC/VM, this virtual machine to virtual machine communication can occur between two systems.

The functions performed by these instructions include:

- Initializing IUCV or APPC/VM communications
- Connecting to another virtual machine
- Sending and receiving messages
- Replying to and waiting for messages
- Severing connections with the other virtual machine
- Terminating IUCV or APPC/VM communications.

```
Virtual Machine X                          Virtual Machine Y

1. HNDIUCV SET,NAME=ONE,EXIT=A             1. HNDIUCV SET,NAME=TWO,EXIT=1
2. Set up the IUCV parameter list
3. CMSIUCV CONNECT,NAME=ONE,EXIT=B
                                           4. CONNECT-pending external
                                              interrupt
                                           5. EXIT 1 receives control
                                           6. CMSIUCV ACCEPT,NAME=TWO,EXIT=2
7. CONNECT-complete external
   interrupt
8. EXIT B receives control
   .                                          .
   .                                          .
   .                                          .
9. CMSIUCV SEVER,NAME=ONE                  10. SEVER external interrupt
                                           11. EXIT 2 receives control
                                           12. CMSIUCV SEVER,NAME=TWO
13. HNDIUCV CLR,NAME=ONE                   13. HNDIUCV CLR,NAME=TWO
    .                                          .
    .                                          .
    .                                          .
14. ONE   DC   CL8'RED'                        .
                                           15. TWO   DC   CL8'BLUE'
```

Figure 9. Sequence of Instructions in Virtual Machine to Virtual Machine Communication

The sequence of instructions shown in Figure 9 is an example of how to use IUCV in CMS. You can substitute APPC/VM instructions where IUCV instructions are shown. Note that the target program (NAME=TWO) must have established itself as a resource (to CP) before the source program (NAME=ONE) can request an APPC/VM connection to it. Refer to the *VM/SP Transparent Services Access Facility Reference* for details about how a resource identifies itself.

See Appendix B, "Sample CMS IUCV Program" on page 425 for an actual program using CMS IUCV.

The following list is an explanation of the sequence of instructions used above.

1. A program running in virtual machine X wishes to communicate with a program running in virtual machine Y. Each program must independently issue the HNDIUCV macro to begin IUCV communications. By issuing HNDIUCV SET, CMS invokes the IUCV DECLARE BUFFER function. The EXIT parameter establishes a general exit to handle IUCV CONNECT PENDING external interrupts.

2. Before issuing a CMSIUCV CONNECT, an IUCV CONNECT parameter list must be set up by the program. The IPVMID field of the IUCV parameter list contains the userid of the virtual machine you are connecting to (virtual machine Y). The first 8 bytes of the IPUSER field of the IUCV parameter list contains the eight-character identifying name of the program that issued a HNDIUCV SET in virtual machine

Y. This name must match the name specified on the HNDIUCV macro issued by the program in virtual machine Y. In this example, the first eight bytes of the IPUSER field equals BLUE.

3. The program in virtual machine X issues a CMSIUCV CONNECT to initiate a communication link with virtual machine Y. By issuing CMSIUCV CONNECT, CMS invokes the IUCV CONNECT function. This associates the exit address, "B", with the IUCV pathid.

4. Virtual machine Y receives a CONNECT-pending external interrupt as a result of the CMSIUCV CONNECT issued by the program in virtual machine X.

5. "EXIT 1" receives control as a result of the external interrupt. ("EXIT 1" receives control because it was specified on the EXIT parameter of the HNDIUCV macro.)

6. To complete the connection, the program in virtual machine Y issues a CMSIUCV ACCEPT. By issuing CMSIUCV ACCEPT, CMS invokes the IUCV ACCEPT function. This completes the IUCV communication link with virtual machine X. The CMSIUCV ACCEPT also associates the exit address, "2", with the pathid.

7. Virtual machine X receives a connection-complete external interrupt as a result of the CMSIUCV ACCEPT issued by the program in virtual machine Y.

8. "EXIT B" receives control as a result of the external interrupt. ("EXIT B" receives control because it is specified on the EXIT parameter of the CMSIUCV macro.)

9. Virtual machine X completed its communications with virtual machine Y and terminates the IUCV communication link. The program in virtual machine X issues an CMSIUCV SEVER to terminate this link. By issuing CMSIUCV SEVER, CMS invokes the IUCV SEVER function and clears the exit associated with the communication link.

10. Virtual machine Y receives a SEVER external interrupt as a result of the CMSIUCV SEVER issued by virtual machine X.

11. "EXIT 2" receives control as a result of the external interrupt. ("EXIT 2" receives control because it was specified on the EXIT parameter of the CMSIUCV macro.)

12. The program in virtual machine Y issues a CMSIUCV SEVER to terminate the communication link. By issuing CMSIUCV SEVER, CMS invokes the IUCV SEVER function and clears the exit associated with the communication link.

13. After all communications are complete and all communication paths have been SEVERed, the program in virtual machine X and the program in virtual machine Y independently issue HNDIUCV CLR.

HNDIUCV CLR terminates IUCV communications and clears the general exit for IUCV PENDING CONNECTs. CMS invokes the IUCV RETRIEVE BUFFER function if there are no other programs in the virtual machine using IUCV.

14. This is the label specified in the NAME parameter. This location contains the identifying name of the program in virtual machine X. The name of this program is RED.

15. This is the label specified in the NAME parameter. This location contains the identifying name of the program in virtual machine Y. The name of this program is BLUE.

Guidelines and Limitations of the CMS IUCV

Some IUCV and APPC/VM functions affect the IUCV and APPC/VM environment of the entire virtual machine. Since CMS cannot intercept any IUCV or APPC/VM functions directly issued by a program, any program using CMS IUCV has certain limitations on its use of IUCV and APPC/VM functions. The program must not issue any IUCV or APPC/VM function that interferes with the operation of other IUCV or APPC/VM applications running in the virtual machine.

The following is a list of IUCV and APPC/VM functions. The list describes their relationship to the CMS IUCV and some guidelines for their use. If any functions listed as "Should not be used..." are indeed used, other programs using CMS IUCV functions in the virtual machine may be affected. For information on coding the following functions, see the Chapter 2, "Inter-User Communications Vehicle" on page 111 for IUCV functions, and the *VM/SP Transparent Services Access Facility Reference* for APPC/VM functions.

The functions that only exist in IUCV are noted, as are the functions that only exist in APPC/VM.

ACCEPT
Is invoked by a program via the CMSIUCV macro. It should not be issued directly by a program.

CONNECT
Is invoked by a program via the CMSIUCV macro. It should not be issued directly by a program.

DECLARE BUFFER
Is used by HNDIUCV to initialize the virtual machine's IUCV environment. It should not be issued directly by a program.

DESCRIBE
Should not be used because this function clears the pending-message external interruption for the described message. This interrupt may not belong to the issuer of the DESCRIBE function. Thus, other

programs running in the same virtual machine may be affected since the message is lost and never reflected to the true target.

PURGE (IUCV only)
Is issued directly by a program.

QUERY
Is used by HNDIUCV to determine the size of the external interrupt buffer and the maximum number of connections for this virtual machine. It may be issued directly by an application program.

QUIESCE (IUCV only)
Is issued directly by a program to quiesce a specific path. However, the issuer must be careful that the IPALL bit is not turned on in the IPFLAGS1 byte of the parameter list. This would quiesce all paths in the virtual machine.

RECEIVE
Is issued directly by the application program. However, the issuer must be careful that a specific message id or path id is specified in the IUCV parameter list. If it is not, IUCV RECEIVEs the first message that has not yet been partially received for the entire virtual machine. This message may not belong to the program that issued the IUCV RECEIVE.

REJECT (IUCV only)
Is issued directly by a program.

REPLY (IUCV only)
Is issued directly by a program.

RESUME (IUCV only)
Is issued directly by a program in order to resume a specific path. However, the issuer must be careful that the IPALL bit is not turned on in the IPFLAGS1 byte of the parameter list. This would resume all paths in the virtual machine.

RETRIEVE BUFFER
Is used by HNDIUCV and CMS abend processing to terminate the virtual machine's IUCV environment. It should not be issued directly by a program.

SEND (IUCV only)
Is issued directly by a program.

SENDCNF (APPC/VM only)
is issued directly by a program.

SENDCNFD (APPC/VM only)
is issued directly by a program.

SENDDATA (APPC/VM only)
is issued directly by a program.

SENDERR (APPC/VM only)
is issued directly by a program.

SENDREQ (APPC/VM only)
is issued directly by a program.

SET MASK
Should not be used because this function disables certain IUCV external interrupts for the entire virtual machine. Thus, other programs running in the same virtual machine may be affected.

SET CONTROL MASK
Should not be used because this function disables certain IUCV external interrupts for the entire virtual machine. Thus, other programs running in the same virtual machine may be affected.

SEVER
Is invoked by a program via the CMSIUCV macro. This IUCV function may be invoked to SEVER all existing paths for the CMS IUCV program that has issued the HNDIUCV CLR macro. This IUCV function should not be issued directly by a program.

TEST COMPLETION
Is issued directly by a program. However, the issuer must be careful that a specified message id or path id is specified in the IUCV parameter list. If it is not, IUCV completes the first message on the REPLY queue for the entire virtual machine. This message may not belong to the application that issued the TEST COMPLETION.

TEST MESSAGE
Should not be used because this function places the entire virtual machine in a wait state if no incoming messages or replies are pending. Thus, other programs running in the same virtual machine may be affected.

Chapter 4. SNA Virtual Console Communication Services

SNA Virtual Console Support provides full VM console capabilities to terminal operators on SNA terminal devices and allows the VM user to use SNA terminals as virtual operator consoles.

SNA Virtual Console Communications Services support the following console functions:

- CP/CMS command processing capabilities
- System product or CMS editor processing mode
- Full screen support for 3270 type terminal devices
- Support for 3290 type terminal devices
- Data stream processing for TWX devices, including APL/ASCII support and display roll-over presentation.

This support is provided through a VTAM service machine (VSM) that acts as an interface between an SNA network and CP. The VSM passes data between the SNA network and the SNA Console Communications Services (CCS) feature of VM/SP. CCS passes data between existing non-SNA CP system console services and the VSM.

The VSM can be a virtual machine running either the VTAM SNA Console Support component (VSCS) of the Advanced Communication Function/Virtual Telecommunications Access Method (ACF/VTAM) or the VM/VTAM Communications Network Application (VM/VCNA).

The screen management services are divided between the SNA CCS and either VSCS or VCNA. VSCS or VCNA is responsible for the physical screen management and therefore, the device dependent characteristics. Thus, VSCS or VCNA handles such things as screen size and redisplay of the input line at the terminal. SNA CCS is responsible for logical screen management and thus remains device independent in most cases. SNA CCS also passes the terminal input to CP and reflects status and actions to and from the rest of the CP system.

System Structure

Figure 10 illustrates a VM system with the SNA virtual console support. The VTAM service machine (VSM) consists of VTAM and VSCS running under the control of VM's Group Control System (GCS).

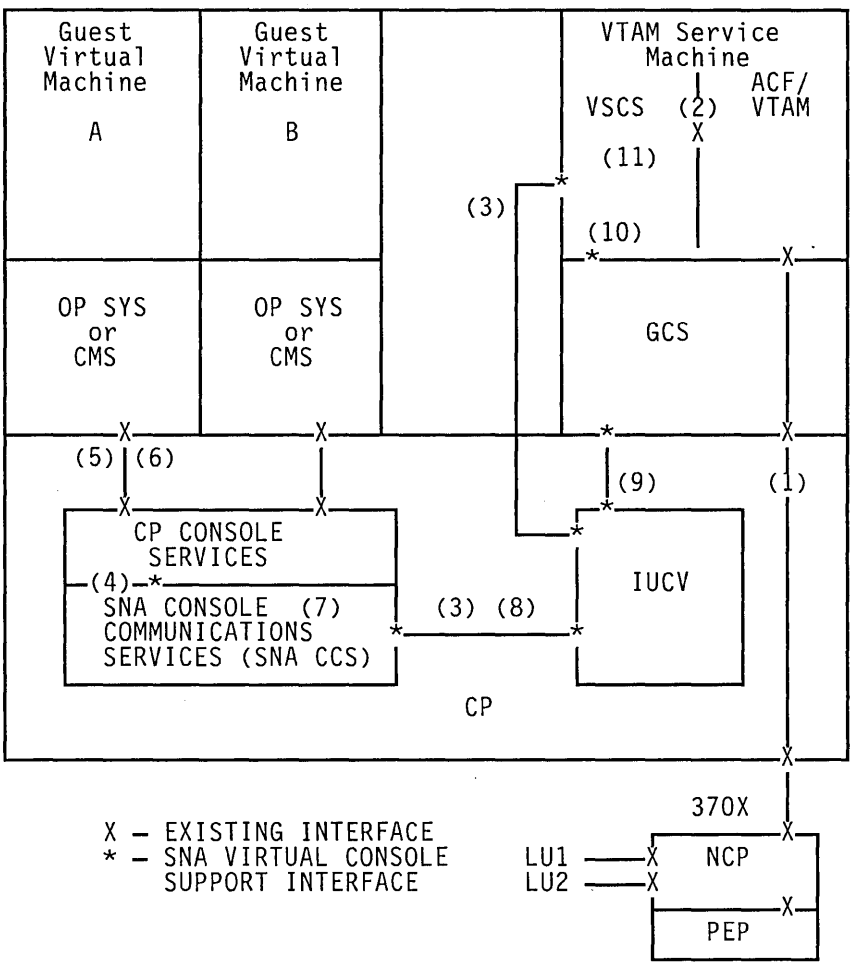


Figure 10. Virtual Console Support in CP

1. SNA CCS supports the SNA terminals (LU1, LU2) as virtual consoles. These SNA terminals are attached to a 3705 or 3725 communications controller dedicated to the VTAM service machine.

   Data entered at the terminal goes through its normal path of the NCP, CP, GCS, and VTAM. The guest virtual machine interface to CP is the S/370 architecture provided by virtual machine simulation.

2. VSCS interfaces to VTAM through the standard Application Program Interface (API) to perform physical I/O to/from the SNA devices.

3. The terminal input is communicated to the SNA CCS through the Inter-User Communication Vehicle (IUCV) SEND, RECEIVE, and REPLY protocols.

4. SNA CCS receives the interface control block (Work Element Block) with the terminal input data. It interprets the control information that describes the screen environment and the user's actions, and determines the action to be reflected to CP. SNA CCS edits the input line, and passes it to CP along with the action required.

5. SNA CCS either processes the line or sends it to the guest virtual machine for processing.

6. Guest virtual machines request console I/O through the Start I/O interface (SIO) or through DIAGNOSE code X'58'.

7. SNA CCS intercepts the I/O request and performs logical screen management. A Work Element Block is built to inform VSCS of the action to be initiated on the screen and to hold the output line.

8. SNA CCS uses the IUCV SEND, RECEIVE, and REPLY protocols to communicate the work transaction to VSCS.

9. The IUCV request from SNA CCS to VSCS in the VTAM Service Machine is intercepted by GCS.

10. GCS notifies VSCS of the incoming message.

11. VSCS receives the Work Element Block, interprets the orders, and performs the physical screen management for the SNA terminal.

Environments Supported

SNA CCS and either VSCS or VCNA handle three 'environments' for screen management: console mode, CMS mode, and full screen support mode. These environments represent the interfaces that CP supports for console services to a virtual user terminal and a guest virtual machine (GVM).

1. Console mode is communications between a display operator and either CP or an operating system in a virtual machine (CMS or another operating system). In this mode, the screen is divided into three areas, (input, output, and status), and data to the output area is always directed to the next available line. Console mode I/O is generated when a guest virtual machine issues an SIO to the 3215 user console or CP generates console I/O requests internally in response to CP commands.

2. CMS mode is DIAGNOSE code X'58', CCW op code X'19' transactions. In this mode, the CMS editor or an application program directs output to specific lines on the screen. As with console mode, the screen is divided into three areas (input, output, and status).

3. Full screen support mode (FSSM) is the environment where the display screen is under control of a full screen application program. In this mode, the format of the screen is under application program control and the application program provides all 3270 orders. The interface to CP from a guest virtual machine is DIAGNOSE code X'58', CCW op code X'29' or X'2A', for a full screen write or read.

Processing Descriptions

Screen Management

In non-SNA processing, DMKGRF handles the console support for local 327x, 3066, and 3290 devices. DMKRGA, DMKRGB, DMKRGC, DMKRGD, and DMKRGE contain the support for remote devices. These modules perform both the logical and physical screen management needed for the graphic display and printer keyboard terminals.

In SNA processing, to support a virtual console for a VTAM service machine terminal user, virtual console support has been divided between the SNA CCS and either VSCS or VCNA.

Either VSCS or VCNA handles the physical, device-dependent characteristics of the screen, setting up the I/O, and maintaining the current state of the screen. VSCS or VCNA uses VTAM to perform the I/O. SNA CCS handles the logical control of the screen, directs the VSCS or VCNA actions, and serves as the interface between the VTAM machine and the existing CP console function support. SNA CCS communicates with VSCS or VCNA via IUCV.

The following modules perform the logical functions for CP SNA CCS that are described above:

- DMKVCP
- DMKVCQ
- DMKVCR
- DMKVCS
- DMKVCT
- DMKVCU
- DMKVCV
- DMKVCW
- DMKVCX

As with non-SNA processing, DMKGRF processes the local 327x/3066 and 3290 devices, and DMKRGA, DMKRGB, DMKRGC, DMKGRD, and DMKRGE support the remote devices.

Note that the logical units supported by VSCS or VCNA are independent of CP; they cannot be mapped to any real device defined to CP (that is, they are not defined in the RDEVICE macro). SNA CCS provides the necessary interface to make the SNA terminal appear to be a real CP device.

Communication Interfaces

To communicate, SNA CCS and VSCS or VCNA pass a work element block (WEBLOK) between them. The WEBLOK contains the transaction orders for the other component (SNA CCS or either VSCS or VCNA), the environment, and the data for the CP system or the user's terminal. See the section "Work Element Block" that follows or see *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for a detailed description of the WEBLOK.

SNA CCS and either VSCS or VCNA communicate via the Inter-User Communication Vehicle (IUCV). Figure 11 on page 224 illustrates the interfaces used in SNA processing. DMKQCN presents requests from CP, CMS or, a guest virtual machine for terminal writes to SNA CCS via CONTASKS. DMKQCO presents requests from CP, CMS or, a guest virtual machine for terminal reads to SNA CCS via CONTASKS. SNA CCS passes input from the SNA terminal to CP and the virtual machines via DMKCFM and DMKVCN. This is the same way DMKGRF handles local terminal support.

In SNA processing, CP handles terminal input and interfaces normally with one exception: CP must use logical unit names, instead of real addresses, to reference SNA terminals.

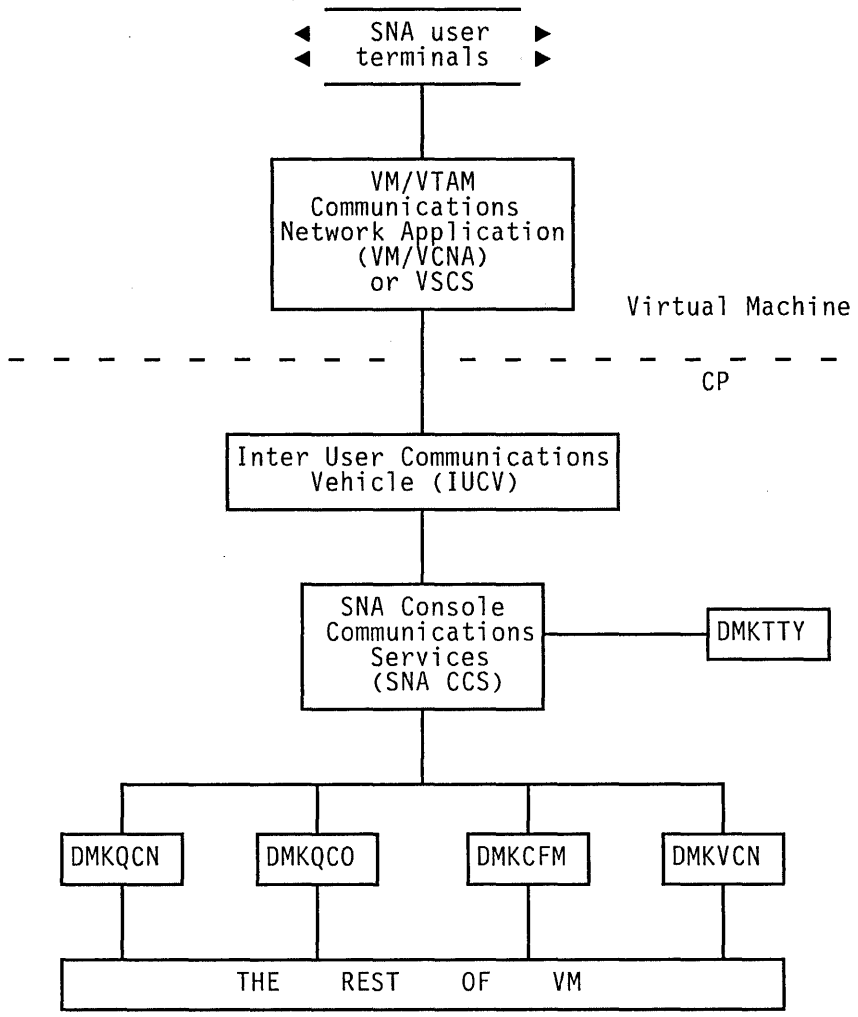


Figure 11. SNA Virtual Console Support Interfaces

Functions

SNA CCS handles the following functions in support of the console, CMS, and full screen mode environments for SNA terminals:

- Connect VTAM service machine and Logical Units
- Logon a Logical Unit
- Request a read
- Request a write
- Process an enter key
- Process a PA1 key
- Process a PA2 key
- Process a PA3 key
- Process a PF key
- Process an Attention Interrupt
- Process a Cursor Back One
- Logoff a Logical Unit

- Process error conditions
- Sever a communications path
- Process data streams from TTY devices
- Pass VSCS the system identification for the status area.

Enabling SNA Terminals

The CP operator must issue the ENABLE SNA command. The SNA parameter on the ENABLE command enables all SNA devices and has no effect on non-SNA devices. The ENABLE ALL command enables both non-SNA and SNA devices.

In the multiple VTAM service machine environment, the operator may selectively enable or disable any given VTAM service machine by using the userid option on the ENABLE/DISABLE SNA command. The operator cannot enable and disable individual logical units, although the VTAM operator may.

Establishing Communications Links

The VTAM service machine issues an IUCV CONNECT under two separate conditions:

- VTAM Service Machine CONNECT

  VSCS or VCNA issues a CONNECT via IUCV to establish an initial global connection between VSCS or VCNA and SNA CCS. This CONNECT notifies SNA CCS that a new VTAM service machine has logged on and is ready to service logical units. If the VM operator has issued the ENABLE SNA or ENABLE ALL command, SNA CCS accepts the CONNECT, and authorizes VSCS or VCNA to allow users to logon to SNA terminals. SNA CCS creates a VTAM Service Machine Block (VSMBLOK) for that VTAM service machine. In a multiple VTAM service machine environment, the VSMBLOK allows SNA CCS to associate the logged on SNA user with the correct VSCS or VCNA. See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for a detailed description of the VSMBLOK.

- Logical Unit CONNECT

  To logon to VM, the SNA terminal user must first logon to VSCS or VCNA running in the VTAM service machine. To logon to VSCS or VCNA, the user issues the ACF/VTAM LOGON command. When logging on, the terminal user may optionally specify the userid (or the userid and password) of his virtual machine in the DATA portion of the ACF/VTAM LOGON command. Specifying information in DATA allows you to reach VM in one step. The commands for logging on differ slightly between VSCS and VCNA. The examples below show VM information included on a VSCS or VCNA logon:

```
LOGON APPLID(VM)
LOGON APPLID(VM) DATA (userid)
LOGON APPLID(VM) DATA ('userid options')
```

If you specify VM in the APPLID field, ACF/VTAM queues a logon request to VSCS.

The syntax for LOGON commands with VCNA is similar, but the APPLID name must be "VCNA" instead of "VM". VSCS also allows a shorter form of the LOGON command:

```
VM
```

The userid and password may be included on the "VM" form.

VSCS also allows terminals it supports to use the CP DIAL command. It can be included as DATA on a "LOGON" or "VM" command. Examples are:

```
LOGON APPLID(VM) DATA ('DIAL userid')
VM DIAL userid
```

Any CP command valid before a CP logon can be used as data. Only display terminals can use the DIAL command; keyboard/printer terminals may not.

If no logon data (VM userid and password) is specified, the system writes a Vitual Machine/System Product logo (VM/SP or VM/SP HPO logo) to the terminal under the control of VSCS or VCNA. From this point on, the user logs on to VM much the same as he does with a local terminal. The attention interrupt generated when the user clears the VM/SP or VM/SP HPO logo from the screen causes VSCS or VCNA to issue an IUCV CONNECT SVC for the terminal. If SNA is still enabled, CCS builds an RDEVBLOK and a SNA Resource Block (SNARBLOK) and chains them to the VSMBLOK built during the VSCS or VCNA connect described above. This ties the user's control block (SNARBLOK) to the VTAM service machine the user is logged onto. The system must tie these blocks together since the logical unit's LUNAME, which is represented by the SNARBLOK, is unique only to its own VTAM service machine. That is, it is possible to have duplicate lunames among two or more VTAM service machines.

After the connection is established, VSCS or VCNA and SNA CCS exchange initialization information. VSCS or VCNA sends luname, device class, device type, line length, pace value (for controlling the number of writes to the screen), model number, and its IUCV path ID for this logical unit and then waits for LOGON processing to complete. SNA CCS initializes the SNARBLOK and RDEVBLOK with the data supplied by VSCS or VCNA. VSCS also exchanges some additional information with SNA CCS. It passes the information from a Write Structured Field Query to SNA CCS and receives the VM system id from SNA CCS.

If the user specified a userid and password on the ACF/VTAM LOGON, the VM/SP or VM/SP HPO logo is not displayed. VM/VCNA sends the logon data to SNA CCS in response to the first CP read request to enter

userid. If the user specified only the userid, CP prompts the terminal user for the password.

The installation may specify "automatic" logon to VSCS or VCNA for SNA terminals. This can be accomplished in two ways:

1. The installation can specify LOGAPPL=(VM) or LOGAPPL=(VCNA) in the logical unit definition. This causes ACF/VTAM to queue a logon request to the appropriate program when the logical unit is started.

2. The ACF/VTAM operator may issue a VARY ACTIVATE command for the logical unit, specifying VSCS or VCNA on the LOGON= parameter.

For further information concerning ACF/VTAM LOGON refer to the *ACF/VTAM System Programmer's Guide* or the *VM/SP Terminal Reference.*

DIAL Command Processing

A VM/SNA user whose display is controlled by VSCS may issue the CP DIAL command to establish communication with an IPLed guest virtual machine. When the DIAL command is issued, DMKDIA tries to locate the VMBLOK in the target virtual machine. If unsuccessful, DMKDIA issues a message to the user indicating that the target virtual machine is not logged on. A VM/VTAM terminal is not allowed to dial to the virtual machine running VTAM. The user receives an error message when the user tries to dial the VTAM virtual machine.

If DMKDIA does locate the desired VMBLOK, it examines the VDEVBLOKs to determine if the virtual machine has defined a local virtual graphic terminal. If not, DMKDIA issues a message to the user indicating that no lines are available. When a user tries to connect to a specific virtual address and DMKDIA cannot locate a VDEVBLOK for that address, the user receives a message indicating that the specified address does not exist.

When the necessary conditions are satisfied, DMKDIA sets flags in the target virtual machine control blocks to show the following:

- VDEVBLOK - dialed SNA device
- SNARBLOK - dialed SNA terminal
- RDEVBLOK - dedicated device.

DMKDIA then stacks an IOBLOK for the virtual machine on behalf of the device to signal a DEVICE END from a virtual power on.

DMKDIA issues a message to the user giving the virtual address of the dialed virtual machine. It also sends a message to the operator saying that this user is dialed to the target virtual machine.

The DIAL session is terminated when the virtual terminal device is reset by the virtual machine. DMKDIB then drops the device from the virtual machine, and issues a related message to the user and to the operator. CCS (Console Communications Services) cleans up SNA control blocks and CP resets all flags set during session initiation. This frees the virtual device to be used by another user.

Real Device Simulation

When VSCS or VCNA connects to SNA CCS for a logical unit, it identifies the SNA logical unit to SNA CCS. Also, VSCS or VCNA identifies any device characteristics that CP or CMS needs to perform their functions. SNA CCS simulates a real device by dynamically building a Real Device Block (RDEVBLOK) and assigning this RDEVBLOK to the SNA user's virtual machine.

SNA CCS initializes the fields for the RDEVBLOK instead of DMKRIO. Also, SNA CCS builds a control block for SNA, a SNARBLOK. The SNARBLOK contains the status and control fields for SNA CCS. See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for a detailed description of the SNARBLOK.

The RDEVBLOK is chained to the VSMBLOK belonging to the VSM that issued the IUCV CONNECT for it, and the RDEVBLOK points to the SNARBLOK for that LU. The RDEVBLOK and SNARBLOK are, however, contiguous in storage. CP references to the RDEVBLOK are still valid in the SNA environment.

As in non-SNA processing, the VMTERM field of the VMBLOK and the VDEVREAL field of the VDEVBLOK point to the RDEVBLOK.

When special SNA processing is necessary, an indicator in the RDEVBLOK (RDEVSNA) denotes that this is a SNA type RDEVBLOK. The RDEVSNA field is an alternate definition for the current RDEVADD field. The real device address has no meaning for SNA logical units.

Command Handling

After VSCS or VCNA completes the initial processing for the SNA logical unit, it passes the user's LOGON request to SNA CCS. SNA CCS edits the LOGON command and all subsequent commands and passes them to CP console services using CP interfaces. CP processes the commands the same way it processes non-SNA commands. However, VSCS or VCNA, rather than CP, manages redisplay of the input line.

Work Element Block

The work element block serves as the interface between SNA CCS and VSCS or VCNA. Both SNA CCS in CP and VSCS or VCNA in the VTAM service machine create work element blocks. In SNA CCS, the work element block is known as the WEBLOK. In VSCS or VCNA, the work element block is known as the DTIWEB. SNA CCS and VSCS or VCNA pass the WEBLOK between them and use it as the interface for all requests for work from the other component. The data portion of the work element block contains input or output lines to be passed and the control portion contains transaction orders and environment data. See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for a detailed description of the WEBLOK.

Work Element Indicator Block

SNA CCS creates the work element indicator block (WEIBLOK) as a header for the WEBLOK. Its function is to identify a unit of work that is in progress or that has not yet been processed. The WEIBLOK points to the WEBLOK and CONTASK associated with a given user. See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for a detailed description of the WEIBLOK.

SNA I/O Processing

For non-SNA processing, CP console services build channel programs, IOBs, and use DMKIOS to perform their I/O. SNA processing moves the physical device management to VSCS or VCNA. Instead of calling DMKIOS, CP then passes control to SNA CCS. SNA CCS does not build any channel programs or IOBs. It determines what action must be taken for the console and sends the transaction to VSCS or VCNA instead of to DMKIOS. VSCS or VCNA and VTAM set up the I/O operations to the terminal and issue an SIO. CP intercepts this SIO and performs the I/O the same way it does for non-SNA processing.

Redisplay of Input Line

Input line redisplay for SNA terminals is handled by VSCS or VCNA.

- VSCS or VCNA redisplay of input line

  To reduce the number of VTAM SENDs to the terminal, VSCS or VCNA does not immediately redisplay the input line. Instead, it holds the I/O operation until SNA CCS requests more I/O to that terminal; for example, the response to the input or a message. When VSCS or VCNA receives a write to the device, it sends the input line to be redisplayed and the information from SNA CCS to be written to the device in one VTAM send.

  VSCS or VCNA clears the input area, updates the status field, and redisplays the line using the same VTAM SEND.

- CCS Redisplay Timer

  VSCS or VCNA passes a timer variable to SNA CCS when it invokes the IUCV CONNECT function. This value indicates to SNA CCS how long it should wait for a command to complete before SNA CCS issues an IUCV SEND to the VTAM service machine to request input line redisplay.

  SNA CCS sets a timer to tell it how long to wait before requesting redisplay of the line. This is necessary since some commands do not produce any output, and CP or CMS might require a significant amount of time to finish the command processing. If the timer expires before CP has output to write to that terminal, SNA CCS issues an IUCV SEND to VSCS or VCNA requesting a write for the redisplay.

TRQBLOK

In non-SNA processing, DMKGRF builds a Timer Request Block (TRQBLOK) that it uses

- For its status flags
- For an interrupt return address after an I/O operation
- After a timer expires and
- As a header to chain CONTASKS when in FSS mode.

In SNA processing, SNA CCS does not use TRQBLOK for the above functions because:

1. Status fields are kept by VSCS or VCNA for each SNA terminal user

2. IUCV mechanisms are used to return control after SEND requests to VSCS or VCNA

3. The timer support for the alarm, MORE/HOLDING state, and NOT ACCEPTED has been moved to VSCS or VCNA

4. SNA CCS has its own control block structure to associate a user with its related CONTASKS, IUCV control blocks, and the work element block. Since the processing of CONTASKs has been streamlined to help performance, the TRQBLOK is no longer needed for queueing CONTASKs.

However, in SNA processing, a TRQBLOK is still created, since a timer is required for the input line redisplay processing described above.

I/O REQUESTS

DMKGRF, the module that manages I/O to a real 3270, performs requests for I/O from DMKQCN synchronously. When DMKQCN requests a response, DMKGRF schedules an IOB for the I/O operation and waits for the I/O to complete before sending the response. SNA CCS takes the virtual machine out of SIO wait state as soon as the I/O to the real device is started. For a write, SNA CCS sends the write request to VSCS or VCNA,

takes the virtual machine out of the SIO wait state, and returns immediately to the caller with a successful completion response as if the I/O had completed successfully.

In some situations, SNA CCS waits for a response from VSCS or VCNA before responding with a return code to the CP system. This is governed by a 'pacing value' equivalent to the number of lines for a full screen. In this way, VSCS or VCNA can reach SNA CCS with a PA1 key indicator to stop processing; SNA CCS does not flood IUCV and VSCS or VCNA with output from some commands (for example, DISPLAY). SNA CCS always waits for a response from VSCS or VCNA for DIAGNOSE code X'58' writes for CMS and full screen support modes before returning to the caller with a response.

SNA CCS queues a CONTASK if it is waiting for a response on either a write or a read request. It does not split CONTASKs for multiple line writes but passes the entire write buffer to VSCS or VCNA, thus reducing IUCV SENDs and RECEIVEs.

VTAM I/O Reduction

SNA CCS batches console function and virtual machine SIO output lines in a 1K byte buffer. The batch lines are sent to VSCS or VCNA when the buffer is full, a read is initiated by a virtual machine or CP to a SNA terminal, the pace value reaches zero, the redisplay timer expires, a DIAGNOSE code X'58' operation takes place, or the virtual machine is dropped from the dispatch queue.

The batching technique and priority structure ensures that either a full screen of information is presented to VSCS or VCNA for each CP or CMS console transaction or the transaction is complete (for those transactions with less than a full screen of data) before control is given to the VSM.

MORE/HOLDING Condition

To reduce the number of IUCV transactions, VSCS or VCNA resolves the MORE/HOLDING condition when it occurs on the screen. VSCS or VCNA takes whatever action is appropriate and avoids notifying SNA CCS of the screen status in most cases. In cases when a mode change may take place (PA1 key) or an interrupt must be reflected to a user's virtual machine (PA1 or PA2 key), VSCS or VCNA resolves the MORE/HOLDING condition, then notifies SNA CCS of which key was pressed and the screen status at the time. VSCS or VCNA resolves pressing of the clear key or enter key in MORE/HOLDING status without notifying SNA CCS.

SNA Accounting

VSCS or VCNA records accounting data on a terminal user basis. When the SNA user logs off, VSCS or VCNA passes a maximum of 62 bytes of accounting data, in the WEBLOK, to SNA CCS. SNA CCS uses the CP accounting module, DMKACO to write a VTAM accounting record (type X'07') in the CP accounting file. Neither SNA CCS nor DMKACO are aware of the contents of the VTAM accounting record.

SNA CCS accrues processor time for a terminal user while it is processing for that user. This time is added to the time CP already accumulated for the user. The time appears in the accounting record produced when the user logs off.

Note: Refer to *VTAM Customization* for details of VSCS records and *VM/VCNA Installation, Operation, and Terminal Use,* for details of VCNA records.

NCP and PEP Sharing

Since CP supports only a back level of NCP that does not support SNA and VTAM loads ACF/NCP, you must prevent CP from loading/reloading the back-level NCP at initialization and at restart. Refer to the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference* for information on how to accomplish this.

User Termination

When a user issues the VM LOGOFF or a ACF/VTAM LOGOFF, the control blocks related to the user's virtual machine and SNA terminal are released to free storage. When VSCS or VCNA issues the SEVER indicating that a user has logged off, SNA CCS need only issue a SEVER for its path. If a SEVER reaches SNA CCS and there is a SNARBLOK that shows the user is disconnected, the path is severed. The control blocks are released when the user is eventually logged off. If SNA CCS gets the SEVER and there is a SNARBLOK but the user is not disconnected, then SNA CCS disconnects the virtual machine associated with the SNARBLOK.

Shutdown

To shutdown the system, the VM system operator should notify users that the system is shutting down. If the SNA operator has the proper class, he can force off any SNA user that did not log off. In this way, VSCS or VCNA can collect accounting data for its users and record it in CP. The DISABLE SNA (userid) command can be used to prevent additional users from logging on. In this way, VSCS or VCNA can be stopped without bringing down the VTAM service machine that it is running in. Any other application in the VTAM service machine may continue to run unaffected.

Operator Considerations

While it is possible for the operator of the VTAM service machine to disconnect from a local terminal, extreme care must be exercised. The VSCS or VCNA operator must issue the command SET RUN ON before disconnecting from the local terminal. If the operator does not do this, unpredictable results occur and a deadlock of the VSCS or VCNA is likely.

The operator of the VTAM service machine (VSM) cannot disconnect from the service machine and then reconnect from a SNA logical unit controlled by that service machine, using the same operator userid that was used for the service machine. That is, the operator cannot logon as the VSCS or VCNA operator at a terminal managed by VSCS or VCNA. The operator of

one VSM (i.e. VSM1) may disconnect from that VSM and reconnect as operator (of VSM1) from a terminal controlled by a second VSM (VSM2). The restriction means that the operator of VSM1 may not have as his terminal, one which is controlled by VSM1. Also, the userid that is running VTAM cannot logon at a terminal controlled by the virtual machine running VTAM.

SNA CCS Entries in CP Internal Trace Table

SNA Console Communications Services (SNA CCS) creates trace table entries in the CP Internal Trace Table to leave an audit trail of its activities.

SNA CCS places an entry in the CP trace table for each inbound transaction; SNA CCS creates a trace table entry for each outbound transaction after going to IUCV to communicate the entry to VSCS or VCNA.

The entry identifies the type of IUCV transaction, the SNA user that initiated the transaction, and the pertinent characteristics of the transaction environment itself. The transaction can be correlated throughout the system by using the CCS and VSCS or VCNA path id's and the IUCV message id. These fields can be matched with corresponding or similar fields in the IUCV trace elements in CP and VSCS or VCNA trace elements in VTAM.

For an error trace, SNA CCS places an entry in the CP trace table for logical errors and errors on IUCV transmissions. If the WEBLOK that is passed between SNA CCS and VSCS or VCNA is invalid, the data in the trace element pertains to the invalid WEBLOK.

Trace Table Entry Formats

The following tables show the formats of trace table entries created by SNA CCS.

ACCEPT (X'00') (VTAM service machine and Logical Unit), CONNECT for Logical Unit (X'12')

	0	1	2	3	4	5	6	7
0	X'16'	TRATNTYP	///////////////		TRAVCSPA		//////////////	
8	TRAUDATA							

PURGE (X'03')

	0	1	2	3	4	5	6	7
0	X'16'	TRATNTYP	TRAMODE	//////	TRAVCSPA		//////////	
8	TRAINSTR				TRAVMADR			

RECEIVE (X'04'), REPLY (X'06'), SEND 1WAY (X'08'), SEND 2WAY (X'09'), LOGIC ERROR in CCS WEBLOK (X'0B'), LOGIC ERROR in VCNA or VSCS WEBLOK (X'13')

	0	1	2	3	4	5	6	7
0	X'16'	TRATNTYP	TRAMODE	TRALGAID	TRAVCSPA		TRAVSAPA	
8	TRAFUNCT	TRACPSAF	TRAEDCHR	TRACHAR	TRAIXBLK			

SEVER (X'0A')

	0	1	2	3	4	5	6	7
0	X'16'	TRATNTYP	TRAUSER1	//////	TRAVCSPA		//////////	
8	TRAUDATA							

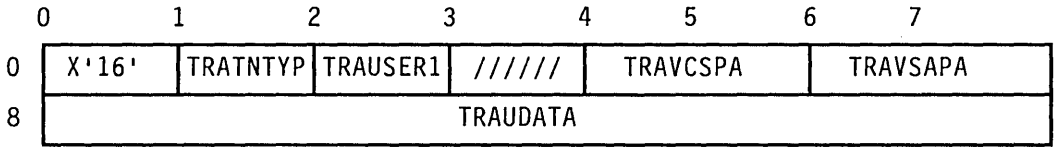
REPLY from VSCS or VCNA (X'0C')

	0	1	2	3	4	5	6	7
0	X'16'	TRATNTYP	TRAMODE	TRALGAID	TRAVCSPA		TRAVSAPA	
8	TRAFUNCT	TRACPSAF	TRAUDIT1	TRAUDIT2	TRAIXBLK			

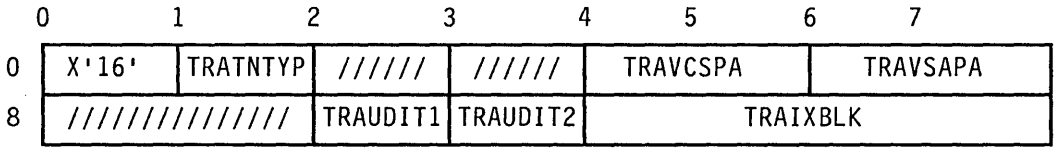
CONNECT for VTAM service machine (X'0E')

	0	1	2	3	4	5	6	7
0	X'16'	TRATNTYP	TRATIMER		TRAVCSPA		TRAMSGLM	
8	TRAUDATA							

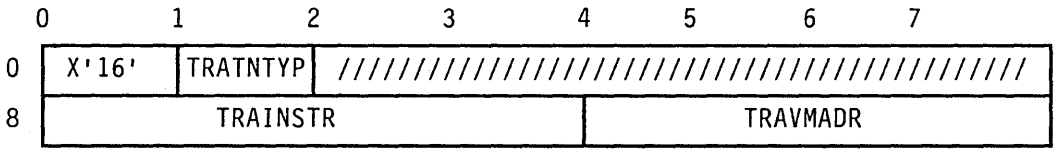
SEVER from VSCS or VCNA (X'10')



MESSAGE COMPLETE (X'11')



ABEND 02 (X'15')



Trace Table Entry Field Definitions

TRATNTYP
Indicates the type of transaction that the trace table entry is for.

Note: The high-order bit when on indicates a nonzero return code from IUCV on this transaction. DMKVCXFU writes the trace table entry. The IUCV return code (IPRCODE) is in TRAIPRCD, a one-byte field in the fourth byte of the trace entry.

Values Defined for TRATNTYP

TRACCEPT	X'00'	ACCEPT
TRACNECT	X'01'	CONNECT (not used)
TRAQUISC	X'02'	QUIESCE (not used)
TRAPURGE	X'03'	PURGE
TRARCEIV	X'04'	RECEIVE
TRAREJCT	X'05'	REJECT (not used)
TRAREPLY	X'06'	REPLY
TRARESUM	X'07'	RESUME (not used)
TRASEND1	X'08'	SEND 1 WAY

TRASEND2	X'09'	SEND 2 WAY
TRASEVER	X'0A'	SEVER
TRAVCSLE	X'0B'	LOGIC ERROR IN CCS WEBLOK
TRAVSARP	X'0C'	REPLY FROM VSCS OR VCNA
TRAVSAQS	X'0D'	QUIESCE (not used)
TRAVSMCN	X'0E'	CONNECT FOR VTAM service machine
TRAVSARM	X'0F'	RESUME (not used)
TRAVSASV	X'10'	SEVER FROM VSCS OR VCNA
TRAVSAMC	X'11'	MESSAGE COMPLETE FROM VSCS OR VCNA - 1 WAY SEND
TRALUCON	X'12'	CONNECT FROM VSCS OR VCNA FOR LU
TRAVSALE	X'13'	LOGIC ERROR IN VSCS OR VCNA WEBLOK
TRAERRSV	X'14'	ERROR IN USER ENVIRONMENT-SEVER USER (not used)
TRACTLBK	X'15'	SNA CONTROL BLOCK CHAIN INVALID

TRAMODE
Mode for the transaction (see WEBLOK (WEBMODE)).

TRALGAID
Logical mapping of the Attention Identifier (AID) for inbound transactions to CCS (see WEBLOK (WEBLAID)). The field does not have meaning for outbound transactions to VSCS or VCNA.

TRAUSER1
First byte from the IUCV user data field

TRATIMER
Two bytes of timer value from the VSM CONNECT

TRAVCSPA
The IUCV path id that identifies the CCS side of the IUCV path for this transaction.

TRAVSAPA
The IUCV path id that identifies the VCNA side of the IUCV path for this transaction.

TRAFUNCT
Transaction to be performed (See WEBLOK (WEBFUN))

TRACPSAF
This field is WEBSAFLG on inbound transactions to CCS and WEBCPFLG on outbound transactions to VSCS or VCNA. (See WEBLOK (WEBFUN))

TRAEDCHR
Editing characteristics (See WEBLOK (WEBEDIT))

TRACHAR
Character set (see WEBLOK (WEBCHAR))

TRAIPRCD
IPRCODE from IUCV IPARML for IUCV return code processing

TRAIXBLK
Address of the IXBLOK constructed for this transaction

TRAMSGLM
IUCV message limit to be specified for CONNECT

TRAUDATA
IPUSER from IUCV external interrupt buffer
For inbound: QUIESCE, RESUME, CONNECT for LU
For outbound: ACCEPT

VM userid
For inbound: CONNECT for VTAM service machine
For outbound: SEVER

LUNAME
For inbound: SEVER

TRAUDIT1
IUCV IPAUDIT1 flags from IXBLOK (used for TRAVSARP,TRAVSAMC)

TRAUDIT2
IUCV IPAUDIT2 flags from IXBLOK (used for TRAVSARP,TRAVSAMC)

TRAINSTR
Address of last instruction issued before invoking abend routine (TRACTLBK) or RDEVBLOK address (for PURGE)

TRAVMADR
Current VMBLOK address-used for abend situations (TRACTLBK)

Chapter 5. The Message System Service

The Message System Service is a CP system service. It allows a virtual machine to read incoming messages and responses from CP, as opposed to displaying them on the terminal.

Establishing Communications with the Message System Service

"*MSG" is the assigned Message System Service userid. Communications are established with the Message System Service (*MSG) through IUCV. The IUCV DECLARE BUFFER function is invoked by the virtual machine to allow communications with IUCV, and the IUCV CONNECT function is invoked to establish the communications path to the Message System Service.

The types of messages that the virtual machine can receive are controlled by specifying the IUCV parameter on the CP SET command. For example, if a user has specified "CP SET MSG IUCV", all messages received from the CP MESSAGE command are sent to the virtual machine through IUCV. IUCV signals the receiving virtual machine with an external interrupt. The message may be retrieved by using the IUCV RECEIVE function and may be used by a program running in the virtual machine.

For a complete list of the CP SET commands that can use the IUCV parameter, see the *VM/SP CP Command Reference* or the *VM/SP HPO CP Command Reference.*

The Message System Service uses the IUCV maximum value of 255 for the number of outstanding messages. If this message limit is exceeded, any additional incoming messages are routed directly to the virtual machine console or alternate console, and the virtual machine is not notified about these messages. This situation is most likely to occur when there is a high volume of incoming messages and the virtual machine is running with external interrupts disabled.

The Message System Service identifies the source of the message it intercepts by a code in the IUCV message class field. The message source is interpreted as follows:

Class	*Message Source*
1	Message sent using CP MESSAGE (MSG) or CP MSGNOH
2	Message sent using CP WARNING (WNG)

3 Asynchronous CP messages, CP responses to a CP command executed by a virtual machine using *MSG, and any other console I/O initiated by CP.

4 Message sent using CP SMSG command

5 Any data directed to the virtual console by the virtual machine (WRTERM, LINEDIT, etc.)

6 Error message from CP (EMSG)

7 Information messages for CP (IMSG)

8 Single Console Image Facility (SCIF) message from CP.

Error and information messages (classes 6 and 7) are types of CP messages and are included in class 3 when EMSG and IMSG are not specifically set to IUCV through the CP SET commands.

The format of the data received from IUCV is as follows:

```
col 1   col 9
|       |
V       V
userid  text
```

The userid portion of the data identifies the sender. If the data is not received by a MSG, WNG, SMSG, or using SCIF, then the userid is that of the recipient.

If a virtual machine has both a valid path to the Message System Service and a secondary user specified in the CONSOLE directory control statement (enabling that virtual machine to use SCIF), then incoming messages (except for SMSGs, which are not console messages) are directed to the secondary user instead of the IUCV Message System Service. If the secondary user is not available, the message is queued on the Message System Service path.

For information on SCIF, see Chapter 12, "Single Console Image Facility" on page 279.

Note: The following types of data are not placed in the console spool file for the indicated conditions:

- CP command output -- if this is being received in a buffer through DIAGNOSE code X'08'.
- Messages and Warnings -- if they are being trapped by IUCV and the Message System Service.

Chapter 6. The Message All System Service

The Message All System Service is an IUCV system service that interacts with the existing Message System Service. Messages not requested by the Message System Service will be sent on the *MSGALL path; they are never sent by IUCV twice.

Connections to the Message System Service have priority; all messages requested by that connection with SET commands are sent on that path. Messages for the Message All System Service are not affected by the SET command settings.

The userid in the IUCV CONNECT function statement must be specified as *MSGALL to use the Message All System Service. If no connection is established to the Message System Service, all currently trapped messages are sent to the virtual machine, through IUCV, over the path established by connecting to the Message All System Service.

Note: The following conditions apply:

Console output sent over the *MSGALL path if unsuccessful with *MSG

- CPCONIO and EMSGs generated as part of a DIAGNOSE code X'08' operation.
- MSGs, WNGs, IMSGs, VMGENIO, and SCIFed messages.

Console output sent directly to the terminal

- Asynchronous CPCONIO (including PER/TRACE events) and EMSGs *not* generated as part of a DIAGNOSE code X'08' operation.

Console output never sent over *MSGALL path

- SMSGs
- Output generated by the CP ECHO and SET LOGMSG commands.

The Message All System Service uses the IUCV maximum value of 255 for the number of outstanding messages. If this message limit is exceeded, any additional incoming messages are routed directly to the virtual machine console or alternate console, and the virtual machine is not notified about these messages. This situation is most likely to occur when there is a high volume of incoming messages and the virtual machine is running with external interrupts disabled.

Chapter 7. The DASD Block I/O System Service

The DASD Block I/O System Service is a CP system service. It provides a virtual machine with device-independent access to its virtual DASD devices. Device types supported are the Count Key Data (CKD) devices: 2314, 2319, 3330, 3333, 3340, 3344, 3350, 3375, and 3380, and the Fixed Block Architecture (FBA) devices: 3310 and 3370. (Device 2319 is formatted as a 2314, device 3333 is formatted as a 3330, and device 3344 is formatted as a 3340.) This service supports logical block sizes of 512, 1024, 2048, and 4096 bytes.

Notes:

1. *The CMS 4K block structure on the first track of a 3340 disk is formatted differently than the other tracks of a 3340 CMS disk. The first track of the mini-disk contains three blocks. The first block has a length of 80 bytes, the second, 4096 bytes, and the third, 80 bytes. The remainder of the mini-disk is formatted as usual, two 4096-byte blocks on each track.*

2. *Multiple I/O requests can be outstanding, and you may continue with asynchronous processing or choose to wait for the completion of an I/O request. Although multiple I/O requests may be requested in a first-in/first-out (FIFO) manner, be aware that queuing within CP may result in the I/O completion in a different sequence.*

Establishing Communications with DASD Block I/O Service

The CMS RESERVE command and the CMS DISKID function should be issued before using the DASD Block I/O System Service. These two facilities enable you to create a uniquely organized CMS file on a DASD and obtain information about the file needed to use the DASD Block I/O System Service. For further information, see "Using the DASD Block I/O System Service from CMS" on page 246, or see the *VM/SP CMS Command Reference* and *VM/SP CMS Macros and Functions Reference.*

DASD Block I/O uses IUCV to set up communication between itself and a virtual machine. The IUCV macro checks the validity of all the IUCV parameters. Any IUCV errors are handled according to IUCV specifications. The DASD Block I/O System Service checks the validity of all the parameters it requires. Any errors resulting from this check are handled as described in the following sections.

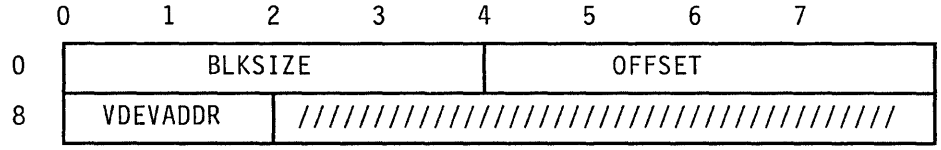
IUCV requires that the virtual machine use the DECLARE BUFFER function to initialize the virtual machine for IUCV communication. This function also specifies a buffer where IUCV can store external interrupt

information. After communication is established with IUCV, the virtual machine must issue an IUCV CONNECT to establish a path between itself and the target communicator. Here, the target communicator, is the DASD Block I/O System Service. Only one CONNECT may be issued to userid *BLOCKIO for each virtual device that is intended to receive I/O requests.

No special authorization is required for a virtual machine to use DASD Block I/O. The MAXCONN (maximum connection) limit in the directory can be enlarged to satisfy the user's requirements. The DASD Block I/O System Service allows connections from any user.

IUCV CONNECT to the DASD Block I/O System Service

An IUCV CONNECT is issued by the virtual machine with USERID=*BLOCKIO and PRMDATA=YES specified in the IUCV CONNECT parameter list. Here, IPUSER, the user data field in the IUCV parameter list, must have the following format. These values are obtained by the CMS DISKID function.



where:

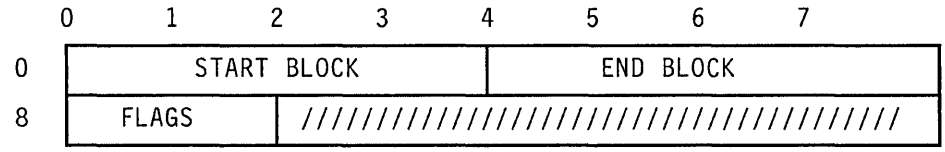
BLKSIZE
is the block size of the specified disk. The block size may be 512, 1K, 2K, or 4K bytes.

OFFSET
associates a physical block number to the first user data block on the disk. Note that this number represents the number of sequential blocks used on the disk by the CMS file system to implement its structure. The DASD Block I/O System Service does not check the validity of this number. Therefore, the application may change this number if desired, but you could overlay files used by CMS.

VDEVADDR
is the virtual device address of the disk where the Block I/O is to be performed.

All reserved fields must be set to zero.

If all the parameters required by DASD Block I/O are valid, DASD Block I/O issues an IUCV ACCEPT on the path specifying PRMDATA=YES. The following information is returned in the IPUSER field of the IUCV Connection-Complete external interrupt buffer:



where:

START BLOCK
: 1 minus the OFFSET specified on the IUCV CONNECT. This value along with END BLOCK is the range of block numbers allowable on the DASD Block I/O request.

END BLOCK
: The number of blocks on the specified virtual device minus the OFFSET specified on the IUCV CONNECT. This value along with START BLOCK is the range of block numbers allowable on the DASD Block I/O request.

FLAGS
: A set of bits defining the status of the virtual device. One bit is defined and the others are reserved.

```
RDONLY   X'0001'   Virtual device is read only
Unused   X'FFFE'   Zero
```

All reserved fields are returned as zero.

If any of the parameters passed to DASD Block I/O are invalid, DASD Block I/O issues an IUCV SEVER on the path and flags the error. The first byte of the IPUSER field contains one of the following error codes:

```
X'01'     Virtual device is not defined
X'02'     Virtual device is not supported
X'03'     Block size is not supported
X'04'     IUCV path already exists for this device
X'05'     Connection is not using PRMDATA=YES
X'06'     Reserved field is not set to zero
```

IUCV SEND to the DASD Block I/O System Service

When the connection to the device is ACCEPTed by DASD Block I/O, you can start sending I/O requests to DASD Block I/O. You can specify TRGCLS=, DATA=PRMMSG, and the PRMMSG= options on the IUCV SEND or you can move the necessary data into the IUCV parameter list yourself. The TRGCLS= option sets the type of I/O requested, read or write. The DATA=PRMMSG option sets a flag in IPFLAGS1, and the PRMMSG= option moves the block number and virtual buffer address into the IUCV parameter list. The following list defines the input necessary for the DASD Block I/O System Service on an IUCV SEND:

```
IPRMMSG1      Block number
IPRMMSG2      Virtual buffer address
IPTRGCLS      Block I/O service requested
       F'01'         Write request (CMS formatted)
       F'02'         Read request (CMS formatted)
```

DASD Block I/O tries to perform the request. It issues an IUCV REPLY to return the results of the I/O requests. One of the following return codes is returned in the IPRMMSG1 field of the IUCV parameter list:

```
F'00'      I/O completed successful
F'01'      Invalid block number
F'02'      Invalid data buffer address
F'03'      Write on read/only DASD
F'04'      Incorrect block size - format error
F'05'      Unrecoverable I/O error
F'06'      Invalid service requested
F'07'      Protection exception on virtual buffer
```

If you have misused IUCV protocol set up for this system service, DASD Block I/O issues an IUCV SEVER on the path and flags the error. The first byte of the IPUSER field contains one of the following error codes:

```
X'07'      IUCV communication was not sent using DATA=PRMMSG
X'08'      No one-way messages are allowed on the path
```

If the device is reset, the path is QUIESCEd and no more requests are allowed. When there are no I/O requests outstanding, DASD Block I/O issues an IUCV SEVER on the path and flags the error. The first byte of the IPUSER field contains the following return code:

```
X'09'      Virtual device has been reset
```

When all communications with the DASD Block I/O System Service are completed, you can terminate communications by issuing either an IUCV SEVER or/and IUCV RETRIEVE BUFFER.

Using the DASD Block I/O System Service from CMS

The DASD Block I/O System Service provides a virtual machine with device-independent access to its virtual DASD devices. Programs using the DASD Block I/O System Service bypass the CMS file system, and they read or write directly from CP.

Before using the DASD Block I/O System Service, you should issue the CMS RESERVE command and the CMS DISKID function.

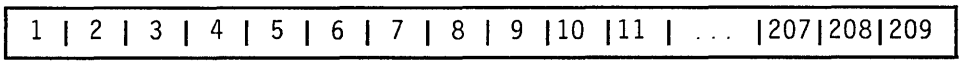
The CMS RESERVE command allocates all available blocks of a 512-, 1K-, 2K-, or 4K-byte block formatted minidisk to a unique CMS file. The file created has the following format:

- Filename, filetype, and filemode letter the user specified

- Filemode number 6, if the filemode number was not specified in the command
- Logical record length equal to the CMS disk block size
- Fixed (F) record format
- The number of records is the total number of blocks available on the disk minus the number of blocks used by CMS. The number of blocks used by CMS is referred to as the offset. This CMS overhead varies with the size of the minidisk. The data blocks physically follow the blocks used by CMS.

The file created can be read or written via the DASD Block I/O System Service or the CMS file system. Because a CMS file structure has been created on the disk, the file may be accessed using the CMS file system. Let's consider the following example:

If you have a 3330 device with one cylinder formatted with 1024-byte block size, 209 blocks are available. After you issue the RESERVE command, the file created has the following format:



where:

Physical block number(s)	**Description**
1 and 2	Contain the IPL records
3	Contains the volume label
4 or 5	Contain the CMS directory file
6	Contains the allocation map
7	Contains the alternate allocation map
8	CMS level 1 pointer block
9 through 209	Data blocks

Physical blocks 1 through 8 are the blocks used by the CMS file system. Physical blocks 9 through 209 are the data blocks. Next, issue the following format of the FILEDEF command:

```
FILEDEF ddname DISK vaddr
```

This command associates the name of the virtual minidisk referred to in your program, "ddname", to the virtual address of the minidisk, "vaddr". The "ddname" is input to the DISKID function.

The DISKID function obtains the necessary information on the physical organization of the RESERVEd minidisk that will be used by the DASD Block I/O Service. The DISKID function obtains the virtual address, the block size, and the offset of this minidisk. (In the above example, the block size is 1024 bytes and the offset is 8.)

Before using the DASD Block I/O System Service, you must initialize your virtual machine for IUCV communications. IUCV enables a program running in your virtual machine to communicate with DASD Block I/O. Use the IUCV DECLARE BUFFER function or the CMS HNDIUCV SET macro to initialize your virtual machine for IUCV communications. You should use the CMS IUCV support since this support allows other programs running in the virtual machine to use IUCV.

To establish a path between your virtual machine and the DASD Block I/O System Service, your program must issue either the IUCV CONNECT function or the CMSIUCV CONNECT macro. The USERID parameter on the IUCV CONNECT macro must be "*BLOCKIO" and the PRMDATA parameter must be "YES". PRMDATA=YES indicates that the your program will receive messages in its parameter list. Information about the minidisk returned by the DISKID function -- the virtual address, blocksize, and offset -- must be moved into the IPUSER field of the IUCV CONNECT parameter list.

If all the parameters required by DASD Block I/O are valid, DASD Block I/O issues the IUCV ACCEPT function with the PRMDATA=YES parameter specified. Here, PRMDATA=YES indicates that the DASD Block I/O System Service will receive messages in its parameter list. If invalid parameters are passed from the CONNECT to DASD Block I/O, DASD Block I/O issues an IUCV SEVER on the path.

If DASD Block I/O issued an IUCV ACCEPT, your virtual machine receives an IUCV Connection Complete external interrupt. The IPUSER field of the Connection Complete external interrupt buffer contains the starting and ending block numbers allowable on the DASD Block I/O requests, and contains flags describing the status of the virtual device. The starting block number (START BLOCK) is 1 minus the offset. The ending block number (END BLOCK) is the total number of available blocks minus the offset.

Since DASD Block I/O bypasses the CMS file system, START BLOCK can contain a negative value (1 - OFFSET) and these blocks can be used by DASD Block I/O. In the example above, START BLOCK would be -7 (1 - 8) and END BLOCK would be 201 (209 - 8). The range -7 to 201 equals 209 - the number of available blocks on the device. Data block 1 is actually physical block 9, data block 2 is actually physical block 10, and the last block (data block 201) is actually physical block 209. Programs using DASD Block I/O should only write data blocks; therefore, a block number less than 1 should never be written. This would destroy a block that was used to implement the CMS file structure.

You can now start sending I/O requests to DASD Block I/O by issuing the IUCV SEND function. You must specify the block number, virtual buffer address, and type of request desired in the IUCV SEND parameter list. Blocks are read or written randomly as requested.

If no error occurred in the IUCV SEND, DASD Block I/O issues an IUCV REPLY to return the results of the I/O requests. If an error occurred in the IUCV SEND, DASD Block I/O issues an IUCV SEVER.

When you want to terminate communications with the DASD Block I/O System Service, issue the IUCV SEVER function, CMS CMSIUCV SEVER macro, IUCV RETRIEVE BUFFER function, or CMS HNDIUCV CLR macro.

Chapter 8. The Signal System Service

The Signal System Service is a CP system service. It allows virtual machines in a virtual machine group to signal each other. The Signal System Service can only be used by virtual machines in a virtual machine group. Each virtual machine in a group is identified by a unique 16-bit signal ID. When a virtual machine connects to the Signal System Service, it may request that a particular signal ID be assigned to it. If you have not set up the virtual machine to request a specific signal ID, the Signal System Service automatically assigns one to your virtual machine.

All members of a virtual machine group can send 8 bytes of signal data (user information) to any member in the group using IUCV SEND, specifying the signal ID of the virtual machine they want to receive the signal data. A virtual machine can also signal all members in a group using a Broadcast signal. Group members can request notification of members entering and leaving the group by specifying Signal-In and Signal-Out flags when they connect to the Signal System Service.

Using the Signal System Service requires no directory authorization. The Signal System Service allows only one connection per virtual machine.

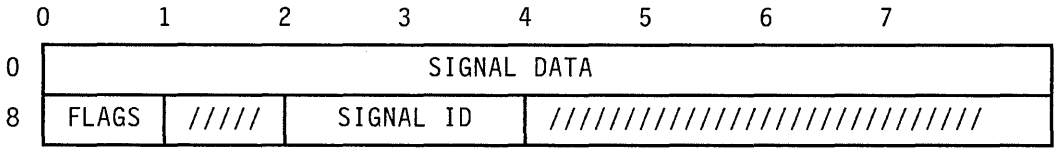
Establishing Communications with the Signal System Service

The Signal System Service uses IUCV to communicate between itself and a virtual machine. The IUCV macro checks the validity of all the IUCV parameters and any errors are handled according to IUCV specifications. The Signal System Service checks the validity of all the parameters it requires. Any errors resulting from this check are handled as described in the following sections.

Your first step in establishing IUCV communications with the Signal System Service is to issue an IUCV DECLARE BUFFER. This initializes the virtual machine for IUCV communication. This function also specifies a buffer where IUCV can store external interrupt information.

IUCV CONNECT to the Signal System Service

After you establish communications with IUCV, you must issue an IUCV CONNECT with USERID = *SIGNAL and PRMDATA = YES in the IUCV CONNECT parameter list. The user data field must have the following format:



where:

SIGNAL DATA
is the eight bytes of user information or signal you want passed to other members of the group. Only group members that have specified the Signal-In flag when they connected will receive the data.

FLAGS
is a set of bits defining the signal options chosen by you for your virtual machine. The first three bits are defined and the others are reserved. The defined bits are:

X'80'	Signal-In
X'40'	Signal-Out
X'20'	Signal ID has been specified
X'1F'	Reserved

SIGNAL ID
is the signal ID you want assigned to your virtual machine. This signal ID is used by other group members to communicate with your virtual machine. This field is only used if you set the signal ID flag bit (X'20') in the FLAGS field.

You must set all reserved fields and flags to zero.

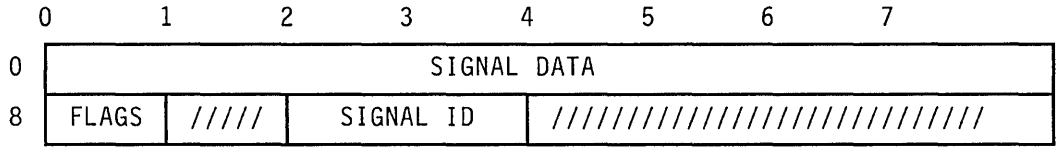
If you specify the Signal-In flag, your virtual machine is signaled when future group members enter your group by connecting to the Signal System Service.

If you specify the Signal-Out flag, your virtual machine is signaled when group members in your group break their connection (via IUCV SEVER) with the Signal System Service.

The IUCV CONNECT function returns a PATHID to your virtual machine. You must specify this PATHID in the IUCV SEND parameter list for all subsequent communication to the Signal System Service.

When you issue IUCV CONNECT to the Signal System Service, the connection is either accepted (the Signal System Service issues an IUCV ACCEPT) or severed (the Signal System Service issues an IUCV SEVER).

If the connection is accepted, the user data field on the IUCV connection complete external interrupt will have the following format:



where:

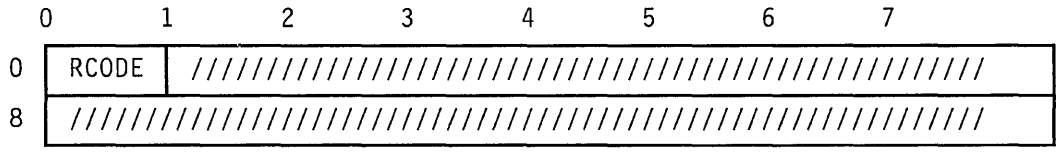
SIGNAL DATA
is unchanged from the IUCV CONNECT.

FLAGS
are unchanged from the IUCV CONNECT.

SIGNAL ID
is the signal ID you assigned to your virtual machine. If you did not assign a signal ID, the Signal System Service assigns a unique signal ID for you and stores it in this field.

All unused fields remain unchanged from the IUCV CONNECT.

If the connection is rejected, the user data field on the IUCV SEVER external interrupt will have the following format:



where:

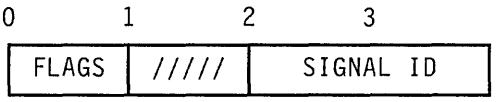
RCODE
is the return code indicating the reason the connection was severed.

Return codes resulting from connection errors:

X'01' You are not a member of a Virtual Machine Group.
X'02' You are already connected to the Signal System Service.
X'03' You did not specify PRMDATA=YES in the IUCV CONNECT parameter list.
X'04' The reserved fields were not set to zero.
X'05' The signal ID you specified was not unique.

Sending Signals

Upon notification of a successful connection, your virtual machine is ready to send signals. You may now issue IUCV SEND requests to the Signal System Service specifying the eight byte signal (parameter list data), the target's signal ID, and the flag settings. Specify the target's signal ID and the flag settings in the target class (TRGCLS) with the following format:



where:

FLAGS
is a set of bits defining the the handling of the signal. Only two bits are defined and the others are reserved. The defined bits are:

X'10' Broadcast Signal
X'08' Invalid Signal ID
X'E7' Reserved

SIGNAL ID
is the target's signal ID.

If you specify the Broadcast Signal (X'10') flag, a signal is sent to all of the other users in your group that are connected to the Signal System Service.

If you send a signal using an invalid Signal ID, the Signal System Service returns the signal to you with an error indicator (X'08') in the FLAGS field. The target class and the signal data remain unchanged.

If the parameter list data option is not used, or if the signal is not one-way, then the connection is severed.

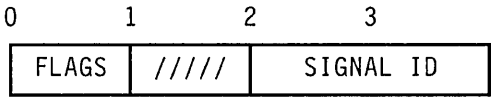
Return codes resulting from send errors:

X'06' The signal was sent without the DATA=PRMMSG option specified.
X'07' The signal sent was not a one-way signal.

Receiving Signals

As a member of a Virtual Machine Group, your virtual machine can receive three types of signals. These are Signal-In, Signal-Out, and a normal signal sent by another group member via an IUCV SEND. The Signal System Service passes these signals to your virtual machine via an IUCV SEND using a one-way message with the signal specified in the parameter list data.

Specify the source's signal ID and the flag settings in the target class (TRGCLS) with the following format:



where:

FLAGS
is a set of bits defining the type of signal sent. Only three bits are defined and the others are unused. The defined bits are:

X'80'	Signal-In
X'40'	Signal-Out
X'10'	Broadcast signal
X'2F'	Unused

SIGNAL ID
is the source's signal ID.

The Signal System Service sets all unused fields to zero.

If the Signal-In flag is on, this signal was specified in the user data of the user's IUCV CONNECT to the Signal System Service.

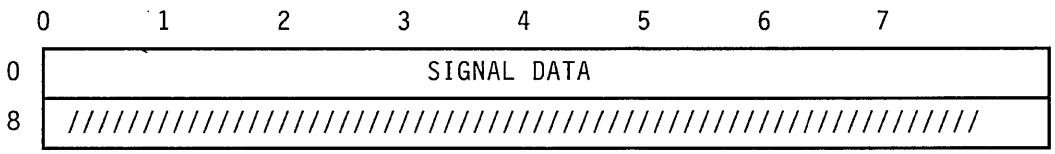
If the Signal-Out flag is on, this signal was specified in the user data of the user's IUCV SEVER to the Signal System Service.

If the Broadcast flag or no flags are on, this signal was specified in the parameter list data of the user's IUCV SEND to the Signal System Service.

If you want to stop receiving signals from the Signal System Service, you can use IUCV QUIESCE. However, the Signal System Service does not queue signals, and the signals from other members of the group will be lost until you issue an IUCV RESUME.

Leaving the Signal System Service

When all communications with the Signal System Service are completed, you can terminate communication by issuing either an IUCV SEVER or an IUCV RETRIEVE BUFFER. The user data field on a SEVER must have the following format:



where:

SIGNAL DATA
is the eight byte signal you want passed to other members of the group. Only group members that have specified the Signal-Out flag when they connected to the Signal System Service will receive the data.

If a SEVER is generated by the CP system, as on a RETRIEVE BUFFER or a virtual machine reset, the SIGNAL DATA is set to all zeroes.

Chapter 9. The Error Logging System Service

The Error Logging System Service is a CP system service. It lets a virtual machine receive a copy of all records currently written to the VM CP Error Recording Area. The virtual machine can record this information, act on it, or report it to other programming support.

Establishing Communications with the Error Logging System Service

"*LOGREC" is the Error Logging System Service userid. The virtual machine communicates with the Error Logging System Service via IUCV in the following way:

- Issues an IUCV DECLARE BUFFER.
- Enables for IUCV interrupts in CR 0 and PSW.
- Has an IUCV *LOGREC authorization card in the directory.
- Issues an IUCV CONNECT to *LOGREC.
- Processes IUCV external interrupts.
- Issues an IUCV RECEIVE for each record.

Interactions

The VM CP Error Recording Area records errors even if a virtual machine does not connect to *LOGREC.

IUCV messages are of the same length and format as records that appear in the CP Error Recording Area. When IUCV sends a message, the IPBFLN1F field in the IUCV externals interrupt information indicates the messages length.

The CP directory must authorize the virtual machine to connect to the Error Logging System Service. Although the directory may authorize many virtual machines to connect to the Error Logging System Service, only one virtual machine may connect at any one time.

IUCV does not send messages under the following conditions:

- When a virtual machine does not connect to the Error Logging System Service.
- When a virtual machine issues an IUCV QUIESCE on the *LOGREC path to prevent *LOGREC from sending messages.
- When a virtual machine is not processing messages and the number of outstanding messages reaches the limit of 255.

Chapter 10. The SPOOL System Service

The SPOOL System Service is a CP system service. No special authorization is required for a virtual machine to use the SPOOL System Service. The directory entry for the virtual machine to be used as logical printers must specify the IUCV *SPL option. It allows users an interface for communication between CP and a printer subsystem. To a VM operator, printers driven by this subsystem appear very similar to existing system printers (1403, 3211, 3800, etc.). These subsystem printers are called logical printers to differentiate them from system printers supported directly by CP.
The SPOOL System Service allows a virtual machine to:

- Select a spool file from the print chain for processing
- Close a SELECTed file
- Send messages or command responses to the operator or other users
- Read the spool records (SFBLOKs and SPLINKs) for a selected file
- Read an external attribute buffer (XAB)[12] for a selected file
- Receive CP commands that affect the logical printer
- Be notified when a print file is available for processing
- Purge a print file.

Establishing Communications with the SPOOL System Service

The SPOOL System Service uses IUCV to communicate between itself and a logical printer. The IUCV macro checks the validity of all IUCV parameters. All IUCV errors are handled according to IUCV specifications. *SPL validity checks all the parameters the system service requires and handles the errors as described below. *SPL is the SPOOL System Service userid. IUCV requires the virtual machine to do a DECLARE BUFFER to initialize the virtual machine for IUCV communication. The virtual machine which runs the logical printer does an IUCV CONNECT to the SPOOL System Service. The connection lets the logical printer use the SPOOL System Service to obtain spool print files for processing. It also makes possible the support of operator and user commands for the logical printer.

---

12 The External Attribute Buffer (XAB) is a control block that contains data the user creates to specify additional information about a print file. Each print file has its own XAB and CP has the facilities to maintain the XABs. See page 96 for more details about an XAB.

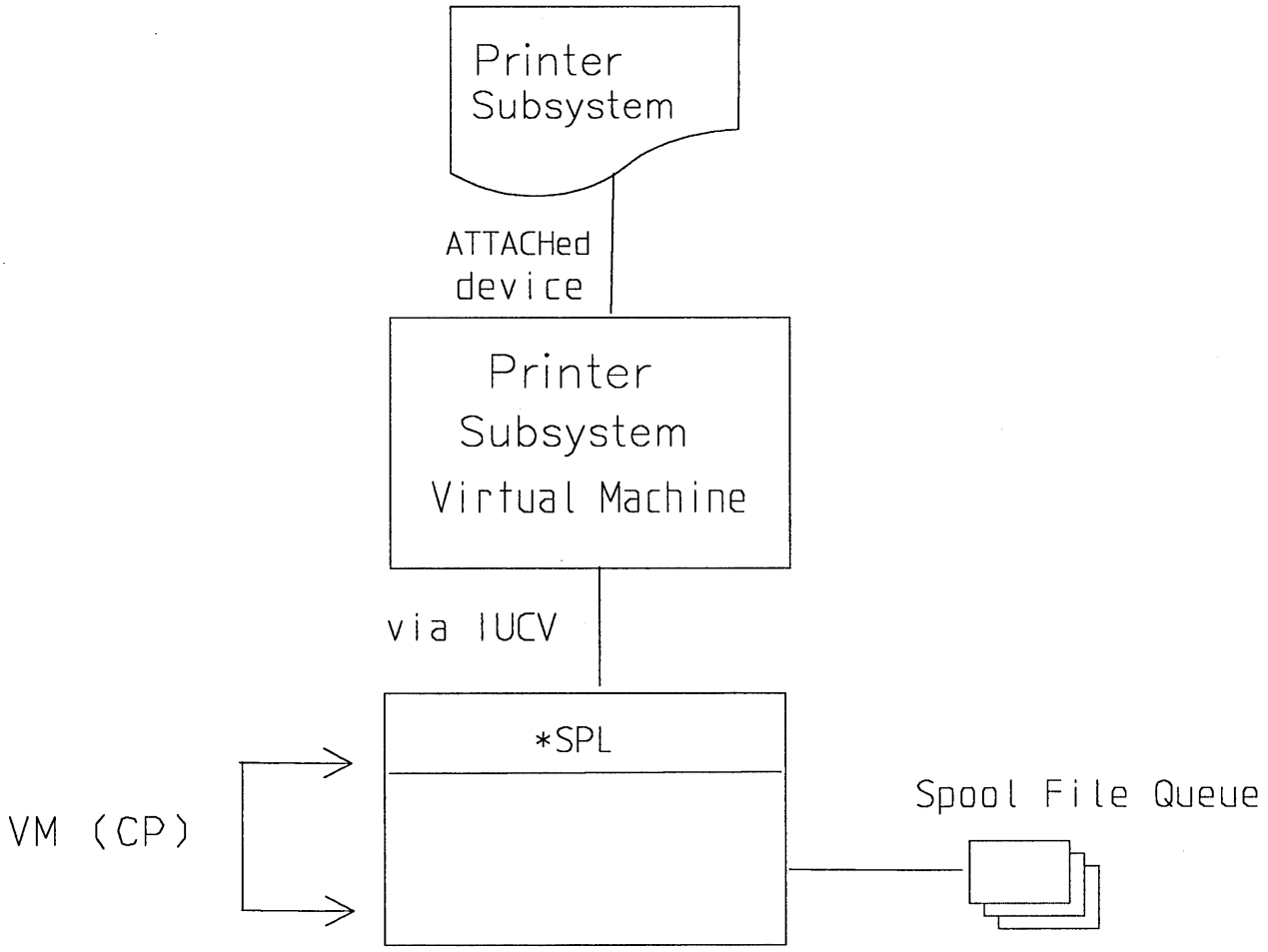


Figure 12. Printer Subsystem Support

IUCV CONNECT to the SPOOL System Service

An IUCV CONNECT is issued by the virtual machine with USERID = *SPL, PRIORITY = NO, PRMDATA = YES, and USERDTA = *logical printer name.*

A logical printer name is a one-to-eight alphameric name your installation assigns. If you specify a logical printer name less than four characters, then the name *cannot* contain all hexadecimal characters. A logical printer name cannot be ALL. A logical printer name **must** not contain imbedded blanks; however, trailing blanks are permitted. It is the installation's responsibility to control the names assigned to logical printers. The name should not conflict with any other names, options, or notation the installation uses.

Example: It is recommended that a logical printer name should:

- Not be the same as a userid on the system
- Not be "PROC" or "PROCESSOR"
- Not use a dash ("-").

FFF is **not** a valid logical printer name, whereas BLDG1 is.

The SPOOL System Service allows multiple logical printers to be connected with the same name. This way, when you use the SPOOL System Service SEND function, each logical printer of the specified name is sent a message. The CP SEND command sends data or commands to the logical printer if the target of the CP SEND is a logical printer.

If you want to create your own IUCV parameter list, then the IPUSER user data field must have the following data.

```
         ORG   IPUSER
SPLNAME  DS    CL8        LOGICAL PRINTER NAME
```

If all the parameters required by SPOOL System Service are valid, SPOOL System Service issues an IUCV ACCEPT to complete the connection with the authorized virtual machine. It also identifies the logical printer to CP. All other SPOOL System Service functions depend on the IUCV CONNECT and ACCEPT being complete.

If any of the parameters passed to SPOOL System Service are invalid, SPOOL System Service issues an IUCV SEVER. However, SPOOL System Service also issues an IUCV SEVER when the logical printer is ready to end its connection with SPOOL System Service. When the IUCV SEVER is issued, SPOOL System Service releases all storage associated with the SEVERed printer. Any spool files being processed at the time of the SEVER are recovered and returned to the print chain.

IUCV SEND to the SPOOL System Service

When the SPOOL System Service ACCEPTs the connection to a logical printer, you can send the following types of requests to the SPOOL System Service with an IUCV SEND:

SELECT
CLOSE
MESSAGE
READ

The pages that follow describe these functions.

The SELECT Function

Through a file-select, a logical printer can obtain a spool print file for processing. To select a file to process, the logical printer sends a SELECT (via IUCV SEND, TYPE=2WAY communication, IPTRGCLS=F'4'[13], DATA=PRMMSG or DATA=BUFFER) to the SPOOL System Service.

The SELECT function does not require that a previously SELECTed file be CLOSEd before another file is SELECTed. Thus several files can have SELECTed status at the same time.

When DATA=PRMMSG (all the user data is in the parameter list), then PRMMSG= contains the following information:

```
IPRMMSG1   bytes 0 and 1      spoolid of file (if applicable;
                                see the SPLSESID flag)
           bytes 2 and 3      reserved
IPRMMSG2   byte 0             a flag
           bytes 1 to 3       reserved

         BITS DEFINED in BYTE 0 of IPRMMSG2
SPLSESID   EQU  X'80' select spool file by spoolid
SPLSECON   EQU  X'40' select spool file for "convert"
                      process
```

Notes:

1. *If the SPLSESID bit is off, then select the next available file matching the default selection criteria.*

2. *If the SPLSECON bit is off, then select a spool file to start the print process.*

   A file that is selected for print processing is similar to files which are being handled by CP driven system printers. The file is removed from the print chain while it is being processed.

   A file that is selected for convert processing is special because it remains on the print chain while it is being processed. The operator or owner of a file for which conversion is active can issue the CHANGE, PURGE, and TRANSFER commands for the file. (See the PURGE function on page 275).

When DATA=BUFFER (the user data is in the buffer and IUCV RECEIVE transfers the data), the BUFLEN=50 and the buffer contains the following information:

---

[13] The value for IPSRCCLS should match the value for IPTRGCLS to correlate the SEND with the REPLY.

```
SPLSEFLG   DS   1X       flag
SPLSETYP   DS   1X       type of file eligible for selection

SPLSECLS   DS   4CL1     CLASSes
SPLSEFSH   DS   CL4      FLASH
SPLSEFRM   DS   CL8      FORM
SPLSEDES   DS   4CL8     DESTinations

         BITS DEFINED in SPLSEFLG
SPLSECHG   EQU  X'80'  change default selection criteria
                      to the new selection criteria sent in
                      the BUFFER.
                      Note:  All selection criteria is
                      changed so all data must be specified
                      in the buffer.
SPLSECON   EQU  X'40'  select spool file for "convert"
                      process.
                      Note:  If this bit is off, files
                      selected will be for the print process.

         BITS DEFINED in SPLSETYP
SPLSE5AC   EQU  X'80'  select files using X'5A' CCWs
SPLSEN5A   EQU  X'40'  select files not using X'5A' CCWs
SPLSECNV   EQU  X'20'  select files that are converted
SPLSENCV   EQU  X'10'  select files that are not converted
SPLSN38L   EQU  X'08'  select files not using 3800 load CCWs
SPLSBE38   EQU  X'04'  select files containing 3800 load CCWs
                      at the beginning of the file
SPLSAN38   EQU  X'02'  select files containing 3800 load CCWs
                      anywhere within the file
```

If the three flags: SPLSN38L, SPLSBE38, and SPLSAN38 are all either B'0' or B'1', then using of 3800 load CCWs within a spool file does not affect whether the file is SELECTed. (The tests are bypassed.) If the setting of these three flags is MIXED, then a particular spool file is SELECTed only if its used 3800 load CCWs matches that of one of the flags that is B'1'.

Notes:

1. *If the SPLSECON bit is off, then select a spool file to start the print process.*

   A file that is selected for print processing is similar to files which are being handled by CP driven system printers. The file is removed from the print chain while it is being processed.

   A file that is selected for convert processing is special because it remains on the print chain while it is being processed. The operator or owner of a file for which conversion is active can issue the CHANGE, PURGE, and TRANSFER commands for the file. (See the PURGE function on page 275).

2. *You must specify at least one each of CLASS, FLASH, FORM, DESTination, and bit in SPLSETYP*

3. *If you specify a value of blank for CLASSes, FLASH, FORM, or DESTinations, then no file is SELECTed.*

4. *Any unused length of the CLASSes, FLASH, FORM, or DESTinations fields must be blanks.*

5. *A value of asterisk (c'*') for a CLASS, means that any CLASS file can be selected.*

6. *A value of c'*____' (each '_' is a blank and there are seven of them), for FORM (or DESTination) means that any FORM or DESTination file can be selected.*

7. *A value of c'*__' (there are three blanks), for FLASH means that any FLASH file can be selected.*

8. *A value of c'OFF___' (there are five blanks), for FORM means that a file is eligible for selection if it has either been assigned a FORM of 'OFF' or if it has been assigned the default printer FORM. (See the DEFPRT options of the SYSFORM macro for DMKSYS ASSEMBLE in the VM/SP Planning Guide and Reference) or the VM/SP HPO Planning Guide and Reference).*

9. *A value of c'OFF__' (there are five blanks), for DESTination means that files which do not have a DESTination assigned to them, or have been assigned OFF, can be selected.*

10. *A value of c'OFF_' (there is one blank), for FLASH means that files which do not have a FLASH assigned to them, or have been assigned OFF, can be selected.*

The SPOOL System Service responds to the SELECT request, using an IUCV REPLY with DATA=PRMMSG, and PRMMSG= containing the following information:

```
IPRMMSG1  bytes 0 and 1 spoolid
          bytes 2 and 3 COPY count for the file
IPRMMSG2  byte  0       flag - primary response information
          byte  1       flag - secondary response information
          bytes 2 and 3 reserved

        BITS DEFINED in BYTE 0 of IPRMMSG2
   EQU  X'04'1  file not SELECTed

        BITS DEFINED in BYTE 1 of IPRMMSG2
   EQU  X'80'2  SFBLOK marked "to be purged"
   EQU  X'40'2  file has been CONVerted
   EQU  X'20'2  file contains X'5A' CCW's
   EQU  X'10'3  specified file not found on PRINT queue
   EQU  X'08'3  specified file in USER HOLD status
   EQU  X'04'3  specified file in SYSTEM HOLD status
   EQU  X'02'   file has been "selected for printing"
   EQU  X'01'   reserved
```

[1] If this bit is 0, then a file has been selected.

[2] The selected file has this characteristic.

[3] The specified spoolid was not selected.

Reasons The SELECT Function May Sever

The following situations are considered user errors and cause the SPOOL System Service to sever the IUCV path to a logical printer.

- Use of TYPE=1WAY communication, return code=X'80'
- Incorrect buffer length, return code=X'80'.
  The specified buffer length must be 50 bytes.
- Conflicting selection criteria options, return code=X'80'
  - DATA=PRMMSG, SPLSESID flag is off, and default selection criteria has not been defined
  - DATA=BUFFER and both the SPLSE5AC and SPLSEN5A flags are off
  - DATA=BUFFER and both the SPLSECNV and SPLSENCV flags are off.

When the SPOOL System Service SELECT does a SEVER, the format of the IPUSER field is:

```
IPUSER + 0:   error code
IPUSER + 1:   X'04'
IPUSER + 2:   IPMSGID from the IUCV parameter list
                   that caused error
```

The CLOSE Function

A logical printer uses the SPOOL System Service CLOSE function when processing of a SELECTed file is complete. The spool file copy count can also be changed with the CLOSE function. When the CLOSE is complete, the logical printer can specify to delete or requeue the print file from the VM spool. To CLOSE a file, the logical printer sends a CLOSE request (via an IUCV SEND TYPE=2WAY communication, IPTRGCLS=F'5'[14], DATA=PRMMSG or DATA=BUFFER) to the SPOOL System Service.

When a logical printer does a CLOSE, it sends the following information to the SPOOL System Service.

- Spoolid of the file
- COPY count processing information
- Flag indicating the status of the file (if applicable)
- New CLASS, FORM, and DESTination (if applicable) of the file.

When DATA=PRMMSG (all the user data is in the parameter list), then PRMMSG= contains the following information:

---

[14] The value for IPSRCCLS should match the value for IPTRGCLS to correlate the SEND with the REPLY.

```
IPRMMSG1    bytes 0 and 1  spoolid of file
            bytes 2 and 3  COPY count processing information
IPRMMSG2    byte  0        a flag
            bytes 1 to 3   reserved

          BITS DEFINED in BYTES 2 and 3  of IPRMMSG1
         EQU X'0000'           CLOSE does not change the
                               file's COPY count (other CLOSE
                               functions performed as usual)
         EQU X'0001'-X'00FF'   CLOSE updates the COPY
                               count (file remains in
                               SELECTed status and no other
                               changes are done)

          BITS DEFINED in BYTE 0 of IPRMMSG2
SPLCLPUR    EQU  X'80' close and purge file
SPLCLUHO    EQU  X'40' close and requeue in user hold
SPLCLSHO    EQU  X'20' close and requeue in system hold
SPLCLCON    EQU  X'10' close and requeue converted
            EQU  X'08' reserved
            EQU  X'04' reserved
            EQU  X'02' reserved
SPLCLUNC    EQU  X'01' close and requeue non-converted
```

Notes:

1. *If bytes 2 and 3 of IPRMMSG1 are greater than X'00FF' then the SPOOL System Service severs the IUCV path.*

2. *When you specify SPLCLPUR, it is invalid to specify any other option in byte 0 of IPRMMSG2.*

3. *It is invalid to specify both SPLCLCON and SPLCLUNC.*

When DATA=BUFFER (the user data is in the buffer and IUCV RECEIVE transfers the data), the BUFLEN=20 and BUFFER= contains the following information:

```
SPLCSPID    DS   1H       file spoolid
SPLCFLCL    DS   1X       flag
SPLCCLAS    DS   1X       CLASS
SPLCFORM    DS   CL8      FORM
SPLCDEST    DS   CL8      DESTination

          BITS DEFINED in SPLSFLCL
SPLCLPUR    EQU  X'80' close and purge file
SPLCLUHO    EQU  X'40' close and requeue in user hold
SPLCLSHO    EQU  X'20' close and requeue in system hold
SPLCLCON    EQU  X'10' close and requeue converted
SPLCLCLA    EQU  X'08' close and requeue with new CLASS
SPLCLFRM    EQU  X'04' close and requeue with new FORM
SPLCLDES    EQU  X'02' close and requeue with new DESTination
SPLCLUNC    EQU  X'01' close and requeue non-converted
```

Notes:

1. *When you specify SPLCLPUR, it is invalid to specify any other option in SPLCFLCL.*

2. *It is invalid to specify both SPLCLCON and SPLCLUNC.*

The SPOOL System Service responds to the CLOSE request using an IUCV REPLY with DATA=PRMMSG, and PRMMSG= containing the following information:

```
IPRMMSG1  bytes 0 and 1     spoolid
          bytes 2 and 3     reserved
IPRMMSG2  byte  0           flag
          bytes 1 to 3      reserved

          BIT DEFINED in BYTE 1 of IPRMMSG2
       EQU  X'04'          file not SELECTed or *SPL processing
                           for the file stopped and a *SPL PURGE
                           message already sent to the logical
                           printer which SELECTed the file.
```

Reasons The CLOSE Function May Sever

The following situations are considered user errors and cause the SPOOL System Service to sever the IUCV path to a logical printer.

- Use of TYPE=1WAY communication, return code=X'80'
- Incorrect buffer length, return code=X'80'.
  The specified buffer length must be 20 bytes.
- Conflicting file-disposition options, return code=X'80'.
  For example, specifying `close and purge file` at the same time you specify `close and requeue in user hold`.
- I/O pending, return code=X'10'
- DATA=PRMMSG and bytes 2 and 3 of IPRMMSG1 contain a value greater than X'00FF', return code=X'80'

When the SPOOL System Service CLOSE does a SEVER, the format of the IPUSER field is:

```
IPUSER + 0:   error code
IPUSER + 1:   X'05'
IPUSER + 2:   IPMSGID from the IUCV parameter list
                   that caused error
```

The MESSAGE Function

A logical printer uses the SPOOL System Service MESSAGE function to send messages or command responses to the operator or other users. To send messages, the logical printer sends a MESSAGE (via the IUCV SEND, TYPE=2WAY communication, IPTRGCLS=F'6', DATA=BUFFER, BUFFER=address, and BUFLEN=length-of-buffer) to the SPOOL System Service.

When a logical printer sends a MESSAGE, it sends the following information to the SPOOL System Service in a buffer.

- Userid to receive the message (8 bytes)
- Type of message
  'S' - sent via CP SMSG
  'M' - sent via CP MSG
  'W' - sent via CP MSGNOH and console 'alarm' is sounded
- Message text
  length - *n*+*2* long
  text - *n* characters long

  where *n* is the number of bytes in the message.

The buffer contains the following information:

```
SPLMSUID DS CL8     USERID to receive the message
SPLMSTYP DS CL1     TYPE of message
SPLMSARE DS 0XL224  Message area (maximum size is 224 bytes)
                    - first 2 bytes:  message area length
                    - next 'n' bytes: message text
SPLMSLEN DS XL2     Message area length
                      (actual size of SPLMSARE)
SPLMSTXT DS CL222   Message text (maximum size is 222.)
```

Note:

When a message is sent to a different node using RSCS, the issuer is responsible for placing the necessary networking control-sequence and the actual message.

For example, if a userid of a local RSCS machine (NETMATCH) wants to send a message (HI from OVERHERE) to a VM node (RCVNODE) userid (OVERTHER), then the issuer needs to specify that the message text be sent to the RSCS machine. To do that, the issuer must use the following format for the message text.

```
   MSG        RCVNODE    OVERTHER    HI from OVERHERE

 RSCS Cmd     VM node     userid      message

  networking-control sequence
```

You can send this message text using the CP SMSG command. The virtual machine receiving an SMSG must be authorized to receive and process that SMSG. For more information about SMSG, refer to the

VM/SP CP Command Reference or the *VM/SP HPO CP Command Reference*. For more information about networking control, refer to the *Remote Spooling Communications Subsystem Operation and Use.*

When the SPOOL System Service MESSAGE finishes processing the issuer's request, the SPOOL System Service MESSAGE replies to the issuer with return codes indicating the success or failure of the function.

The reply is sent using an IUCV REPLY, DATA=PRMMSG, and PRMMSG=return-code.

Return codes which do *not* sever the path are:

X'00' Successful message
X'2B' Userid not logged on
X'39' Userid not receiving

Reasons The MESSAGE Function May Sever

The following situations are considered user errors and cause the SPOOL System Service to sever the IUCV path to a logical printer.

- Use of TYPE=1WAY communication, return code=X'80'
- Use of DATA=PRMMSG, return code=X'80'.
  The SPOOL System Service MESSAGE only allows DATA=BUFFER.
- Target userid is all blanks, return code=X'80'.
  The SPOOL System Service MESSAGE does not allow character blanks for the target userid.
- Invalid message type, return code=X'80'.
  The SPOOL System Service only allows message types 'S', 'M', and 'W'.
- Invalid data buffer length, return code=X'80'.
  The SPOOL System Service only allows the data buffer length to be greater than 12 and less than 233.
- Invalid message text format, return code=X'80'.
  The SPOOL System Service MESSAGE has the following restrictions on the format of the message text.
  - It cannot extend beyond the end of the buffer (the message line cannot extend beyond the end either) lines
  - The length of the message line area must be greater than two but less than 224.

When the SPOOL System Service MESSAGE does a SEVER, the format of the IPUSER field is:

```
IPUSER + 0:   error code
IPUSER + 1:   X'06'
IPUSER + 2:   IPMSGID from the IUCV parameter list
                   that caused error
```

The READ Functions

The SPOOL System Service provides three READ functions for use by logical printers:

READ SPLINK
READ SFBLOK
READ XAB (external attribute buffer)

The SPOOL System Service READs are IUCV SEND TYPE=2WAY communications DATA=PRMMSG, and PRMMSG= from a logical printer to the SPOOL System Service.

The READ-SPLINK Function

This function is used to read print lines of a file. With this function, a logical printer can read the CP DASD records (SPLINKs) for a SELECTed spool file. For The SPOOL System Service READ-SPLINK, IPTRGCLS=F'1' and PRMMSG= contains the following information:

```
IPRMMSG1 bytes 0 and 1  spoolid of file
         bytes 2 and 3  number of 4K DASD records to read
IPRMMSG2                buffer address (on a 4K boundary)
```

Note: The size is the number of contiguous 4K buffers for the SPLINKs to be read (each SPLINK has a size of 4K bytes). SPLINKs are read in sequential order. The first READ for a SPLINK after the file has been SELECTed reads the first *n* (number to be read) SPLINK(s) for that file. Following READs read the next *n* SPLINK(s) for the file.

When the READ SPLINK is complete, the SPOOL System Service replies using an IUCV REPLY with DATA=PRMMSG, and PRMMSG= containing the following:

```
IPRMMSG1  bytes 0 and 1     spoolid
          bytes 2 and 3     number of SPLINKs read
IPRMMSG2  byte  0           flag
          bytes 1 to 3      reserved

        BITS DEFINED in BYTE 1 of IPRMMSG2
      EQU  X'20'     CP I/O error
      EQU  X'04'     file not SELECTed or *SPL processing
                     for the file stopped and a *SPL PURGE
                     message already sent to the logical
                     printer which SELECTed the file.
      EQU  X'02'     end-of-file for READ-SPLINK
```

Reasons The READ-SPLINK Function May Sever

The following situations are considered user errors and cause the SPOOL System Service to sever the IUCV path to a logical printer.

- Use of TYPE=1WAY communication, return code=X'80'
- Use of DATA=BUFFER, return code=X'80'.

The SPOOL System Service READ SPLINK only allows DATA=PRMMSG.

- Protection or addressing violation for the user buffer, return code=X'08'.
  Any of the following has happened:
  - The address of the user buffer in not on a 4K boundary
  - There is a storage protection violation involving the user buffer
  - The address range specified for the buffer is not addressable.
- Numbers of buffers specified was zero, return code=X'08'
- I/O pending, return code=X'10'.
  The READ-SPLINK function was done while a previous READ-SPLINK request was outstanding for the file. The IUCV REPLY for a previous READ-SPLINK request must be received before the next READ-SPLINK can be done.

When the SPOOL System Service READ-SPLINK does a SEVER, the format of the IPUSER field is:

```
IPUSER + 0:   error code
IPUSER + 1:   X'01'
IPUSER + 2:   IPMSGID from the IUCV parameter list
                   that caused error
```

The READ-SFBLOK Function

You use this function to read the SFBLOK. With this function, a logical printer can read the information contained in a SELECTed spool file's SFBLOK. For the SPOOL System Service READ-SFBLOK, IPTRGCLS=F'2' and PRMMSG= contains the following information:

```
IPRMMSG1   bytes 0 and 1      spoolid of file
           bytes 2 and 3      buffer size
IPRMMSG2                      buffer address
```

When READ-SFBLOK is complete, the SPOOL System Service replies using an IUCV REPLY with DATA=PRMMSG, PRMMSG= containing the following information:

```
IPRMMSG1  bytes 0 and 1     spoolid
          bytes 2 and 3     length of data
IPRMMSG2  byte  0           flag
          bytes 1 to 3      reserved

        BITS DEFINED in BYTE 0 of IPRMMSG2
      EQU  X'20'    CP paging error
      EQU  X'08'    user buffer not large enough to
                    hold requested data
      EQU  X'04'    file not SELECTed or *SPL processing
                    for the file stopped and a *SPL PURGE
                    message already sent to the logical
                    printer which SELECTed the file.
```

Notes:

1. *If the READ-SFBLOK was successful, then bytes 2 and 3 of IPRMMSG1 contain the actual length of the SFBLOK.*

2. *If the READ-SFBLOK was unsuccessful because the specified file was not SELECTed, then bytes 2 and 3 of IPRMMSG1 contain 0's.*

3. *If the READ-SFBLOK was unsuccessful because the user buffer was not large enough to hold the SFBLOK or because of a CP paging error, then bytes 2 and 3 of IPRMMSG1 contain the actual length of the SFBLOK.*

Reasons The READ-SFBLOK Function May Sever

The following situations are considered user errors and cause the SPOOL System Service to sever the IUCV path to a logical printer.

- Use of TYPE=1WAY communication, return code=X'80'

- Use of DATA=BUFFER, return code=X'80'.
  The SPOOL System Service READ SFBLOK only allows DATA=PRMMSG.

- Protection or addressing violation for the user buffer, return code=X'08'.
  There is either a storage protection violation involving the user buffer, or the address range specified for the buffer is not addressable.

When the SPOOL System Service READ-SFBLOK does a SEVER, the format of the IPUSER field is:

```
IPUSER + 0:   error code
IPUSER + 1:   X'02'
IPUSER + 2:   IPMSGID from the IUCV parameter list
                   that caused error
```

The READ-XAB Function

This function is used to read the external attribute buffer (XAB) of a file. The READ-XAB functions allows a logical printer to read the information contained in a SELECTed spool file's XAB. For the SPOOL System Service READ-XAB, IPTRGCLS=F'3' and PRMMSG= contains the following information:

```
IPRMMSG1   bytes 0 and 1      spoolid of file
           bytes 2 and 3      buffer size[15]
IPRMMSG2                      buffer address
```

When the SPOOL System Service READ-XAB is complete, the SPOOL System Service replies using an IUCV REPLY, with DATA=PRMMSG, and PRMMSG= containing the following information:

---

15 The maximum valid size is 32,767 bytes.

```
IPRMMSG1  bytes 0 and 1     spoolid
          bytes 2 and 3     length of data
IPRMMSG2  byte  0           flag
          bytes 1 to 3      reserved

        BITS DEFINED in BYTE 0 of IPRMMSG2
      EQU  X'20'    CP I/O error
      EQU  X'08'    user buffer not large enough to
                    hold requested data
      EQU  X'04'    file not SELECTed or *SPL processing
                    for the file stopped and a *SPL PURGE
                    message already sent to the logical
                    printer which SELECTed the file.
```

Note: If there is no XAB then the size returned is zero. If the size returned is greater than the size of the buffer provided, then no data was put in the buffer. Here, the requester must get a buffer large enough to hold the XAB and repeat the READ-XAB function.

Reasons The READ-XAB Function May Sever

The following situations are considered user errors and cause the SPOOL System Service to sever the IUCV path to a logical printer.

- Use of TYPE=1WAY communication, return code=X'80'
- Use of DATA=BUFFER, return code=X'80'.
  The SPOOL System Service READ XAB only allows DATA=PRMMSG.
- Invalid user buffer length, return code=X'08' The buffer length cannot be greater than 32,767 bytes.
- Protection or addressing violation for the user buffer, return code=X'08'.
  There is either a storage protection violation involving the user buffer, or the address range specified for the buffer is not addressable.

When the SPOOL System Service READ-XAB does a SEVER, the format of the IPUSER field is:

```
IPUSER + 0:   error code
IPUSER + 1:   X'03'
IPUSER + 2:   IPMSGID from the IUCV parameter list
                   that caused error
```

The SPOOL System Service to a Logical Printer

When the SPOOL System Service accepts the connection to a logical printer, the SPOOL System Service can send the following types of requests to a logical printer.

SEND
NOTIFY
PURGE

The pages that follow describe these functions.

The SEND Function

The SPOOL System Service uses the SPOOL System Service SEND function to pass printer type commands directly to a logical printer.

The SPOOL System Service SEND function is used for two cases.

1. You specifically issue the CP SEND command to pass data from a non-privileged user to a logical printer.

2. Operator commands are routed to logical printers using the SPOOL System Service SEND because they deal with the internals of logical printers. The commands in this group are those existing class D, Spool Operator, CP commands that control physical system printers. Commands such as:

   - BACKSPAC
   - DRAIN
   - FLUSH
   - QUERY
   - REPEAT
   - SPACE
   - START
   - VARY

   The syntax for all these commands specifically identifies the particular logical printer that is the target of the command. For example, "QUERY LPRT1" where LPRT1 is the name of the logical printer.

The SPOOL System Service SEND function is an IUCV SEND TYPE=1WAY communication, IPTRGCLS=F'9' (DATA=BUFFER, BUFFER=address, and BUFLEN=253).

Note: SPOOL System Service SEND handles the parameter "ALL" as a special keyword for the DRAIN, QUERY, and START commands. When you use "ALL", SPOOL System Service SEND sends the text to every CONNECTed logical printer. Also, when logical printers have the same name, the SPOOL System Service SEND function is done to each of them independently.

It is the responsibility of the logical printer to do the necessary privilege class and authorization checks for all commands sent to it via the SPOOL System Service SEND function. The logical printer uses the SPOOL System Service MESSAGE function to respond to the issuer of the command.

With SPOOL System Service SEND, the SPOOL System Service passes the following information to a logical printer:

```
SPLSNUID    DS    CL8       USERID of the issuer of the command
SPLSNPCS    DS    XL4       Privilege CLASSes of the issuer
SPLSNTXT    DS    CL240     TEXT of the command
            DC    X'15'     end-of-text indicator
```

The end-of-text indicator can occur anywhere within the SPLSNTXT field or in the byte following this field.

The NOTIFY Function

The SPOOL System Service NOTIFY function signals an idle logical printer when a print file is available for processing. The SPOOL System Service NOTIFIES a logical printer via an IUCV SEND TYPE=1WAY communication, IPTRGCLS=F'8', DATA=PRMMSG. (No information is sent to the logical printer concerning the spool file(s) associated with the NOTIFY. There is no significant data in the PRMMSG fields).

With SPOOL System Service NOTIFY, a logical printer does not have to do a SELECT function. If a logical printer is sent a response of *file not available* when SELECTing the next file, then a message is sent to the logical printer when a file becomes available. The SPOOL System Service continues to send notification of other files as they become available.

The SPOOL System Service stops sending notifications to the virtual machine when a SELECT request results in a file being chosen for the logical printer.

The PURGE Function

The SPOOL System Service signals a logical printer to stop processing immediately a print file that the logical printer has SELECTed. For the PURGE function, the SPOOL System Service does an IUCV SEND TYPE=1WAY, IPTRGCLS=F'7', DATA=PRMMSG, and PRMMSG=the spoolid of the file to purge (in bytes 0 and 1 of IPRMMSG1), to the logical printer.

PURGE is done when a PURGE, CHANGE, or TRANSFER command is issued for a file being CONVERTed. The SPOOL System Service checks for a previous PURGE request for the specified file. If the SPOOL System Service receives the PURGE request while I/O is outstanding, the user's buffer is still in use until a READ request is done. The file will not be PURGEd until I/O is complete.

Chapter 11. The Special Message Facility

The Special Message Facility enables a virtual machine to send messages to another virtual machine by issuing the CP SMSG command. The Special Message Facility may be used with the Virtual Machine Communication Facility (VMCF) or with the Inter-User Communication Vehicle (IUCV). However, the sending virtual machine does not need to perform the initialization required by VMCF or IUCV. Initialization is handled by CP and is described later in this topic.

To send a message, a virtual machine need only prepare the message text and issue the class G SMSG command. Parameters on the SMSG command identify the USERID of the receiving virtual machine and specify the message text. The format of the message text must be acceptable to the receiving virtual machine. The SMSG command is described in the *VM/SP CP Command Reference* and *VM/SP HPO CP Command Reference.*

For VMCF: Before the receiving virtual machine can receive special messages via VMCF, it must:

- Enable itself to receive external interrupts.
- Set bit 31 of control register 0 to a value of 1.
- Authorize itself by issuing DIAGNOSE code X'68', AUTHORIZE. The parameter list, VMCPARM, specified with DIAGNOSE code X'68' must contain a pointer to an external-interrupt buffer, should specify a buffer length of 280 bytes, and must have the special message flag (VMCPSMSG) turned on.

  Note that you may receive a message, "Message too large" if you issue the SMSG command from a 3279 or 3287 Model 5 terminal to send a message longer than what the receiving virtual machine has specified.
- Turn on this special message flag (VMCPSMSG) by setting VMCPSMSG to a value of B'1' or by issuing the class G command, SET SMSG ON. For information on using DIAGNOSE code X'68', see "Description of VMCF Functions" and "Invoking VMCF Functions." in the VMCF chapter or the section on DIAGNOSE code X'68' in this manual.

To understand how a special message is presented to the receiving virtual machine via VMCF, see "The SENDX Protocol" in the section, "VMCF Protocol."

For IUCV: Before the receiving virtual machine can receive special messages via IUCV, it must do the following:

- Enable itself to receive external interrupts
- Set bit 30 of control register 0 to a value of 1
- Issue the IUCV DECLARE BUFFER function
- Issue the IUCV CONNECT function to the CP Message System Service
- Turn on the special message flag by issuing the class G command SET SMSG IUCV.

When a virtual machine no longer wishes to accept special messages, it may turn off the special message flag by issuing the command SET SMSG OFF. To resume receiving messages, the virtual machine may issue the command SET SMSG ON or SET SMSG IUCV. CP sends an error message to any virtual machine that tries to send a special message to another virtual machine that is not accepting special messages.

Special messages are queued only as long as the virtual machine is logged on. If the virtual machine sets SMSG off or logs off, this queue of SMSGs is lost. A system IPL also loses this queue of messages for the virtual machine.

CP handles VMCF/IUCV initialization and special message processing as follows. When the SMSG command is issued, CP verifies that no invalid options were specified and that a valid USERID was specified. CP also verifies that the receiving virtual machine is accepting special messages. CP then obtains storage for the message, builds the appropriate parameter list, and sends the message to the receiving virtual machine.

Chapter 12. Single Console Image Facility

The Single Console Image Facility allows one user logged on to a single virtual machine to control multiple disconnected virtual machines. CP prefixes any output coming to the controlling virtual machine, from or on behalf of the originating virtual machine, with the userid of the originating virtual machine. The controlling virtual machine uses the CP class G SEND command to communicate with the virtual machines it is controlling.

The user whose virtual machine is being controlled is the primary user. The user whose virtual machine controls the primary user's virtual machine is the secondary user. The secondary user may run disconnected if he has a valid path to the IUCV Message System Service. Refer to the Chapter 5, "The Message System Service" on page 239 for more information.

Using the Single Console Image Facility

To enable a virtual machine to use the Single Console Image Facility, the installation must specify the userid of the secondary user on the CONSOLE directory control statement of the primary user. See *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference* for a description of the CONSOLE directory control statement.

When the primary user disconnects his virtual machine and the secondary user is logged on, the secondary user receives control of the primary user's virtual machine. Even if the secondary user is not logged on when the primary user disconnects, the secondary user receives control of the disconnected virtual machine whenever he does logon. The primary user can regain control of his virtual machine at his own terminal by entering the LOGON command.

After the primary user disconnects, all console output from the disconnected virtual machine appears on the console of the secondary user if he is logged on. Output from the primary user's disconnected virtual machine is prefixed with the userid of the primary user.

The secondary user uses the CP SEND command to communicate with the primary user's disconnected virtual machine. See *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for a description of the SEND command.

Notes:

1. *When the message, "DMKQCO150A User userid has issued a CP read" is received by the secondary user, the secondary user must reply with a SEND command, sending a CP command to the disconnected user named in the message.*

2. *When the message, "DMKQCO150A User userid has issued a VM read" is received by the secondary user, the secondary user must reply with a SEND command, sending a virtual machine command or a virtual machine reply to the disconnected user named in the message.*

3. *The console attributes of the secondary user are used for the display of messages. For example, if the primary user console is spooled TERM and the secondary user console is spooled NOTERM, only the messages that would normally be displayed with the NOTERM option are displayed at the secondary user's console.*

4. *The language setting of the primary user determines what language is used for error messages and commands. The secondary user should have the same language set as the primary user to avoid unexpected results.*

Chapter 13. Logical Device Support Facility

The Logical Device Support Facility allows an application running in a virtual machine to create within CP one or more logical devices. 3270 extended data streams are supported to enable logical devices to utilize full color, programmed symbol sets, and extended highlighting capabilities. 3284, 6, 7, 8, 9 logical printer devices are supported to allow the presentation of status from a logical device printer. Applications are allowed to create logical 328x printers in addition to logical 327x display devices. Except for the logical device support facility, CP is unaware of the fact that this device has no real existence and is driven by the application program. In particular, CP sees it as a local 3270 device. Any output directed to a logical device is redirected to the virtual machine for which the device was created. The virtual machine can also transfer data to CP to be entered as input from a specific logical device, as if it were interactively produced on a real terminal.

The logical device support facility is made up of two data transfer functions, four control functions, a special external interrupt (code X'2402'), and an external control word for passing control information with the external interrupt.

To implement this facility, functions are invoked using DIAGNOSE code X'7C'. Registers Rx, Rx+1, Ry, and Ry+1 are used to indicate the function, logical device identification, and other function-dependent information.

A special interrupt code (X'2402') is used by module DMKHPS to notify a virtual machine of pending logical device status for a logical device created for that virtual machine. Along with this interrupt, the virtual machine receives a control word at a virtual storage location indicating the ID of the associated logical device and the reason for the interrupt.

Figure 13 is a summary of logical device support facility functions. More complete information about each of these functions is included under "Description of Logical Device Support Facility Functions."

Data is directed to a logical device using the logical device ID. This ID is assigned by CP during execution of the INITIATE function. Data transfer takes place within CP at a channel command level. I/O directed to a logical device proceeds within CP via the normal path for a local device up to the point that DMKIOS is normally called to start I/O. At that point, control passes to DMKHPS to process the CCW string. Channel commands requiring interaction cause external interrupts to the virtual machine for which the associated logical device was created.

The format of data from the virtual machine must conform to 3270 architecture for local devices.

The addresses of logical devices are kept in a table and accessed through an indexing algorithm. There is a limit of 512 logical devices per host and a maximum of 4096 logical devices for the system. There is no limit to the number of hosts as long as the number of logical devices does not exceed 4096 devices.

Function	Description
INITIATE	Initiate logical device communications
ACCEPT	Transfer data written to logical device to virtual machine storage.
PRESENT	Transfer data from virtual machine to CP as input from logical device.
TERMINATE	Drop a specific logical device.
TERMINATE ALL	Drop all logical devices created for this virtual machine.
STATUS	Allows status to be returned to CP after an ACCEPT function is performed.

Figure 13. Summary of Logical Device Support Facility Functions

The VM/Pass-Through Facility licensed program is an example of an application using the logical device support facility. Through the combined support of these two facilities, a VM user attached to system A via a 3270 Display Station can access VM system B as though the display station were locally attached to system B.

Chapter 14. The Virtual Machine Communication Facility

The Virtual Machine Communication Facility (VMCF) is part of the CP component of VM. VMCF provides virtual machines with the ability to send data to and receive data from any other virtual machine.

VMCF is made up of five data transfer functions, seven control functions, a special external interrupt (code X'4001') to asynchronously alert virtual machines to pending messages, and an external interrupt message header to pass control information (and data, at times) to another user.

VMCF is implemented by means of functions invoked using the DIAGNOSE instruction code X'68' and a special 40-byte parameter list called VMCPARM. A VMCF function is indicated by a particular function subcode in the VMCPFUNC field in the parameter list.

Note: Before you can use any other VMCF function, you must use the AUTHORIZE function for communications. Before you can communicate with another user, that user must also have used the AUTHORIZE function.

A special external interrupt (code X'4001') is used by module DMKVMC to notify one virtual machine of a pending transfer of data. This interrupt is also used to synchronize sending and receiving of data.

Along with this interrupt, the virtual machine receives a message header that is logged into a preassigned virtual storage area. This message header is used to define the type of request and to provide data transfer information, such as length of data. The message header is also used to notify the originator of a transaction of the success or failure of the transaction. In this case, the message header includes such information as residual counts and data transfer return codes.

Figure 14 lists the VMCF functions and gives a brief description of each. The functions are described in detail in the section "Descriptions of VMCF Functions".

Messages and data are directed to other virtual machines logically via the userid. Data is transferred in up to 2048-byte blocks from the sending virtual machine's storage to the receiving virtual machine's storage. The amount of data that can be moved in a single transfer is limited only by the sizes of virtual machine storage of the respective virtual machines. Use of real storage is minimal. Only one real storage page per virtual machine (a total of two pages, one for the sender and one for the receiver) need to be locked during the data transfer.

The special message facility uses VMCF to send messages from one virtual machine storage area to another virtual machine storage area. For a description of the special message facility and how it uses VMCF, see Chapter 11, "The Special Message Facility" on page 277.

Function	Code*	Comments
AUTHORIZE	Control	Initializes VMCF for a given virtual machine. Once AUTHORIZE is executed, the virtual machine can execute other VMCF functions and receive messages or requests from other users.
UNAUTHORIZE	Control	Terminates VMCF activity.
SEND	Data	Directs a message or block of data to another virtual machine.
SEND/RECV	Data	Directs a message or block of data to another virtual machine, and requests a reply.
SENDX	Data	Directs data to another virtual machine on a faster but more restrictive protocol than the SEND function.
RECEIVE	Data	Allows you to accept selective messages or data sent via a SEND or SEND/RECV function.
CANCEL	Control	Cancels a message or data transfer directed to another user but not yet accepted by that user.
REPLY	Data	Allows you to direct data back to the originator of a SEND/RECV function, simulating full duplex communication.
QUIESCE	Control	Temporarily rejects further SEND, SENDX, SEND/RECV, or IDENTIFY requests from other users.
RESUME	Control	Resets the status set by the QUIESCE function and allows execution of subsequent requests from other users.
IDENTIFY	Control	Notifies another user that your virtual machine is available for VMCF communication.
REJECT	Control	Allows you to reject specific SEND or SEND/RECV requests pending for your virtual machine.

Figure 14. Virtual Machine Communication Facility (VMCF) Functions

* The word "Data" in this column indicates a data transfer function whereas the word "Control" indicates a VMCF control function.

Using the Virtual Machine Communication Facility

The following discussion presents ideas and suggestions for using the Virtual Machine Communication Facility (VMCF).

VMCF Applications

The VM system with VMCF provides the user with the potential to apply new and different techniques to current applications.

Multitasking Programming

The VMCF functions may be used to multitask virtual machines. Each virtual machine can become a subtask, parallel or otherwise, of another virtual machine. A virtual machine task can be a simple program or a large processor. The VMCF functions provide the WAIT/POST, serialization and communication facilities to control such an environment. The existing VM functions provide efficient scheduling, dispatching, and basic resource controls. The advantage of such an environment is that a user is less restricted by operating system (software) limitations and gains the flexibility of machine languages and hardware.

Resource Sharing

VMCF provides a clear and concise method for sharing and serializing resources between virtual machines. The resources can range from multi-write minidisks to entire processors. The control functions for resource sharing (such as, resource management, serialization) can be contained in a virtual machine.

Virtual Extensions to VM

It is conceivable that functions could be added to VM without altering the control program (CP). A special privilege class virtual machine could be used to provide additional functions to non-privilege class users using the VMCF interface. Similarly, CMS capabilities could be expanded (or at least appear to be expanded) by linking CMS with other virtual machines.

Program Testing

The program testing capabilities offered by VMCF can range from device simulation to teleprocessing network simulation. In particular, VMCF can be used to provide external interactions from one virtual machine to another. A simulated teleprocessing network could be constructed with virtual machines. Each virtual machine would effectively become a node within the network. The network structure could range from a simple tree type structure to a complicated multi-path mesh type structure. The program logic within each node virtual machine would be the same logic as required for a real teleprocessing node. In theory, a reasonably complicated structure could be simulated without requiring the physical hardware.

The significant testing capability provided by VMCF is the ability to link the test system with test/simulation routines in another virtual machine.

Intra-Virtual Machine Communication

Although the VMCF interface is intended for communication from one virtual machine to another it can also be used to communicate within a single virtual machine (wrap connection). The VMCF interface could conceivably be used to link one or more operating system tasks that are logically separated by the software. This would allow task-to-task communication rather than virtual machine-to-virtual machine communication.

Virtual Multiprocessing

The VMCF interface could possibly be used to simulate a virtual multiprocessing environment.

Security and Data Integrity

The VMCF interface provides the following security aids:

- The user doubleword in the external interrupt message header can be used to contain a security code to prevent unwarranted users from accessing a shared data base or other confidential information.
- The AUTHORIZE SPECIFIC option allows a user to restrict messages sent to his virtual machine. This option is useful when slave machines are to communicate only with a host machine. The slave machines can AUTHORIZE SPECIFIC with the host and prevent unwarranted users from clogging their message queues.
- The design of VMCF prevents malicious users from intercepting transactions in process for other users (for example, user D cannot execute a RECEIVE, REPLY, REJECT or CANCEL to a message sent to user B from user A).

The VMCF support module is designed such that a user is always informed of conditions that could threaten the integrity of his own data. The user is notified either with a DIAGNOSE code X'68' return code or data transfer error code. There is no internal buffering of user data within the control program (CP), a message is always retained by either the SOURCE or SINK virtual machine. If a SEND type request fails, the SOURCE still has a copy of the original message. If a SINK REPLY fails, the SINK user still has a copy of the REPLY data. The DIAGNOSE return code or data transfer error code can indicate to a user that a transaction failed. It is up to the user to preserve the associated transaction data. A VMCF user should consider the following notes:

1. The buffer used for SOURCE data in a SEND, SENDX or SEND/RECV request should not be freed or reused until the final response external interrupt is received by the SOURCE.

2. The buffer used for SINK data in a REPLY function can be reused by the SINK after the DIAGNOSE instruction (REPLY) has successfully completed.

3. The user parameter list, VMCPARM, may be re-used upon completion of the DIAGNOSE instruction. At that point the VMCPARM data has been copied to a VMCF control block, VMCBLOK, by the control program. A user should, however, maintain queues of VMCPARM data to associate an external interrupt message header, VMCMHDR, with a particular request.

4. A user should always interrogate the DIAGNOSE return code or data transfer error code for possible error conditions. It is the user's responsibility to determine the types and extent of error recovery. The DIAGNOSE return code 19 for a SOURCE SEND, SEND/RECV or SENDX request indicates that an error was associated with the SINK user and for a SINK RECEIVE or REPLY request indicates that an error was associated with the SOURCE user. The user who receives this return code does not have to invoke error recovery for himself but only be aware that the transaction did not complete successfully because of an error associated with the other user.

Performance Considerations

There are several factors that can affect the performance of VMCF:

- The VMCF support module, DMKVMC, is a pageable CP module. If a user has significant paging activity, it may be advantageous to either lock the module in real storage (CP LOCK command) or alter the CP LOADLIST to make DMKVMC resident.

- It is to a user's benefit to have the user parameter list, VMCPARM, in the same 4K page as the DIAGNOSE code X'68' instruction. This may eliminate a paging operation.

- User support modules using the VMCF interface should be written as reentrant modules and be contained within a CP shared segment whenever possible. This helps reduce CP paging overhead.

- For applications that involve serial message processing, the SENDX function is the most efficient. The SENDX function eliminates the need for the SINK to do a RECEIVE operation.

Note: Overall system VM performance is not affected when VMCF is not being used by an installation.

General Considerations

The SENDX function is a fast way to transfer messages or data and can be used in place of the CP MSG command where the message length exceeds the capacity of the terminal input line. Its use is somewhat restricted in that the maximum data length must be agreed upon by all VMCF users and then remains fixed unless renegotiated.

The SEND and SEND/RECV functions are better suited to transfer high volume data base type information. This type of data transfer requires the flexibility of a wide range of data lengths along with rigorous management and control techniques.

The QUIESCE function allows a virtual machine to discontinue receiving messages. The virtual machine can process those messages already stacked and then use the RESUME function to continue reception. The QUIESCE function also allows a virtual machine to process all queued messages prior to terminating VMCF operation.

The user parameter list, VMCPARM, is designed such that it can be used for any function by simply varying the contents of its fields.

Users should keep copies of VMCPARMs for all requests made via the SEND, SEND/RECV, or SENDX functions. When a final response interrupt is received and the interrupt message header indicates no data transfer errors, the corresponding VMCPARM copy can be released. If a data transfer error is indicated, the copy can be used to reinitiate the transaction.

VMCF Protocol

VMCF provides four types of protocol:

- SEND
- SEND/RECV
- SENDX
- IDENTIFY.

The protocol used to communicate between two virtual machines depends on the application of VMCF and conventions established by virtual machine users authorized to use VMCF. A virtual machine must invoke the AUTHORIZE function before it is allowed to use any of the other functions.

The types of transactions that virtual machines can be involved in are described by a series of VMCF protocols. In these protocols the originating virtual machine is called the "source" virtual machine. The destination virtual machine is called the "sink" virtual machine.

The protocol for a transaction remains in effect for the duration of the transaction.

The SEND Protocol

The SEND protocol defines a one-way transfer of data from source virtual machine storage to sink virtual machine storage. The SEND protocol uses the SEND and RECEIVE functions, as described in Figure 15. The source virtual machine first transfers data to the sink virtual machine. This is done by executing the SEND function which specifies the userid of the sink virtual machine, a message ID, and the address and length of the data being sent. The sink virtual machine receives an external interrupt from CP notifying it of the data transfer request. The sink virtual machine can then respond via the RECEIVE function. The RECEIVE request specifies the address and the length of the SINK buffer that is to receive the data and causes the data to be transferred from source virtual machine storage to sink virtual machine storage. When the data transfer is complete, the source virtual machine receives an external interrupt from CP, indicating that the transaction is complete and that the sink virtual machine has received the data.

All virtual machines authorized to use VMCF can send data using this protocol.

The amount of data transferred is limited only by virtual machine storage size. Data is transferred in blocks of up to 2K (when necessary) and only one real page frame is locked during the data transfer operation.

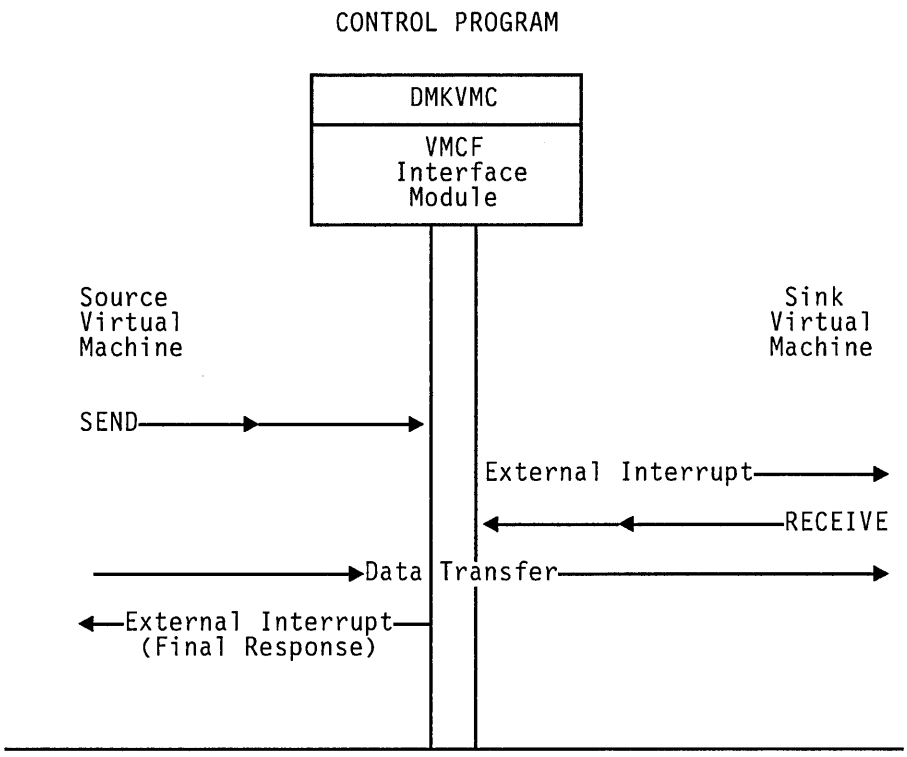


Figure 15. The SEND Protocol

The SEND/RECV Protocol

The SEND/RECV protocol defines a transaction calling for two-way transfer of data, as described in Figure 16. The SEND/RECV protocol uses the SEND/RECV, RECEIVE, and REPLY functions.

The source virtual machine initiates the transaction using the SEND/RECV function. Using an external interrupt, CP notifies the sink virtual machine that there is a message waiting. The sink virtual machine uses the RECEIVE function to cause the data to be transferred from the source virtual machine's storage to the sink virtual machine storage. The sink virtual machine now uses the REPLY function to cause data to be transferred from its storage to the source virtual machine's storage. When the REPLY function completes processing, CP causes an external interrupt in the source virtual machine, notifying it that the transaction is complete.

The SEND/RECV request requires that the source virtual machine specify the address and length of the data to be transferred and the address where data is expected from the REPLY function. (Both addresses are in source virtual machine storage.) These addresses, along with the length of the data to be transferred, are specified via the VMCPARM parameter list, described below.

When RECEIVE is issued by the sink virtual machine in response to the SEND/RECV request, VMCPARM contains the address in sink virtual

machine storage where data is to be received. Finally, when the REPLY request is issued, VMCPARM contains the address in the sink virtual machine storage from which data is to be transferred.

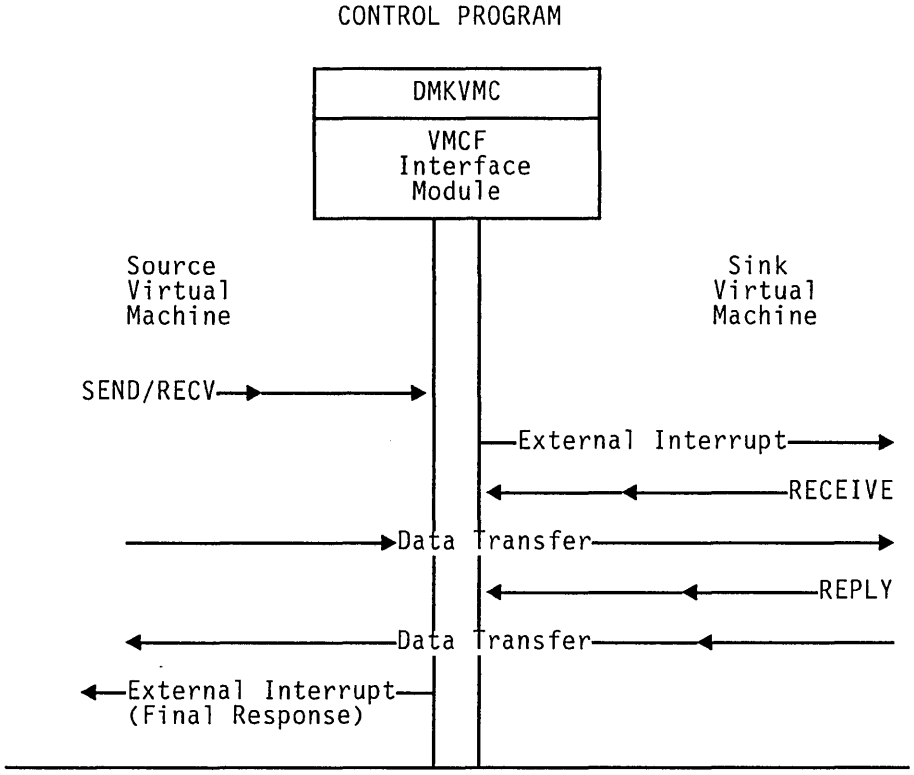


Figure 16. The SEND/RECV Protocol

The SENDX Protocol

The SENDX protocol defines a transaction calling for an expedited one-way transfer of data. Figure 17 shows the SENDX protocol graphically. SENDX differs from the SEND protocol in that the sink virtual machine need not issue the RECEIVE function; data is transferred from source virtual machine storage to sink virtual machine storage at the same time the external interrupt from CP notifies the sink virtual machine of the transaction. Data sent by the source virtual machine is placed in the external interrupt buffer of the sink virtual machine.

Virtual machines using the SENDX protocol are responsible for specifying the userid for the sink virtual machine, a message ID, the address and length of the data being sent, and the external interrupt buffer address and data length for the sink virtual machine. A virtual machine to be used as a sink virtual machine with the SENDX protocol must specify this information via VMCPARM when that virtual machine issues the AUTHORIZE function. The data length specified must be at least as long as the maximum amount of data to be transferred during a transaction; it need not be limited to the usual 40-byte external interrupt buffer. Effective use of the SENDX protocol requires that VMCF users agree on a maximum

size for SENDX data and then issue the AUTHORIZE function with the appropriate external interrupt buffer size.

If the sink virtual machine has not provided enough SENDX buffer area in the external interrupt buffer, CP notifies the source virtual machine that the transaction was not completed.

When a SENDX data transfer is complete, CP directs a response external interrupt to the source virtual machine, notifying it that the transaction is complete.

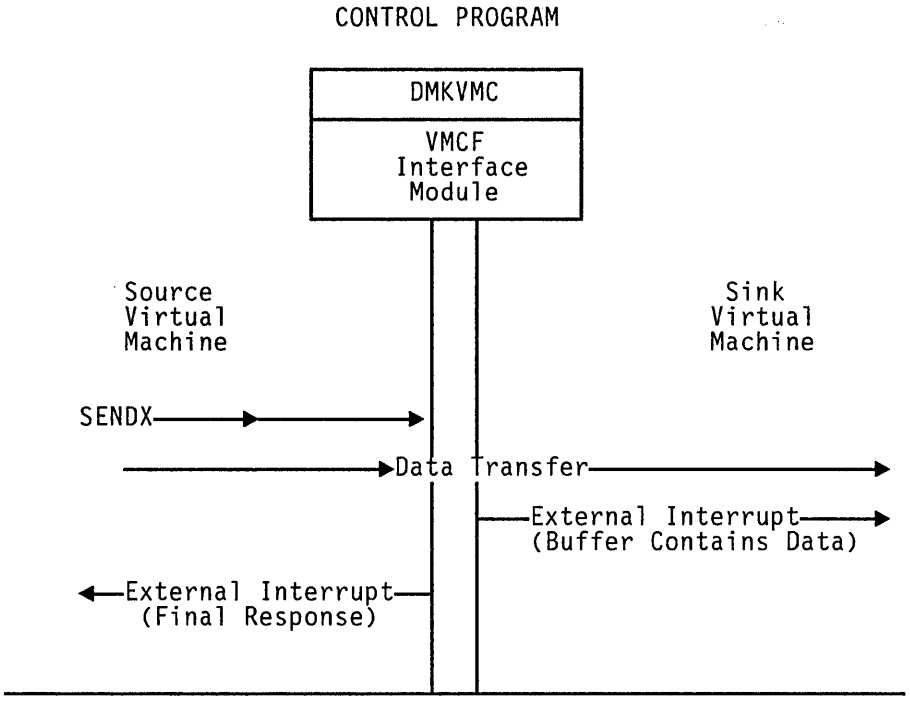


Figure 17. The SENDX Protocol

The IDENTIFY Protocol

The IDENTIFY protocol defines a means for virtual machines to identify themselves to other virtual machines by passing user-defined control information via a standard VMCF message header. Figure 18 shows the IDENTIFY protocol graphically.

When the IDENTIFY function is issued, CP directs an external interrupt to the sink virtual machine. Along with the external interrupt, the sink virtual machine receives a standard VMCF message header that contains user-defined information. The IDENTIFY protocol does not cause a response external interrupt to be directed to the source virtual machine.

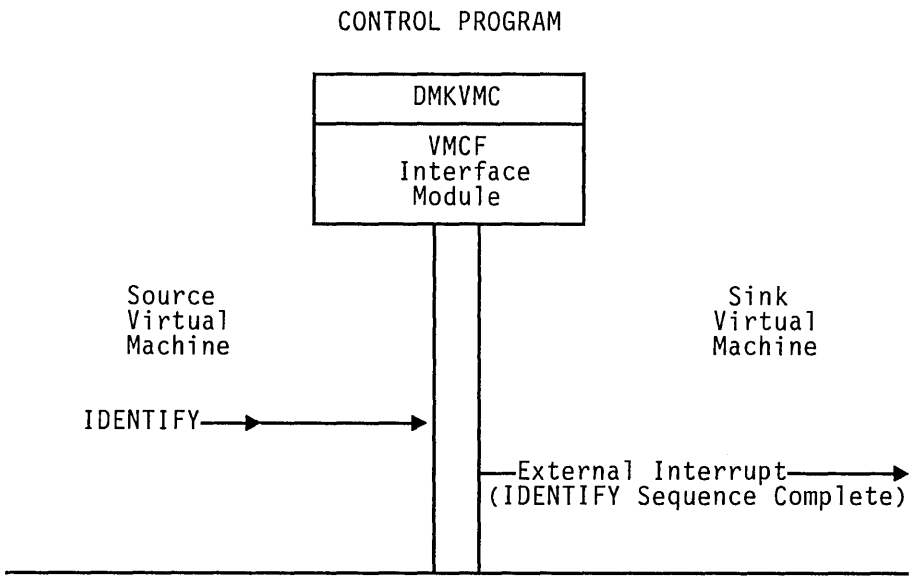


Figure 18. The IDENTIFY Protocol

Descriptions of VMCF Functions

There are two types of VMCF functions:

- Control functions
- Data transfer functions.

The Control Functions

The VMCF control functions allow efficient management of data transfer operations from your virtual machine console. The control functions are:

- AUTHORIZE
- UNAUTHORIZE
- CANCEL
- QUIESCE
- RESUME
- IDENTIFY
- REJECT.

AUTHORIZE: DIAGNOSE Code X'68' Subcode X'0000'

AUTHORIZE enables VMCF for a virtual machine; once AUTHORIZE has been executed, the virtual machine can execute other VMCF functions and receive messages and data from other authorized VMCF virtual machines. It is possible to specify three options with the AUTHORIZE function: SPECIFIC, PRIORITY, and VMCPSMSG.

The SPECIFIC option authorizes communication with a specific virtual machine. Any messages sent to the virtual machine from other than the specified virtual machine will be rejected. The SPECIFIC option can be used in an application where virtual machines desire to communicate with a master controller but not among themselves. Under the special message facility, CP is authorized with every virtual machine that is to receive messages sent with the SMSG command. Virtual machines that are to receive messages must authorize themselves.

The PRIORITY option allows a virtual machine to authorize the receipt of priority messages. A virtual machine is allowed to send priority messages to another virtual machine only if the other virtual machine is authorized to receive priority messages. A priority message is one that is queued ahead of nonpriority messages and therefore accepted first.

When you execute the AUTHORIZE function, you must specify the address and length of the external interrupt buffer for your virtual machine. The buffer must be large enough to contain a fixed message header (40 bytes). The message header identifies messages sent by other virtual machines or responses to messages you might send to your own virtual machine.

If you are going to accept SENDX-type communications, you must specify the size of the external interrupt buffer as 40 plus the maximum size of SENDX data that you plan to accept. This has the effect of authorizing SENDX protocol. That is, a virtual machine may receive data along with the external interrupt in its external interrupt buffer. When a virtual machine sends data to another virtual machine via the SENDX function the data must fit in that virtual machine's external interrupt buffer or the function is rejected. It is recommended to specify a buffer length of 280 bytes.

Note: You may receive a message, "Message too large" if you issue the SMSG command from a 3279 or 3287 Model 5 terminal to send a message longer than what the receiving virtual machine has specified.

Any AUTHORIZE options in effect can be reset or changed by executing the AUTHORIZE function again. If there are errors during execution of the AUTHORIZE function, a virtual machine's authorization status is not changed.

UNAUTHORIZE: DIAGNOSE Code X'68' Subcode X'0001'

UNAUTHORIZE terminates VMCF activity for a virtual machine. The UNAUTHORIZE function causes any stacked or queued messages associated with the virtual machine to be purged. A virtual machine should execute the QUIESCE function before executing UNAUTHORIZE if messages that are already queued are to be handled. When a virtual machine executing UNAUTHORIZE has pending final response external interrupts, the interrupts are purged. If a virtual machine has pending SEND external interrupts from another source virtual machine, a RESPONSE interrupt is reflected to the source indicating that the virtual machine is no longer available.

CANCEL: DIAGNOSE Code X'68' Subcode X'0006'

CANCEL cancels a message or data transfer pending for but not accepted by another VMCF virtual machine. A virtual machine can CANCEL messages it originates with SEND, SENDX, or SEND/RECV functions. A message cannot be canceled if any of the following conditions exist:

- The request was SENDX or IDENTIFY and the sink had already received the SEND external interrupt.
- The request was SEND and the sink had already executed the RECEIVE or REJECT functions.
- The request was SEND/RECV and the sink had already executed the REPLY or REJECT functions.

If the original request was SEND/RECV and the sink virtual machine had executed the RECEIVE function but not the REPLY, the REPLY can be canceled. A virtual machine is notified of this condition with a DIAGNOSE return code. (For a description of the return codes, see Figure 19.)

QUIESCE: DIAGNOSE Code X'68' Subcode X'0008'

QUIESCE temporarily rejects SEND, SENDX, SEND/RECV, or IDENTIFY requests from other virtual machines. QUIESCE allows a virtual machine to receive any stacked or queued messages but reject further SEND, SENDX, IDENTIFY, or SEND/RECV requests from other virtual machines. QUIESCE can be used to indicate to other virtual machines that the virtual machine is in QUIESCE status, authorized for communication but not able to accept messages at this time (e.g., entering slowdown, my buffers are full, try again later). The IDENTIFY function could be used to inform other virtual machines that a particular user is no longer in QUIESCE status. You should execute the QUIESCE function before executing the UNAUTHORIZE function to avoid losing messages (see "UNAUTHORIZE: DIAGNOSE Code X'68' Subcode X'0001'.") A virtual machine can reset the QUIESCE status (exit slowdown) by executing the RESUME function. (See "RESUME: DIAGNOSE Code X'68' Subcode X'0009'.") A virtual machine in QUIESCE status may continue to send messages to other virtual machines. QUIESCE status for a virtual machine only affects messages sent from other virtual machines.

RESUME: DIAGNOSE Code X'68' Subcode X'0009'

RESUME cancels the QUIESCE status, allowing your virtual machine to resume reception of VMCF requests from other virtual machines. You can use the IDENTIFY function to inform other virtual machines that your virtual machine is no longer in QUIESCE status. (See "IDENTIFY: DIAGNOSE Code X'68' Subcode X'000A'.")

IDENTIFY: DIAGNOSE Code X'68' Subcode X'000A'

IDENTIFY notifies another virtual machine that your virtual machine is available for VMCF communication. Use the IDENTIFY function after issuing the AUTHORIZE function or after your virtual machine has been in the VMCF QUIESCE state and you have issued the RESUME function. IDENTIFY causes an external interrupt to be stacked for a specified virtual machine. The virtual machine executing the IDENTIFY function specifies the userid of the user to receive the external interrupt. The external interrupt identifies the virtual machine executing the IDENTIFY function. The IDENTIFY function is provided to inform a host or controller virtual machine that a virtual machine is activated (logged on) and ready for VMCF communication. The IDENTIFY function can also be used to inform other virtual machines that your virtual machine has exited QUIESCE state. There is no response external interrupt associated with the IDENTIFY function.

The IDENTIFY function can also be used to pass virtual machine defined control information. The fields in the VMCF parameter list, VMCPARM, not used by the IDENTIFY function may be used to contain additional virtual machine data.

REJECT: DIAGNOSE Code X'68' Subcode X'000B'

REJECT selectively rejects pending SEND or SEND/RECV requests from other VMCF virtual machines. REJECT causes a response external interrupt to be reflected to the originator of a message. The external interrupt indicates to the originator that the message was rejected. The user doubleword within the external interrupt header may tell a user why the message was rejected. When the user of a virtual machine executes the REJECT function, he specifies within the VMCF parameter list, VMCPARM, the message ID of the message to be rejected. A virtual machine cannot reject a message sent with the SENDX function since the message is received at the same time the external interrupt is received. The REJECT function can be executed as response to either SEND or SEND/RECV requests.

The Data Transfer Functions

The data transfer functions are:

- SEND
- SEND/RECV
- SENDX
- RECEIVE
- REPLY.

These operations involve the movement of data from one virtual machine storage to another virtual machine storage.

SEND: DIAGNOSE Code X'68' Subcode X'0002'

SEND directs a message or block of data to another virtual machine. Specify the virtual address and length of data to be sent within the user parameter list, VMCPARM. Also, specify in the parameter list a message ID to be associated with the message and the userid of the user to receive the message data. You can also send a doubleword of data to be transmitted within the external interrupt message header (refer to the section "VMCF User Doubleword"). If the SEND function is executed with a data length of zero, only the user doubleword is transmitted to the sink virtual machine. The sink virtual machine can then respond with a RECEIVE function (zero length) and pass back a doubleword of data to the source virtual machine. The external interrupt message header identifies the SEND request. When the sink virtual machine executes a RECEIVE function, the message is transmitted from the source virtual machine storage to the sink virtual storage. There is no internal buffering of data within the control program (CP). All data is transferred in 2K blocks from virtual storage to virtual storage. Data is transferred in 2K blocks to test for STORE/FETCH protection violations. When the data transfer function is complete, the source virtual machine receives a response external interrupt indicating that the SEND request is complete. The sink virtual machine receives a DIAGNOSE code X'68' return code indicating that the RECEIVE function is complete. The return code can indicate error conditions associated with the RECEIVE function or normal completion.

The sink virtual machine has the option to reject a message rather than execute the RECEIVE function. (See "REJECT: DIAGNOSE Code X'68' Subcode X'0011'.") The source virtual machine may cancel a SEND request before the sink virtual machine has executed a RECEIVE function or REJECT function. (See "CANCEL: DIAGNOSE Code X'68' Subcode X'0006'.")

If you are executing the SEND function, you may specify the PRIORITY option. The PRIORITY option causes the external interrupt for the sink virtual machine to be queued ahead of all other nonpriority external interrupts. If there are other PRIORITY external interrupts pending for the sink virtual machine, the queuing is done in a first in first out manner. That is, PRIORITY interrupts are queued FIFO among themselves but ahead of all nonpriority interrupts.

SEND/RECV: DIAGNOSE Code X'68' Subcode X'0003'

SEND/RECV provides the capability to both send and receive data in a single VMCF transaction. The SEND/RECV function causes an external interrupt to be queued for the sink virtual machine. When the sink virtual machine receives the external interrupt, it can respond with the RECEIVE function. The RECEIVE function causes data to be transferred from the source virtual storage to sink virtual storage. The sink virtual machine can then respond with a REPLY function. The REPLY function causes data to be transferred from specified sink virtual storage to a REPLY buffer in the source virtual storage. The source virtual machine then receives a response external interrupt indicating that the SEND/RECV request is complete.

When the source virtual machine executes the SEND/RECV function it specifies the address and length of both the SEND buffer and REPLY buffer. The address and length specifications are contained within the user parameter list, VMCPARM. The user parameter list also contains a message ID and userid of the user to receive the data. (See the "VMCPARM Parameter List.")

The source virtual machine can cancel a previously executed SEND/RECV request provided the sink virtual machine has not yet executed the REPLY or REJECT function. If the sink virtual machine has already executed the RECEIVE function, only the REPLY can be canceled. (See "CANCEL: DIAGNOSE Code X'68' Subcode X'0006'.")

The sink virtual machine can execute the REJECT function in response to the SEND/RECV request and cause the entire operation to be terminated. (See "REJECT: DIAGNOSE Code X'68' Subcode X'0011'.")

The sink virtual machine can respond to a SEND/RECV request with the REPLY function without executing the RECEIVE function. This has the effect of informing the source virtual machine that the sink virtual machine cannot accept data but that it can send data. The source virtual machine could have executed the SEND/RECV function only as a means to solicit data from the sink virtual machine. The application of this protocol is up to VMCF users. The user doubleword can be used as a means to control such an application. (See "VMCF User Doubleword.")

You can execute a SEND/RECV request using the PRIORITY option. The PRIORITY option causes the sink external interrupt for the SEND/RECV request to be queued ahead of any other nonpriority external interrupts. Response external interrupts directed to the source of a PRIORITY message are also queued in priority order.

SENDX: DIAGNOSE Code X'68' Subcode X'0004'

SENDX directs data to another virtual machine via a faster but more restrictive protocol than the SEND function. SENDX function data reaches the sink virtual machine at the same time the SEND external interrupt reaches the sink. To use the SENDX function, the sink virtual machine must have an external interrupt buffer large enough to contain both the standard message header and the data. The size of the external interrupt buffer is specified when you execute the AUTHORIZE function. Attempts to execute SENDX are rejected when the sink virtual machine's external interrupt buffer is not large enough to contain the data. After the sink virtual machine receives the SEND external interrupt and data, a response external interrupt is directed to the source virtual machine. The SENDX function eliminates the need for a sink virtual machine to execute a RECEIVE function.

A SENDX request can be canceled by the source virtual machine provided the SENDX external interrupt has not yet been reflected to the sink virtual machine. (See "CANCEL: DIAGNOSE Code X'68' Subcode X'0006'.")

Specify the SENDX buffer address and length in the user parameter list, VMCPARM. The message ID and userid of the sink virtual machine are also specified in VMCPARM.

The SENDX function can be executed with the PRIORITY option allowing the SEND external interrupt to be queued ahead of all nonpriority external interrupts for the sink virtual machine.

A SENDX request cannot be rejected by the sink virtual machine since the message is received at the same time the external interrupt is received.

You can execute the SENDX function with a zero data length causing only the message header and user doubleword to be transmitted.

RECEIVE: DIAGNOSE Code X'68' Subcode X'0005'

RECEIVE allows you to selectively accept messages or data sent via the SEND or SEND/RECV functions. You must specify in the user parameter list, VMCPARM, the virtual address and length of the RECEIVE buffer. The parameter list also contains the message ID of the message to be received and userid of the virtual machine that originated the SEND or SEND/RECV request. When a virtual machine has more than one message pending, the RECEIVE function can be executed to select messages in any order by message ID.

You can execute the REJECT function to reject messages sent by other virtual machines. The REJECT function terminates the SEND or SEND/RECV request. (See "REJECT: DIAGNOSE Code X'68' Subcode X'000B'.")

You can execute the RECEIVE function in response to a SEND/RECV request and then execute a REJECT function rather than a REPLY. The user doubleword passed back with the REJECT function could indicate "RESEND", for example, if the original data was not received correctly (depending on how you want to use the protocol).

REPLY: DIAGNOSE Code X'68' Subcode X'0007'

REPLY allows you to direct data back to the sender of a SEND/RECV function. This simulates full duplex communication. The REPLY function is used with the SEND/RECV function. A user who receives a SEND/RECV external interrupt normally responds by executing the RECEIVE function. The RECEIVE function causes data to be transferred from the source virtual storage to the sink virtual storage. The sink virtual machine can then respond with the REPLY function causing data to be transferred from specified sink virtual storage to the source virtual storage. The REPLY function causes a response external interrupt to be reflected to the source virtual machine.

The user parameter list, VMCPARM identifies the virtual buffer address and length of reply data. When the REPLY function is executed, the user parameter list, VMCPARM, also contains the message ID and the userid of the virtual machine to receive the reply.

The REPLY function can be executed with a zero data length indicating no response. You can transmit a reply, zero length or otherwise, using the user doubleword.

A reply can be executed in response to a SEND/RECV request without executing the RECEIVE function. This indicates that you do not want to receive the message but may want to send a reply. A reply of zero length could be executed simply to terminate the SEND/RECV request. The application of the REPLY function is a user decision. It must be used to terminate a SEND/RECV request, however, unless the REJECT function is executed. (See "REJECT: DIAGNOSE Code X'68' Subcode X'0011'.") The reply is complete when the source virtual machine receives the external interrupt response.

A REPLY function cannot be executed in response to a SEND request, this is a protocol violation.

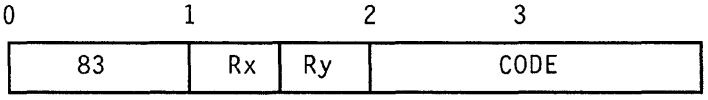
Invoking VMCF Functions

VMCF functions are invoked by means of:

- DIAGNOSE code X'68' subcodes
- The VMCPARM parameter list
- External interrupt code X'4001'
- The external interrupt message header.

DIAGNOSE Code X'68'

All VMCF functions are invoked from within assembler language programs by means of DIAGNOSE code X'68':



where:

83
is X'83' and interpreted by the assembler as the DIAGNOSE instruction.

Note: There is no mnemonic for DIAGNOSE.

Rx
specifies a register containing the address of the VMCPARM parameter list.

Ry
is a register in which the return code is stored.

CODE
is X'0068' and specifies that you are requesting execution of a VMCF.

The VMCPARM Parameter List

The Rx register of DIAGNOSE code X'68' contains the address of a parameter list, VMCPARM. This parameter list is used to specify the VMCF function to be executed, along with other information required by VMCF to execute that function. The address of VMCPARM must be doubleword-aligned. The following is the format of the VMCPARM parameter list and a description of each of the fields in that list.

0	V*1	V*2	VMCPFUNC	VMCPMID
8	VMCPUSER			
10	VMCPVADA			VMCPLENA
18	VMCPVADB			VMCPLENB
20	VMCPUSE			

where:

V*1 (VMCPFLG1)
is a flag byte used to specify options associated with a particular function.

This flag byte can be set to the following values:

VMCPAUTS (X'80')
Indicates, for the AUTHORIZE function, an AUTHORIZE SPECIFIC request. When this bit is set, the VMCPUSER field must contain the userid of the sink virtual machine. The status of the specified sink virtual machine is not checked by the control program (CP) at this time.

VMCPPRTY (X'40')
Indicates, for SEND, SEND/RECV, SENDX, and IDENTIFY requests, a PRIORITY message request. For an AUTHORIZE request, it indicates an AUTHORIZE PRIORITY request. You cannot send PRIORITY messages to another virtual machine unless that virtual machine has been authorized for PRIORITY messages. The SEND and RESPONSE external interrupts for a PRIORITY message are queued ahead of pending nonpriority external interrupts.

VMCPSMSG (X'20')
Indicates that the virtual machine accepts messages sent via the SMSG command.

Bits 3 through 7 are reserved for IBM use.

V*2 (VMCPFLG2)
Reserved for IBM use, and therefore should be X'00' initially.

VMCPFUNC
Contains the halfword DIAGNOSE code X'68' subcode that defines the VMCF function being requested as follows:

Command	Hexadecimal Subcode	Function
VMCPAUTH	X'0000'	AUTHORIZE
VMCPUAUT	X'0001'	UNAUTHORIZE
VMCPSEND	X'0002'	SEND
VMCPSENR	X'0003'	SEND/RECV
VMCPSENX	X'0004'	SENDX
VMCPRECV	X'0005'	RECEIVE
VMCPCANC	X'0006'	CANCEL
VMCPREPL	X'0007'	REPLY
VMCPQUIE	X'0008'	QUIESCE
VMCPRESM	X'0009'	RESUME
VMCPIDEN	X'000A'	IDENTIFY
VMCPRJCT	X'000B'	REJECT

VMCPMID
Contains a unique message identifier associated with a transaction. The source virtual machine must originate the message ID for SEND, SEND/RECV, and SENDX requests. The message ID is used by the sink virtual machine (along with VMCPUSER) to respond to the source request with a RECEIVE, REPLY, or REJECT request. The message ID allows the sink virtual machine to selectively RECEIVE, REPLY, or REJECT messages when more than one message is enqueued. The message ID is used by both the source and sink as a unique identification for all messages. You may send messages with the same message ID to multiple users; you cannot send multiple messages with the same message ID to one user. Once a transaction is completed, however, the message ID may be reused.

VMCPUSER
Specifies the userid of the sink virtual machine for SEND, SEND/RECV, SENDX, IDENTIFY, and CANCEL requests and the userid of the source virtual machine for RECEIVE, REPLY, and REJECT requests. The sink virtual machine uses this field in combination with the message ID (VMCPMID) to respond to source requests. When the original source parameter list VMCPARM is passed to the sink as the message header VMCMHDR, the userid is changed from sink to source.

This field is also used to specify the SPECIFIC userid for an AUTHORIZE SPECIFIC request.

VMCPVADA

Contains one of four addresses, depending upon which VMCF function is requested.

For SEND, SEND/RECV, and SENDX requests, VMCPVADA contains the address of the source virtual machine data. For RECEIVE requests, VMCPVADA contains the address of a sink virtual machine RECEIVE buffer. For REPLY requests, VMCPVADA contains the address in sink virtual machine storage where REPLY data is located. For an AUTHORIZE request, VMCPVADA specifies the address of the virtual machine external interrupt buffer.

The length of the associated data or buffer is specified in the VMCPLENA field.

VMCPLENA

Contains the length of the data sent by a user, the length of a RECEIVE buffer, or the length of an external interrupt buffer, whichever is specified in the field VMCPVADA. The size of the value specified in VMCPLENA is restricted only by virtual machine storage size.

The sink virtual machine can use the value in this field as the data length for RECEIVE operations.

VMCPVADB

Contains the address of a source virtual machine's REPLY buffer for a SEND/RECV request. When the sink virtual machine issues a REPLY in response to a SEND/RECV from the source virtual machine, the REPLY data is moved in this buffer. The length of the REPLY buffer is contained in the field VMCPLENB.

VMCPLENB

Specifies the length of the source virtual machine's REPLY buffer. The sink virtual machine uses this field to determine the maximum length of the REPLY. A corresponding field within the response message header contains a residual data count. The source virtual machine uses this residual count to determine the length of the sink reply. The original REPLY buffer length (less the residual count) is the length of the REPLY from the sink virtual machine.

VMCPUSE

Contains the VMCF user doubleword. The user doubleword is transmitted to the sink virtual machine in the SEND message header for SEND, SEND/RECV, SENDX, and IDENTIFY requests. For RECEIVE, REPLY, and REJECT requests, the user doubleword is transmitted to the source virtual machine within the RESPONSE message header. The sink virtual machine can transmit the user doubleword to the source virtual machine with REJECT or REPLY requests only if the original

request was a SEND/RECV. The user doubleword is transmitted only with requests that result in SEND or RESPONSE external interrupts.

The following chart summarizes the VMCPARM fields required for execution of each of the VMCF functions. Possible return codes associated with each function are also listed. A discussion of the return codes and their meanings can be found in the section "DIAGNOSE Code X'68' Return Codes".

VMCF Function	Applicable VMCPARM Parameters	Return Codes
AUTHORIZE	VMCPFLG1 - SPECIFIC/PRIORITY option VMCPFUNC - X'0000' - subcode VMCPUSER - SPECIFIC userid VMCPVADA - external interrupt buffer address VMCPLENA - external interrupt buffer length	0,1,2,6,15
UNAUTHORIZE	VMCPFUNC - X'0001' - subcode	0,2,4,15
SEND	VMCPFLG1 - PRIORITY option VMCPFUNC - X'0002' - subcode VMCPMID - message identifier VMCPUSER - sink userid VMCPVADA - SEND data address VMCPLENA - SEND data length VMCPUSE - user doubleword (See Note)	0,1,2,4,5,8,9, 10,15,18
SEND/RECV	VMCPFLG1 - PRIORITY option VMCPFUNC - X'0003' - subcode VMCPMID - message identifier VMCPUSER - sink userid VMCPVADA - SEND data address VMCPLENA - SEND data length VMCPVADB - REPLY buffer address VMCPLENB - REPLY buffer length VMCPUSE - user doubleword	0,1,2,4,5,8,9, 10,15,18
SENDX	VMCPFLG1 - PRIORITY option VMCPFUNC - X'0004' - subcode VMCPMID - message identifier VMCPUSER - sink userid VMCPVADA - SEND data address VMCPLENA - SEND data length VMCPUSE - user doubleword (See Note)	0,1,2,4,5,7,8, 9,10,15,18

Figure 19 (Part 1 of 2). VMCF Functions, Parameters, and Return Codes

VMCF Function	Applicable VMCPARM Parameters	Return Codes
RECEIVE	VMCPFUNC - X'0005' - subcode VMCPMID - message identifier VMCPUSER - source userid VMCPVADA - RECEIVE buffer address VMCPLENA - RECEIVE buffer length VMCPUSE - user doubleword	0,1,3,2,4,5,6, 12,13,15,16,17
CANCEL	VMCPFUNC - X'0006' - subcode VMCPMID - message identifier VMCPUSER - sink userid	0,2,3,4,5,11, 12,14,15,20
REPLY	VMCPFUNC - X'0007' - subcode VMCPMID - message identifier VMCPUSER - source userid VMCPVADA - REPLY data address VMCPLENA - REPLY data length VMCPUSE - user doubleword	0,1,2,3,4,5,6, 12,13,15,16, 17,19
QUIESCE	VMCPFUNC - X'0008' - subcode	0,2,4,15
RESUME	VMCPFUNC - X'0009' - subcode	0,2,4,15
IDENTIFY	VMCPFLG1 - PRIORITY option VMCPFUNC - X'000A' - subcode VMCPUSER - sink userid VMCPUSE - user doubleword (See Note)	0,2,4,5,9,10 15,18
REJECT	VMCPFUNC - X'000B' - subcode VMCPMID - message identifier VMCPUSER - source userid VMCPUSE - user doubleword	0,2,3,4,12,13, 15

Figure 19 (Part 2 of 2). VMCF Functions, Parameters, and Return Codes

Note: Fields within the user parameter list that are not used by a particular function may be used to contain additional user data. The data, however, can only be passed to the sink virtual machine by the source virtual machine. The REPLY buffer address and length fields (VMCPVADB+VMCPLENB) may be used to transmit additional user data for SEND and SENDX requests. All fields except VMCPFLG1, VMCPFLG2, VMCPFUNC, and VMCPUSER may be used to pass control information with an IDENTIFY request.

External Interrupt Code X'4001'

External interrupt code X'4001' is a special interrupt code recognized by CP as part of a VMCF transaction. Just as virtual machines use the DIAGNOSE instruction to communicate with CP, so too CP uses this interrupt code to communicate with virtual machines. External interrupt code X'4001' and DIAGNOSE code X'68' provide the mechanism VMCF uses to synchronize message processing.

The External Interrupt Message Header

Associated with external interrupt code X'4001' is a storage area referred to as the external interrupt message header. The external interrupt message header (VMCMHDR) contains the control information required to SEND and RECEIVE messages. The fields within the message header are, for the most part, a copy of VMCPARM parameter list fields.

Before the receiving virtual machine can receive special messages via VMCF, it must

- Enable itself to receive external interrupts
- Set bit 31 of control register 0 to a value of 1
- Authorize itself by issuing DIAGNOSE code X'68', AUTHORIZE.

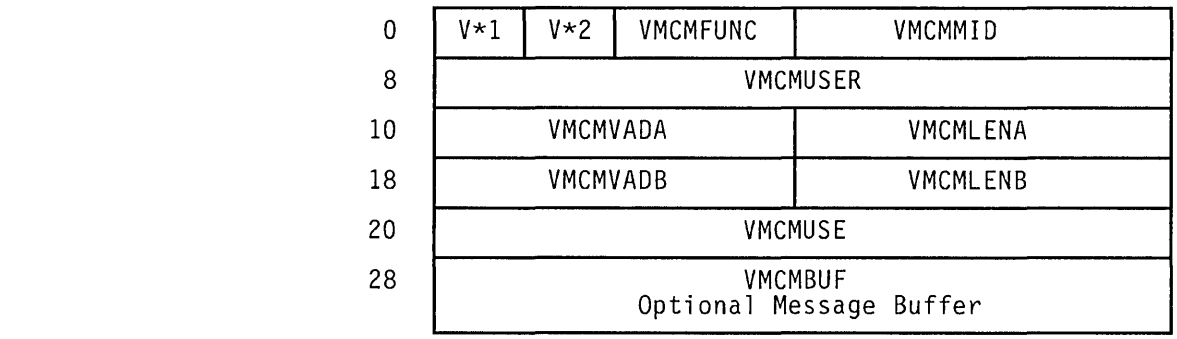
The parameter list, VMCPARM, specified with DIAGNOSE code X'68' must

- Contain a pointer to an external-interrupt buffer
- Specify a buffer length of 169 bytes
- Have the special message flag (VMCPSMSG) turned on.

The receiving virtual machine may turn on this flag by setting VMCPSMSG to a value of B'1'. Optionally, the receiving virtual machine may turn on the special message flag by issuing the class G command, SET SMSG ON. For information on using DIAGNOSE code X'68', see "Description of VMCF Functions" and "Invoking VMCF Functions."

CP passes the external interrupt buffer (containing the external interrupt message header) to the user's interrupt handler for processing. The user must specify the address and length of this buffer when he executes the AUTHORIZE function. Then, when the user sends or receives messages, CP knows the address of the buffer and passes it to the appropriate interrupt handler routine.

Fields VMCMFUNC through VMCMUSE correspond to the fields VMCPFUNC through VMCPUSE in the VMCPARM parameter list transmitted by the source virtual machine. The format of the message header and optional SENDX data buffer is:



where:

V*1 (VMCMSTAT)
: is a status byte associated with the message header. The bits within the status byte are defined as follows:

  VMCMRESP (X'80')
  : Indicates final external interrupt (transaction complete). This bit is set for all RESPONSE external interrupts and the SEND external interrupt resulting from an IDENTIFY request.

  VMCMRJCT (X'40')
  : This bit is set in a RESPONSE external interrupt to indicate that the sink virtual machine rejected the message via the REJECT function.

  VMCMPRTY (X'20')
  : This bit is set in both SEND and RESPONSE external interrupts to indicate a priority message. A virtual machine must be authorized for priority messages before it can receive them.

V*2 (VMCMEFLG)
: Contains a data transfer error code indicating success or errors associated with a data transfer operation. (Refer to the section "Data Transfer Error Codes".)

VMCMFUNC
: Contains the function subcode of the original request. The sink virtual machine uses this field to determine the type of request. The possible subcodes are:

  VMCPSEND X'0002' - SEND
  VMCPSENR X'0003' - SEND/RECV
  VMCPSENX X'0004' - SENDX
  VMCPIDEN X'000A' - IDENTIFY

VMCMMID
: Contains the message ID associated with the original source request.

VMCMUSER
: Contains the userid of the source virtual machine for SEND external interrupts and the userid of the sink virtual machine for RESPONSE external interrupts. If a SMSG command was issued, "SYSTEM" appears in this field.

VMCMVADA
: Contains the address of the original SEND data for SEND requests. This field would normally have no meaning to the sink virtual machine.

VMCMLENA
: Indicates the length of SEND data for SEND external interrupts. It indicates a data transfer residual count for RESPONSE external interrupts.

VMCMVADB
: Contains the virtual address of the REPLY buffer for SEND/RECV requests. This field has no meaning to the sink virtual machine.

VMCMLENB
: Contains the length of the source virtual machine REPLY buffer for SEND/RECV external interrupts; contains the residual REPLY count for RESPONSE external interrupts. The sink virtual machine uses this field to determine the maximum length of the REPLY; the source virtual machine uses this field to determine the length of the sink virtual machine REPLY data.

VMCMUSE
: Contains the user doubleword, which is transmitted to the sink virtual machine with SEND external interrupts and to the source virtual machine with RESPONSE external interrupts. If a SMSG command was issued, this field contains the virtual machine identifier of the issuer of that command.

VMCMBUF
: This is the optional data buffer used by the SENDX function. The data sent with the SENDX function is moved into this buffer. The buffer size is specified when a virtual machine executes the AUTHORIZE function.

VMCF User Doubleword

VMCF provides a doubleword for user data that can be transmitted within the external interrupt message header. A user supplies the doubleword of data within the parameter list (VMCPARM) for certain VMCF requests (that is, SEND, SENDX, SEND/RECV, RECEIVE, REPLY, IDENTIFY, and REJECT). You can use the user doubleword in any manner you desire. The doubleword is transmitted within the external interrupt message header for both SEND and RESPONSE type external interrupts.

The user doubleword can be used for control information in a user-defined higher level protocol. That is, you could have your own message headers defined within the data transmitted from one virtual machine to another. The user doubleword could be used to control such a protocol.

The user doubleword can also be used as a security code or provide additional information for functions such as IDENTIFY and REJECT. You can specify a zero data length for any VMCF transaction. The effect of this is that only the external interrupt message header with user doubleword is transmitted or received.

DIAGNOSE Code X'68' Return Codes

The virtual machine initiating a VMCF request receives a return code in the general purpose register specified as "Ry" in the DIAGNOSE instruction. The return code indicates successful completion of the request or error conditions associated with the request. Figure 20 is a description of all possible return codes returned to a virtual machine executing DIAGNOSE code X'68'.

Return Code	Meaning
0	The normal response. Indicates successful completion of a request or successful initiation of a request. For example, for an AUTHORIZE request, 0 indicates that the AUTHORIZE function is complete; for a SEND request, 0 indicates that the SEND was successfully initiated. The SEND request, of course, would not be complete until the final RESPONSE external interrupt was received by the source virtual machine.
1	Invalid virtual buffer address or length. A virtual machine tried to execute a VMCF function but specified an invalid address or length: • External interrupt buffer not within virtual storage. • External interrupt buffer address not doubleword aligned. • Message data or buffer not within virtual storage. • External interrupt buffer less than the standard message header length.
2	Invalid function code. A virtual machine tried to execute a VMCF function but specified an unsupported subcode.

Figure 20 (Part 1 of 3). DIAGNOSE Code X'68' Return Codes

Return Code	Meaning
3	Protocol violation. A virtual machine tried to execute a function which would violate the defined protocol: • Cancel a message it did not originate. • Reply to a message not sent via SEND/RECV. • Executed more than one RECEIVE to a SEND or SEND/RECV request.
4	Source virtual machine not authorized. A virtual machine tried to execute a function (other than AUTHORIZE) but was not authorized to use VMCF (had not successfully executed the AUTHORIZE function).
5	User not available. A virtual machine tried to execute a function and specified a virtual machine currently not available for VMCF communication: • Not logged on. • Not authorized for VMCF communication. • Virtual machine authorized SPECIFIC for some other virtual machine.
6	Protection violation. A virtual machine tried to execute a VMCF function that would result in a STORE or FETCH protection violation. The virtual machine specified a data or buffer address that contained a storage key other than its current PSW key (assume the key was nonzero). This return code is also set if a virtual machine tries to receive data in a CP-owned shared segment.
7	SENDX data too large. A virtual machine tried to execute a SENDX request but specified a SENDX data length larger than the sink virtual machine external interrupt buffer.
8	Duplicate message. A virtual machine tried to execute a SEND-type function and specified a message ID and virtual machine userid for which there was already an active message.
9	Target virtual machine in QUIESCE status. A virtual machine tried to execute a SEND-type function and specified a sink virtual machine userid of a virtual machine in QUIESCE status.
10	Message limit exceeded. A virtual machine tried to execute a SEND function but already had 50 messages active. The virtual machine should clear any pending RESPONSE external interrupts or CANCEL previously sent messages to continue processing.
11	REPLY canceled. The source virtual machine executed a CANCEL to a previous SEND/RECV request. The sink virtual machine had already RECEIVED the message but had not yet executed a REPLY. The sink virtual machine REPLY in this case is canceled. The sink virtual machine receives return code 12 (message not found) when it executes the REPLY function.
12	Message not found. A virtual machine tried to execute a function and specified a message ID and virtual machine userid for a message that does not exist. The message may have existed at one time but could have been cancelled by the originator.
13	Synchronization error. The sink virtual machine tried to respond to a message for which it had not yet received the SEND external interrupt. This condition can occur if the sink virtual machine is anticipating certain messages but does not wait for the SEND external interrupt.

Figure 20 (Part 2 of 3). DIAGNOSE Code X'68' Return Codes

Return Code	Meaning
14	CANCEL too late. A virtual machine tried to CANCEL a message that had already been processed. The sink virtual machine had already responded with RECEIVE or REJECT (SEND request) or REPLY or REJECT (SEND/RECV request). This return code is also set if a virtual machine tries to CANCEL a SENDX request for which the sink virtual machine had already received the SEND external interrupt.
15	Paging I/O error. A virtual machine tried to execute a function which resulted in an uncorrectable paging I/O error. This is a hardware failure.
16	Incorrect length. A virtual machine executed a RECEIVE or REPLY function and specified a RECEIVE buffer length less than the source virtual machine SEND data length or a REPLY data length larger than the source virtual machine REPLY buffer length. The source virtual machine receives a data transfer return code identifying the condition.
17	Destructive overlap. A virtual machine executed a RECEIVE or REPLY function and specified a RECEIVE buffer address which overlapped the source virtual machine SEND data address or a REPLY data address that overlapped the source virtual machine REPLY buffer address. This condition can occur only when a virtual machine is sending messages to itself (a "wrap connection").
18	User not authorized for PRIORITY messages. A virtual machine tried to send a PRIORITY message to a virtual machine that was not authorized to accept PRIORITY messages (that is, had not executed the AUTHORIZE function with the PRIORITY option).
19	Data transfer error. A virtual machine executed a request that resulted in a data transfer error condition associated with the other virtual machine. The return code is returned to the sink virtual machine to indicate that the transaction did not complete successfully.
20	CANCEL - busy. A virtual machine tried to cancel a message being processed. If this is a SEND/RECV request and the RECEIVE function is in process, repeated retries may cancel the REPLY function.

Figure 20 (Part 3 of 3). DIAGNOSE Code X'68' Return Codes

Data Transfer Error Codes

When a virtual machine executes a SEND, SENDX, or SEND/RECV function, the normal DIAGNOSE return code is zero, indicating that the request was successfully initiated. However, when the actual data transfer takes place, errors can occur. All errors occurring at data transfer time are communicated to the source virtual machine in the RESPONSE external interrupt message header, VMCMHDR. Figure 21 shows error codes indicating conditions that are possible after the SENDX, SEND, or SEND/RECV request is initiated. The error codes correspond to DIAGNOSE return code numbers.

Error Code	Meaning
0	The normal response (no errors).
1	Invalid buffer address or length. The SEND and/or RECEIVE buffers used for a data transfer operation are not within the virtual machine's virtual storage. The beginning and ending addresses were valid when a request was initiated but all addresses are not valid.
5	User not available. The sink virtual machine executed the UNAUTHORIZE function, executed the AUTHORIZE SPECIFIC function again, or implicitly reset his virtual machine after the source virtual machine request was initiated.
6	Protection violation. The storage key for a virtual machine's SEND or RECEIVE buffer did not match its PSW key at the time the transfer was initiated (assume the key was nonzero). This error code is also set if a virtual machine tries to RECEIVE data into a CP-owned shared segment.
7	SENDX data is too large. The sink virtual machine executed AUTHORIZE again and specified an external interrupt buffer size less than the buffer size at the time a SENDX function was executed. The SENDX data no longer fits in the sink virtual machine buffer.
15	Paging I/O error. An uncorrectable paging I/O error occurred during the data transfer operation trying to fetch a virtual machine SEND or RECEIVE buffer. This is a hardware failure.
16	Incorrect length. The sink virtual machine executed a RECEIVE function with a data length (VMCPLENA) smaller than the original SEND data length or a REPLY function with a REPLY data length larger than the source virtual machine REPLY buffer length.
17	Destructive overlap. A virtual machine was communicating with itself in a "wrap connection" and his SEND or RECEIVE buffers overlapped one another (intra-virtual machine communication).
19	Data transfer error. A data transfer error occurred which was associated with the other virtual machine. The transaction did not complete successfully.

Figure 21. DIAGNOSE Code X'68' Data Transfer Error Codes

Part 2. VM System Applications

Part 2 contains a chapter about the following VM system application:

- The Programmable Operator Facility
- Getting Languages on Your System

Chapter 15. The Programmable Operator Facility

Overview

The **Programmable Operator Facility** is designed to increase the efficiency of system operation and to allow remote operation of systems in a distributed data processing environment. It does this by intercepting all messages/requests directed to its virtual machine and by handling them according to preprogrammed actions. It determines whether a message is to be simply recorded for future reference, whether the message is to be acted upon, or whether the message is to be sent on to the operator to handle.

Note: The programmable operator should always run with American English as the system national language. Routing tables and messages to the programmable operator should always be in American English to ensure that the uppercasing and routing table comparisons are handled correctly.

The tasks that can be performed by the programmable operator facility include:

- Logging messages
- Suppressing message display and routing messages to a logical (real) operator
- Executing commands
- Responding with preprogrammed message responses.

Using the Programmable Operator Facility in Various System Environments

Use in a Single System

When the programmable operator facility is operational in a single-system environment, it can:

- Ease message traffic to the system operator, by:
  - Filtering (logging) non-essential, information-only messages
  - Routing messages (for example, I/O intervention requests) to someone else for specialized action.

- Increase productivity, by freeing the system operator from certain **routine** responses or tasks. Such responses (whether they consist of one or a series of commands, whether VM or guest operating system) may be preprogrammed to execute automatically upon receipt of a given message.

  Thus, only essential, non-routine messages (that is, those requiring the skill and experience of a system operator to handle) are sent on to the operator for response or action.

Use in Distributed VM Systems

The capabilities of the programmable operator, outlined above, also allow for the remote operation of systems in a distributed VM environment. When the programmable operator facility is operational in a distributed VM system, it can:

- Issue responses and perform tasks that do not require an on-site operator
- Filter (log) non-essential, information-only messages
- Route messages requiring on-site (that is, manual) intervention to someone, not necessarily an operator, at the distributed site for action
- Route messages that require the skill and experience of a system operator to handle to the operator at the host system. The operator at the host site can also send commands to the programmable operator facility to control its operation, as well as commands to execute on the distributed system to control the system itself.

By running the programmable operator facility on VM systems distributed at several different locations (network nodes), one operator at a host site can control a network of systems.

Use in a Mixed Environment

The programmable operator facility also provides for distributed data processing in an SNA environment with mixed VM, OS/VS, and VSE distributed systems and host systems. We will call this a "mixed environment". The Programmable Operator/NCCF Message Exchange (PMX) provides an interface with the Network Communications Control Facility (NCCF) or NetView so that an NCCF or NetView operator can operate a VM distributed system, whether the operator is on a VM, OS/VS, or VSE system.

Notes:

1. *The Group Control System (GCS) is a requirement to use the programmable operator in a mixed environment. The Programmable Operator/NCCF Message Exchange (PMX) uses facilities unique to GCS, and therefore cannot run on any other supervisor. For more information on GCS, see the VM/SP Group Control System Command and Macro*

Reference and the VM/SP Planning Guide and Reference or the VM/SP HPO Planning Guide and Reference.

2. *NCCF and NetView are network management VTAM applications. NCCF requires the Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM) Version 3 for VM. NetView requires the Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM) Version 3 Release 1 Modification Level 1 for VM. NCCF and NetView run on subsequent release of ACF/VTAM unless otherwise stated.*

3. **The programmable operator facility does not allow a user logged on to a VM virtual machine to issue NCCF or NetView commands.** *A VM user cannot try to control NCCF or NetView as an authorized operator. The system only lets authorized NCCF or NetView operators issue VM commands within the programmable operator virtual machine.*

The Logical Operator

Occasionally the programmable operator must send messages to another virtual machine. To ensure that the programmable operator will function properly, a user (other than the programmable operator virtual machine on the local system, in a distributed system, or in a mixed environment) is identified to the programmable operator to receive these messages. This user is called the **logical operator**, as opposed to the CP system operator. When the programmable operator starts (in the CP system operator virtual machine, for example), the logical operator receives an initiation message. The logical operator also receives error messages for severe errors, such as logging errors, and receives all messages routed to the logical operator explicitly or by default.

In a mixed environment, an NCCF or NetView operator can be assigned as the logical operator to control a VM distributed system. For more information, see "The NCCF or NetView Logical Operator" on page 333.

The default logical operator is specified in the LGLOPR statement of the routing table. However, other logical operators can be dynamically assigned, released, or replaced using the LGLOPR command of the programmable operator facility. Both methods of identifying the logical operator are described in detail later in this section.

Routing Table Information

The programmable operator routing table identifies the programmable operator facility environment, including the default logical operator's userid and nodeid. It also specifies the action to take for each message, and authorizes certain users to invoke specific programmable operator commands. For a complete description of the information contained in a routing table, see "The Routing Table" on page 336.

The routing table is a separate CMS file that must be tailored for a specific use. The first routing table to be used is specified when the programmable operator facility is invoked. If no routing table name is specified, the default filename "PROP" is used.

The installation may define multiple routing tables to cover varying situations. For example, multiple routing tables can be defined to cover shift changes. Only one routing table can be active at a time. The active routing table may be replaced by issuing the LOADTBL command. Any person authorized in the active routing table may issue this command.

Action Routines

Action routines are programs or EXECs that receive control in response to the match of a message and a routing table entry. They handle a particular type of message or command intercepted by the programmable operator facility. A set of action routines is provided with the programmable operator facility. These need no tailoring to provide you with the control and function needed to operate the programmable operator facility. You can extend the programmable operator facility by writing a new action routine and adding it to the appropriate routing table. Action routines can be EXECs or written in Basic Assembler Language. (Basic Assembler Language action routines must also be added to the PROPLIB LOADLIB. You can do this by invoking CMSGEND PROP.)

How it Works

The programmable operator facility runs in a CMS virtual machine. Although it can run in any virtual machine, because of its programmed capability to log, handle, or redirect messages, it is most commonly run in the CP system operator's virtual machine.

The programmable operator facility compares all messages directed to it against entries listed in a routing table (a CMS file). When a match occurs, the prescribed action is performed. Any messages requiring a real operator's response or action are sent on to the defined operator (system, network, NCCF, etc.) at another console, a "logical" operator console. If the logical operator is on a virtual machine in the same system, the programmable operator sends the messages with either the CP MESSAGE or CP MSGNOH command. If the logical operator is on a virtual machine in a different VM system (network node), a host system for example, it sends the messages via RSCS Networking.

Consider this example:
The SYSOPR macro in DMKSYS specifies the userid OPER1 for the CP system operator. Set up the programmable operator to run in the OPER1 virtual machine and establish another virtual machine with userid OPERX. In the routing table file(s), specify OPERX as the logical operator. Now any CP or user messages sent to the system operator virtual machine can be handled or filtered by the programmable operator or routed to userid OPERX.

Note: The logical operator cannot receive the messages when the logical operator virtual machine is not receiving, such as (but not limited to), being logged off, disconnected, has SET MSG OFF, or passed through to another system. If another user sends messages to the virtual machine running the programmable operator, the originator of the messages receives the message DMKMSG057. This happens because the TELL command, which sends the messages from the programmable operator virtual machine to the logical operator virtual machine, verifies the ability of the logical operator to receive messages.

If the logical operator is an NCCF or NetView operator, the programmable operator sends messages though the Programmable Operator/NCCF Message Exchange (PMX) portion of the programmable operator facility.

Flow of Operation

When the programmable operator facility is running in a virtual machine, CP intercepts all messages intended for that virtual machine console. CP then passes these messages to the programmable operator facility via IUCV[16]. The messages are logged in a CMS file. The programmable operator facility then uses the active routing table to analyze the message and determine if further action is needed. Based on the contents of the routing table (such as message texts, message types, and user authorizations), the message can be passed to some specified action routine for further action.

If the message is to be routed to the logical operator, and that person is on another virtual machine in the same physical machine, the programmable operator facility routes the message directly to the logical operator. To do this, it uses the CP MSGNOH or CP MESSAGE command depending on the classification of the programmable operator virtual machine. If the logical operator is on a different physical machine, the programmable operator facility prefaces the message with the appropriate tag information and sends the message to RSCS Networking using the CP SMSG command. If the logical operator is an NCCF or NetView operator, the programmable operator facility sends the message through the Programmable Operator/NCCF Message Exchange (PMX) which passes the message on to the NCCF or NetView logical operator.

The programmable operator facility usually operates in a disconnected virtual machine. If someone logs on to this disconnected virtual machine with the programmable operator facility running, no messages are displayed

---

16 **Warning**: The maximum number of outstanding messages which IUCV queues for presentation to the programmable operator facility is the IUCV maximum value of 255. A high volume of incoming messages while executing an action routine or an exit EXEC, which requires external interrupts to be disabled for a period of time, may exceed the message limit. If this happens, additional incoming messages are routed directly to the programmable operator virtual machine console or alternate console without notifying the programmable operator facility. These additional messages cannot be handled or logged by the programmable operator facility.

(unless the programmable operator facility is running in DEBUG mode). All messages are being intercepted or received by the programmable operator program from IUCV. If that person should enter a command, the programmable operator facility gets control and reads the command entered. Only two commands are accepted from this environment; the STOP command and the SET command. The programmable operator facility rejects any other commands. However, in some situations the redisplay of the entered command is CP console I/O and is presented to the programmable operator facility as a type-3 message. If the text of this command matches a record in the active routing table, the programmable operator facility may invoke an action routine.

If a CMS abend occurs while the programmable operator facility is executing, all files are closed and abend error messages are sent to the logical operator. A dump of the virtual machine storage is taken using the CP VMDUMP command and the last system or device that was IPLed is re-IPLed. If the abend occurs while an action routine is executing, abend error messages are sent to the logical operator and the requester (if any). Control is returned to the point in the programmable operator facility immediately following the action routine call.

Relationship to RSCS Networking

When the programmable operator facility is running in a VM network environment, it is a normal user of RSCS Networking. This means that the programmable operator facility communicates to RSCS using the CP SMSG command. Any configuration of systems and networks supported by RSCS Networking can use the programmable operator facility. The time needed for a message to go from the system at a distributed site to the logical operator at the host system, or vice versa, depends on the number and type of communications links between the message sources and destinations.

When the logical operator is an NCCF or NetView operator, the programmable operator does not use RSCS to route messages to the logical operator. It, instead, passes the messages to NCCF or NetView through the Message Queueing Service so that the messages are presented at the appropriate NCCF or NetView operator console.

A programmable operator can check on its ability to communicate with a host or distributed system. See "Communications Checking" on page 359. later in this section.

The Programmable Operator Virtual Machine

Installing the Programmable Operator Facility

The VM product contains the file PROPLIB LOADLIB which is the basis for the programmable operator facility. After receiving and installing VM, take the following steps before running the programmable operator facility.

1. Reserve enough minidisk space to contain the log file(s) and feedback file for the virtual machine that the programmable operator facility will be running in. The amount of space needed depends on the amount of message traffic that will be going through the programmable operator facility, and on the number of comments you expect users to place in the log and feedback files. To help you determine the amount of space needed for the log and feedback files, see "The Log File" on page 354 and "The Feedback File" on page 358.

2. The sample routing table is located on the CMS 190 minidisk. To use the sample PROP RTABLE, take the following steps:

   ```
   ACCESS 190 C/A
   COPYFILE PROP RTABLE C = = A
   ```

   This places the sample routing table on a read/write minidisk accessed by the virtual machine. Edit the sample routing table (PROP RTABLE) to include the functions and authorizations to meet the various needs of the installation For more information on tailoring the routing table see "The Routing Table" on page 336. Place the edited file on a minidisk accessed by the programmable operator facility virtual machine.

3. Optionally, if you have made any changes to the supplied action routines, link the TEXT file to the PROPLIB LOADLIB. The CMSGEND EXEC, using the CMSGEND PROP function, allows user-modified routines to be added or replaced in the PROPLIB LOADLIB.

   If you have written any additional action routines, use the CMS LKED command to add these routines to the PROPLIB LOADLIB. Copy the PROPLIB LOADLIB from the CMS system disk to a read/write disk because any changes would invalidate the directory entry on the system disk. For example, to link a user-written action routine named ACTIONA to the PROPLIB LOADLIB, you would issue:

   ```
   LKED ACTIONA (LET LIBE PROPLIB
   ```

   Action routines written in Basic Assembler Language must be put in the PROPLIB LOADLIB. EXEC action routines need not be put in the PROPLIB LOADLIB, but can reside on any minidisk accessible to the programmable operator.

If you are operating in a mixed environment and need the Programmable Operator/NCCF Message Exchange (PMX) to route messages to the NCCF or NetView logical operator, you must first install the PMX. For details on the installation procedure, see "Installing the PMX" on page 328.

Invoking the Programmable Operator Facility

Manual Invocation

Before loading and invoking the programmable operator facility, load CMS in the virtual machine that will be running the programmable operator facility.

Use the PROPST EXEC to invoke the programmable operator facility manually. The PROPST EXEC drops *any* IBM-supplied programmable operator routines that are currently loaded as a nucleus extension, and loads the programmable operator as a nucleus extension. It then invokes the programmable operator facility with the specified RTABLE. If you do not specify a routing table, the default RTABLE name is "PROP". You may specify a disconnect parameter to disconnect the programmable operator before it is invoked. PROPST EXEC is located on the CMS System Disk (190) and can be tailored to your needs. The format of the invocation EXEC is as follows:

PROPST	[*rtable-name* / **PROP**] [**DISConn**]

Alternatively, you can take the following steps before each time you invoke the programmable operator facility:

1. Issue a FILEDEF command to assign a CMS filename to the PROPLIB LOADLIB file so CMS can read and load from it. Do this with the following command:

   ```
   FILEDEF PROPLIB DISK PROPLIB LOADLIB *
   ```

2. Next, load the programmable operator program as a CMS nucleus extension via the NUCXLOAD command. Issue the command as follows:

   ```
   NUCXLOAD PROP DMSPOP PROPLIB
   ```

   (See the *VM/SP CMS Command Reference* for more details on the NUCXLOAD command.) These first two steps may be omitted for subsequent invocations as long as you do not:

   - IPL CMS or

- Have a CMS abend from which the programmable operator does not automatically recover.

3. Following its loading as a CMS nucleus extension, invoke the programmable operator facility as if it were a CMS command. The format of the invocation is:

```
PROP [ rtable-name ]
     [ PROP        ]
```

rtable-name is the filename of the routing table that is to be used for the programmable operator facility. "PROP" is the default filename of the routing table if no other is specified at invocation.

Automatic Invocation

If you wish, you can set up the programmable operator facility to start running when the system is IPLed and to restart automatically in the event of CP system restart. This can be done as follows:

1. Place an "IPL CMS PARM AUTOCR" entry for programmable operator's virtual machine in the CP directory. You can do this even if the programmable operator virtual machine is the CP system operator.

2. Place the following entry in the PROFILE EXEC of the programmable operator's virtual machine:

```
EXEC PROPST rtable-name [DISConn]
```

   You can precede or follow the invocation of the PROPST EXEC by issuing any virtual machine commands that you wish to have executed before or after the programmable operator facility is invoked. Virtual machine commands that are placed after the invocation of the PROPST EXEC are not executed until the programmable operator facility is stopped.

3. If you want the programmable operator to run in other than the operator's virtual machine, place an AUTOLOG entry for the programmable operator's virtual machine in the PROFILE EXEC of the system operator or the AUTOLOG1 user.

4. Once this is complete, if the logical operator is not already logged on, he should do so on the appropriate system.

The following example shows how to place entries in the CP Directory and the PROFILE EXEC of operator's virtual machine. These entries automatically invoke the programmable operator facility in the operator's virtual machine when the system is IPLed. The userid of the programmable operator virtual machine is "OPERATOR". The default, "PROP RTABLE", is the name of the routing table being used.

CP directory entries:

```
   :
   :
USER OPERATOR password 1M 2M ABCDEFG
IPL CMS PARM AUTOCR
   :
   :
```

Entries in the PROFILE EXEC of the Operator's virtual machine:

```
   :
   :
EXEC PROPST DISCONN
   :
   :
```

Once these changes (or similar ones) have been made, IPLing the system causes the programmable operator to be invoked automatically in the disconnected system operator virtual machine. After the DISCONNECTED message has been written to the console, indicating that the system operator virtual machine is disconnected, the operator can log on to whatever virtual machine is normally used, for example, the default logical operator virtual machine specified in the routing table.

Initialization

The programmable operator facility is initiated by IPLing a CMS virtual machine of at least 1Meg in size and invoking the programmable operator facility. When the programmable operator facility gets control, it locates the specified or default routing table and loads it into virtual storage. The action routines named in this routing table are in turn loaded as CMS nucleus extensions. For each action routine specified in the routing table, an EXEC file or a corresponding member in a CMS simulated OS load library named PROPLIB LOADLIB must exist. If an EXEC does not exist, the LOADLIB member is loaded as a nucleus extension via the NUCXLOAD command. If both exist, the EXEC takes precedence.

If upon invocation, the programmable operator facility cannot find an action routine named in the routing table, an error message is issued, and, after displaying all detectable routing table errors, the programmable operator facility terminates operation. Otherwise, the programmable operator facility is fully initialized, and writes a message to the programmable operator's console, to the logical operator, and to the LOG file, indicating that the programmable operator facility has started.

The programmable operator facility then waits for either an incoming message or a programmable operator console command (STOP and SET are the only valid commands). The operator can disconnect at this point by having specified the DISConn parameter for the PROPST EXEC, by entering CP (pressing the PA1 key or equivalent) and typing CP DISCONN or by issuing #CP DISCONN where "#" is the logical line end character. After the DISCONNECTED message has been written to the console, indicating that the system operator virtual machine is disconnected, the operator can log on to whatever virtual machine is normally used, for example, the default logical operator virtual machine specified in the routing table.

When the programmable operator starts, it executes an EXEC called PROPPROF, which is located on the CMS System Disk (190). This EXEC can be copied to your A-disk and tailored. PROPPROF is called after the programmable operator initialization is complete, that is, after the programmable operator message environment has been set up. PROPPROF tailors the programmable operator message environment to your specific needs. For example, to prevent the programmable operator from trapping warning messages, PROPPROF could look like this:

```
/*  PROPPROF Example  */
"CP SET WNG ON"
Exit
```

PROPPROF can be used for any special programmable operator start processing desired without having to rewrite the PROPST EXEC or create a new EXEC to invoke the programmable operator facility. After executing PROPPROF, the programmable operator then begins routing messages as defined by the routing table entries and the LGLOPR statement in the specified routing table.

When the programmable operator facility is automatically restarted following an abend, it, if possible, uses the routing table in effect at the time of the abend rather than that specified at invocation. Also, if a logical operator (other than the default logical operator) had been assigned, that assignment is restored.

Note: If the user enters a nodeid into the SYSTEM NETID file that is invalid as a CMS filetype, the programmable operator cannot start because it is not able to open the log file.

How the Programmable Operator Establishes Communications with IUCV

The programmable operator facility automatically establishes communications with CP through the Inter-User Communications Vehicle (IUCV). When the programmable operator facility is initialized, a CMSIUCV CONNECT, specifying *MSG and an application id of PROP, is issued to establish the communications path with the Message System Service (See the "Message System Service" section earlier in this manual). This allows the programmable operator program to read and evaluate messages directly from CP.

Several CP command settings determine the types of messages that the programmable operator facility can receive. The programmable operator facility issues these SET commands when initializing, and resets them when terminating. The commands issued during initialization are:

```
SET MSG IUCV
SET WNG IUCV
SET SMSG IUCV
SET EMSG IUCV
SET IMSG IUCV
SET CPCONIO IUCV
SET VMCONIO OFF
```

VMCONIO is set OFF so that any messages produced by CMS or the programmable operator during initialization of the programmable operator facility are typed on its virtual machine console. If VMCONIO was set to IUCV, such data would be trapped by IUCV and not displayed.

When the programmable operator STOP command is issued, the following SET commands are issued:

```
SET MSG ON
SET WNG ON
SET SMSG OFF
SET EMSG ON
SET IMSG ON
SET CPCONIO OFF
SET VMCONIO OFF
```

Then, after the existing messages are handled, the IUCV connection is severed using the IUCV SEVER function.

Some other virtual machine settings that the programmable operator facility modifies are "SET RUN ON", "SET TIMER REAL", and "TERMINAL MODE VM" at initialization, and "SET RUN OFF" and "SET TIMER ON" at termination. "SET RUN ON" is issued to ensure that the programmable operator is not held up in CP console function mode for excessive periods of time, either because of some operator command entry or because of logging on to a disconnected programmable operator virtual machine. "TERMINAL MODE VM" is to ensure that programmable operator console commands are handled correctly.

Notes:

1. *Single Console Image Facility (SCIF) operation supersedes IUCV Message System Service operation. If the programmable operator virtual machine has a SCIF secondary user, messages would be sent via SCIF to the secondary user rather than handled by the programmable operator virtual machine through the IUCV Message System Service. However, the programmable operator facility may be a SCIF secondary user for another virtual machine. For example, this can be used to control the operation of a guest operating system running in another virtual machine. In this case, SCIF messages are presented to the programmable operator virtual machine as IUCV message-type 8.*

2. *If you have installed VM using the Starter System but you plan to run the programmable operator facility in the operator's virtual machine, you must remove the secondary user from the operator virtual machine directory entry. Otherwise, the Single Console Image Facility (SCIF) overrides the programmable operator facility.*

Message Output Format

The messages and responses from the programmable operator facility are sent to local VM users via the CP MSGNOH command if the programmable operator virtual machine has user class B authorization. Otherwise, the CP MESSAGE command is used. Regardless of which message command is used the messages from another user that are routed to the logical operator are prefixed with the userid and nodeid of the originating user.

The format of these messages appears as follows:

```
col 1     col 10     col 20
|         |          |
V         V          V
userid    nodeid:    text
```

Messages that the programmable operator facility sends as responses to the issuer of a programmable operator command or an asynchronous message to the CP operator originating at the programmable operator virtual machine have no such prefix. ***These responses are displayed in the language that is set for the programmable operator virtual machine. This language should always be American English.***

Stopping the Programmable Operator Facility

The programmable operator facility is easily stopped with the programmable operator STOP command. This command can be issued by a user logged on to the programmable operator facility virtual machine by typing STOP and pressing the ENTER key (or its equivalent). The programmable operator facility completes processing of all pending messages before stopping and returning control to CMS.

An alternative way to stop the programmable operator facility is for an authorized user to pass the STOP command to the programmable operator facility virtual machine from another virtual machine. The facility will stop processing as described above and return control to CMS. If the programmable operator facility virtual machine was running disconnected, it is left disconnected and the programmable operator facility is not active. To restart the programmable operator facility, someone must log on to the programmable operator facility virtual machine and invoke the facility as described in "Invoking the Programmable Operator Facility" on page 322.

PROPEPIF EXEC: Before the programmable operator facility terminates completely, it executes an EXEC called PROPEPIF, which is located on the CMS System Disk (190). This EXEC can be copied to your A-disk and tailored. PROPEPIF lets you restore any virtual machine settings that the programmable operator may have modified during operation. For example, when the programmable operator stops, it issues the command SET EMSG ON. If you want the final setting of EMSG to be something other than ON, simply put a command in the PROPEPIF EXEC, such as:

```
/*  PROPEPIF Example  */
"CP SET EMSG TEXT"
Exit
```

PROPEPIF, along with PROPPROF, simply gives you more control over the changes that the programmable operator makes to a virtual machine while operating.

Effect on an NCCF or NetView Logical Operator: If you use the STOP command to stop the programmable operator facility, the system sends a message to you (the NCCF or NetView logical operator). Someone must then logon to the programmable operator virtual machine and restart the programmable operator facility to continue operating in the mixed environment. If the programmable operator stops because of an abend and is able to recover, the programmable operator facility tries to re-establish the IUCV connection between the programmable operator and the PMX. In either case (STOP or ABEND), the programmable operator informs any assigned (using the LGLOPR ASN command) NCCF or NetView logical operator of its status (stopped or abended) via the PMX.

Running the Programmable Operator Facility from NCCF or NetView

To enhance the use of the programmable operator, it can be controlled by an NCCF or NetView operator, thus giving this operator control over mixed VM, OS/VS, and VSE distributed systems and host systems.

The Programmable Operator/NCCF Message Exchange

The Programmable Operator/NCCF Message Exchange (PMX) serves as a pipeline to transfer programmable operator commands from NCCF or NetView to the programmable operator virtual machine and to transfer routed messages and programmable operator responses to NCCF or NetView. The PMX executes in a GCS virtual machine with NCCF or NetView thus making GCS a requirement for programmable operator and NCCF or NetView communication. PMX uses IUCV to communicate with the programmable operator facility. The PMX also uses an NCCF or NetView command processor and the DSIMQS macro to communicate with NCCF or NetView logical operators. See *NCCF Customization*, Version 2, Release 1, SC27-0662 or *NetView Customization*, LY30-5586.

Installing the PMX

To use the Programmable Operator/NCCF or NetView connection, the user must:

1. Authorize the programmable operator virtual machine and the GCS virtual machine in which the PMX will execute to obtain IUCV connections with each other. The maximum number of IUCV connections allowed for the programmable operator must be at least three. For the PMX it must be at least two.

   Make this authorization by adding IUCV statements for each of these users to the CP directory. For more information on the CP directory,

please refer to the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference.*

2. Set up the GCS virtual machine in which the PMX will run as an unauthorized GCS virtual machine. That is, the PMX (along with NCCF or NetView) should run in problem state rather than supervisor state.

3. Specify in the GCS PROFILE (filename filetype, PROFILE GCS) that the PMX should start when GCS is IPLed. The format of the GCS command provided by NCCF for invoking the PMX is:

   ```
   NCCF START PMX PARM userid
   ```

   where 'userid' is the userid of the programmable operator virtual machine. Do this by modifying the EXEC NCCFSTRT GCS that is provided with NCCF.

   The format of the GCS command provided by NetView for invoking the PMX is:

   ```
   NETVIEW START PMX PARM userid (LOCAL)
   ```

   where 'userid' is the userid of the programmable operator virtual machine. Do this by modifying the EXEC NETSTRT GCS that is provided with NetView.

   If you do not specify this in the GCS PROFILE or the EXECs mentioned above, then the PMX must start manually after the programmable operator is started by issuing the same GCS command. When started, the PMX ATTACHes NCCF or NetView; that is, it runs as a subtask of the PMX in the same GCS virtual machine. When you specify the 'LOCAL' option, the PMX attaches the hardware component of NetView which captures LOCAL device errors.

   Note: For NCCF, see *NCCF Version 2 for VM/SP Installation and Resource Definition*, SC30-3264. For NetView, see *NetView Installation and Administration*, SC30-3360.

4. Run the VMFLKED EXEC with input file PROPMX LKEDCTRL to build the PROPMX LOADLIB from which the PMX module is loaded by the NCCF or NetView command. The PROPMX LOADLIB must then be placed on a minidisk which is accessed by the virtual machine running PMX and NCCF or NetView. For example, it could be placed on the same minidisk as the NCCF LOADLIB. The PROPMX LOADLIB must also be "GLOBALed" (that is, GLOBAL LOADLIB PROPMX), along with the NCCF LOADLIB prior to invoking the NCCF or NetView command. This GLOBAL command can also be issued in the NCCFSTRT or NETSTRT GCS EXEC. The commentary in the EXEC describes how to do this.

5. Ensure that entries are added to the appropriate routing table files to authorize one or more NCCF or NetView operators to be assigned as

the logical operator; that is, authorize them to issue the programmable operator LGLOPR command. For more uses of the LGLOPR command, see "Assigning or Changing the Logical Operator" on page 334.

```
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* AUTHORIZE NCCF OR NETVIEW OPERATORS TO CHANGE LGLOPR ASSIGNMENT
*------------------------ --- --- -- -------- -------- -------- --------
/LGLOPR /                   1   7 30          *NCCF    DMSPOR   LGLOPR
*------------------------ --- --- -- -------- -------- -------- --------
```

Figure 22. LGLOPR Command Authorization for an NCCF or NetView Operator

Figure 22 would authorize any NCCF or NetView operator to issue the LGLOPR command. Restricted or specific authorization would be provided by putting the desired NCCF or NetView operator id in the user field of the RTABLE entry.

6. A PROP command name must be supplied to NCCF or NetView through the CMDMDL statement in the file DSICMD NCCFLST, such as,

   ```
   PROP      CMDMDL    MOD=CSIPNP
   ```

   For more information on NCCF, NetView, and GCS, please read the appropriate documentation for each product.

Communication Between the Programmable Operator and NCCF or NetView

An NCCF or NetView operator (not necessarily the logical operator) and the programmable operator interact through both the NCCF or NetView PROP command and the Programmable Operator/NCCF Message Exchange (PMX). The PROP command is the NCCF or NetView operator's only means of sending data (commands) to the programmable operator. The PMX mediates all messages intended for NCCF or NetView logical operators.

An NCCF or NetView operator issues a programmable operator command by using PROP, an NCCF or NetView command, like this:

```
PROP CMD DETACH 00A FROM USER1
```

In a different domain, the operator must first establish an NCCF or NetView cross-domain session with the NCCF or NetView START command. Then, any programmable operator commands to be issued in that domain must be encapsulated in an NCCF or NetView ROUTE command. For example,

```
START   DOMAIN=DOMN2
   :
   :
   :
ROUTE   DOMN2,PROP CMD SET LOGMSG 1 SHUTDOWN IS AT 7PM TONIGHT
```

Refer to NCCF or NetView documentation for further information on domains and cross-domain sessions.

PMX Communication Protocol

This section describes the protocol to be used by the programmable operator facility and PMX application programs in establishing and breaking the IUCV communications path between them. Also described are other actions relating to this communication protocol.

Starting up:

Programmable operator: If PMX and the programmable operator have previously been active, CONNECT to the PMX using the previously saved userid. If a non-zero return code is received, reset any saved information relating to the PMX. If the PMX was not previously active then the programmable operator waits for a CONNECT PENDING interrupt from the PMX. The programmable operator ACCEPTs only the first PMX CONNECT.

PMX: CONNECT to the programmable operator (having obtained the userid from the input parameters). If necessary, retry 10 times with a 15-second wait preceding each retry.

Stopping the programmable operator:

Programmable operator: Perform a CMSIUCV SEVER, in addition to the normal STOP processing.

PMX: Reset the saved LGLOPR identification and wait for a CONNECT PENDING from the programmable operator.

A programmable operator abend occurs:

Programmable operator: For a programmable operator mainline abend or a CMS system abend, CP does an implicit SEVER when the virtual machine is re-IPLed.

PMX: Notify the NCCF or NetView logical operator, if any, and wait for a CONNECT PENDING from the programmable operator.

A PMX abend occurs:

Programmable operator: A SEVER is received from the PMX. Reset any saved information relating to the PMX and if the current logical operator is an NCCF or NetView operator, enqueue a "LGLOPR RLS" command for that operator at the top of the

programmable operator message queue. The programmable operator then continues normal operation and waits for a CONNECT PENDING interrupt from the PMX. When it is received, the old PMX path is SEVERed and the new one ACCEPTed.

PMX: Notify the NCCF or NetView logical operator, if any, perform a SEVER and wait for NCCF or NetView to terminate. (Note: NCCF or NetView will have to be CLOSEd to restart the PMX.)

Stopping the PMX

If PMX stops or the NCCF or NetView operator session that is controlling the programmable operator stops, the NCCF or NetView logical operator (if any) is implicitly released. The programmable operator continues with the default logical operator specified on the LGLOPR statement of the active routing table.

The PMX may stop because:

- VM GCS stops
- NCCF or NetView stops (or the hardware monitor component of NetView, if the 'LOCAL' option of the GCS NETVIEW command was used when starting the PMX.)
- The PMX abends.

There are no PMX commands, per se; stopping NCCF or NetView normally stops the PMX.

When the PMX stops, NCCF or NetView does not necessarily stop -- the PMX tries to wait until NCCF or NetView has stopped. However, you must stop NCCF or NetView before restarting the PMX.

The Logical Operator

The programmable operator facility is most commonly run in the CP system operator's virtual machine. It intercepts messages to the system operator, logs them in a CMS file and then performs additional actions on the messages, as defined by the active routing table. Sometimes, it is necessary for the programmable operator to send an intercepted message to another user for handling, because the programmable operator does not know or is not capable of performing the required action. (One example would be a message requesting that a tape be mounted. Since the programmable operator cannot perform physical tasks, such as mounting tapes, it simply sends the message to another user for handling.) For this reason, a user, other than the programmable operator virtual machine, is identified to the programmable operator to receive these messages. This user is called the logical operator, as opposed to the CP system operator.

When the programmable operator starts, the logical operator is sent an initiation message. While the programmable operator is active, the logical operator receives error messages for severe errors, such as logging errors, as well as all messages routed to the logical operator, either explicitly or by default. The logical operator also receives a message when the programmable operator is stopped.

The programmable operator facility can have only one logical operator at any given time. However, the role of the logical operator can be passed dynamically between several VM users and/or NCCF or NetView operators by means of the programmable operator LGLOPR command. Refer to the "LGLOPR Command" on page 373 and "Assigning or Changing the Logical Operator" on page 334 for more information.

The Default Logical Operator

Although the role of the logical operator can be passed from one VM or NCCF (or NetView) user to another, a single VM user must still be identified to the programmable operator. This user is called the DEFAULT logical operator and is identified to the programmable operator facility by means of the LGLOPR statement of the active routing table. The programmable operator facility begins routing messages to this DEFAULT logical operator when no other logical operator has been assigned or when the current logical operator issues the LGLOPR RLS command, implicitly or explicitly.

During programmable operator initialization, a message is sent to the logical operator indicating that the programmable operator facility is running. Normally (i.e. unless the programmable operator is restarting after an ABEND situation), this logical operator is the DEFAULT logical operator specified in the active routing table, since the programmable operator is just starting up and no other users have been assigned yet. The DEFAULT logical operator will continue to be the current logical operator until an authorized VM or NCCF (or NetView) user issues the programmable operator LGLOPR command. Thus, if the active routing table does not authorize ANY users to issue the LGLOPR command, the DEFAULT logical operator will ALWAYS be the logical operator, until a new routing table is loaded.

Again, the DEFAULT logical operator MUST be a local or remote VM user; that is, the DEFAULT logical operator CANNOT be an NCCF or NetView operator.

Note: To avoid ambiguity, it should be noted that the term 'logical operator' throughout this section is actually referring to the currently assigned logical operator, which MAY also be the DEFAULT logical operator. References to the 'DEFAULT logical operator' are made when the information being discussed pertains ONLY to the DEFAULT logical operator.

The NCCF or NetView Logical Operator

To run a VM system from NCCF or NetView an NCCF or NetView operator must be assigned as the logical operator of the programmable operator in that particular system. This operator is the **NCCF logical operator**. or the **NetView logical operator**. The routing table should specify which NCCF or NetView operators may be assigned as logical operators. The network installation through NCCF or NetView and VTAM defines where such NCCF or NetView operators may be logged on. This is independent of the programmable operator facility.

An NCCF or NetView operator may be authorized to be the logical operator by allowing him to issue the programmable operator LGLOPR command. An NCCF or NetView operator can also be authorized to issue other programmable operator commands (QUERY, SET, CMD, etc.) without being authorized to use the LGLOPR command. That is, an NCCF or NetView operator can issue programmable operator commands without being authorized to be a logical operator. For example, assume the following record was in the active routing table.

```
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
/QUERY /                    1   6 30 NTWKOP1  *NCCF    DMSPOR   QUERY
*------------------------ --- --- -- -------- -------- -------- --------
```

Figure 23. QUERY Command Authorization for an NCCF or NetView Operator

The entry above allows the NCCF or NetView operator, NTWKOP1, to issue the programmable operator QUERY command and receive the responses, even if he is not authorized to use the programmable operator LGLOPR command to assign himself as logical operator.

Assigning or Changing the Logical Operator

The LGLOPR command provides three options for assigning and releasing logical operators. These three options allow the role of logical operator to be passed back and forth between VM users and/or NCCF or NetView operators. For the format of this command, see "LGLOPR Command" on page 373. To authorize operators to use the LGLOPR command, place an entry for "/LGLOPR /" in the routing table to be used. Specify DMSPOR as the action routine, and LGLOPR as the parameter to DMSPOR.

Using the ASN (assign) and RLS (release) options is sufficient for many installations where perhaps there is only one VM operator or key operator (specified as the default logical operator) and one NCCF or NetView operator.

When more than one NCCF or NetView operator is capable of controlling the system or when a combination of multiple VM and NCCF or NetView logical operators are capable, the installation might want to allow some or all of the logical operators to use the RPL (replace) option. RPL forces the issuer of the command to be assigned as the logical operator, whether or not a logical operator is already assigned. This ensures that no messages are lost while changing logical operators because no messages are handled during the change.

For example, in establishment X, two NCCF or NetView operators on an MVS host system share responsibility for control of a set of VM distributed systems. The role of logical operator is passed back and forth between them depending on their individual workload. These operators pass control using the RPL option of the LGLOPR command so that no messages are routed to the default logical operator between releasing the old logical operator and assigning the new one.

The RPL option is also useful for forcing the assignment of a logical operator when the current logical operator is for some reason cut off from communication with the programmable operator and cannot do a RLS (release).

For example, a communications line connecting the terminal for an NCCF or NetView logical operator to the NCCF or NetView subsystem has gone down. The NCCF or NetView logical operator informs a local VM operator by telephone that the VM operator will have to take over as logical operator until the communications line can be brought back up. The VM operator, if authorized, can issue a LGLOPR RPL command to take over the role of logical operator.

When the logical operator is re-assigned, the programmable operator tries the HOSTCHK function, if specified, following the re-assignment. However, if the new logical operator is an NCCF or NetView operator or a VM user on the same system as the programmable operator, the HOSTCHK function is suspended until a remote VM user is again assigned as the logical operator. For information on the HOSTCHK function, see "Communications Checking" on page 359 and the HOSTCHK statement in "Routing Table Entry Formats" on page 337.

If a user is assigned as the logical operator by the LGLOPR command, the assignment is accepted. The user specified on the LGLOPR statement remains a default logical operator in the event that the assigned logical operator is released. The user specified on the LGLOPR statement is always maintained as the default, and can never be released.

Here are some sample uses of the LGLOPR command. In the samples, ACTIONX is an action routine that is executed each time an unauthorized user issues some form of the LGLOPR command. ACTIONX could be DMSPOS sending the offending command to the logical operator or it could be a user-written routine taking some other action against the issuer.

```
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* GENERAL AUTHORIZATION FOR LGLOPR COMMAND -- ASN, RLS, AND RPL
*------------------------ --- --- -- -------- -------- -------- --------
/LGLOPR /                   1   7    NTWKOP1  *NCCF    DMSPOR   LGLOPR
*------------------------ --- --- -- -------- -------- -------- --------
* EXPLICIT AUTHORIZATION FOR LGLOPR REPLACE
*------------------------ --- --- -- -------- -------- -------- --------
/LGLOPR /RPL/               1  16    OPER2    HOSTSYS  DMSPOR   LGLOPR
/LGLOPR /RPL/                                             ACTIONX
*------------------------ --- --- -- -------- -------- -------- --------
* EXPLICIT AUTHORIZATION FOR LGLOPR ASSIGN
*------------------------ --- --- -- -------- -------- -------- --------
/LGLOPR /ASN/               1  16    NTWKOP4  *NCCF    DMSPOR   LGLOPR
/LGLOPR /ASN/                                             ACTIONX

An entry or entries could also be added as follows:

*------------------------ --- --- -- -------- -------- -------- --------
* AUTHORIZATION FOR LGLOPR ASSIGN AND RELEASE
*------------------------ --- --- -- -------- -------- -------- --------
/LGLOPR /¬RPL/               1       NTWKOP5  *NCCF    DMSPOR   LGLOPR
```

Figure 24. Sample LGLOPR Command Entries in a Routing Table

The Routing Table

The **routing table** is a CMS file that contains the information used to control the operation of the programmable operator facility. The routing table enables the programmable operator facility to recognize a message as a command, to determine the action to take when a message comes in, and to recognize the authorized users of programmable operator functions.

Note: The routing table should always be coded in American English. Messages to the programmable operator should also always be in American English to ensure that the uppercasing and routing table comparisons are handled correctly.

How the Programmable Operator Facility Uses the Routing Table

When the programmable operator facility receives an IUCV interrupt with an incoming message, the active routing table is searched to find a matching entry. When the routing table is searched, all fields are checked. For a match to occur, each field must either match or be blank. If a matching entry is found, that entry contains information pertaining to any action to be taken. The action routine name tells the programmable operator facility which action routine to invoke when a routing table entry matches the incoming message. If no matching entry is found in the active routing table, no action is taken besides logging the message.

The order that the entries are placed in the routing table affects the way the programmable operator facility performs. The routing table is searched from top to bottom until a match is found. As the table is searched, lines that begin with an asterisk (*) in column 1 are ignored, and therefore may be used to place comments in the routing table. Also, lines that are completely blank are ignored in the routing table search and can be used to separate lines of text for easier reading. All entries must be made in upper case.

Note: The routing table format is different from the format in the initial version of the programmable operator facility in Release 2 of VM. The original format from Release 2 is not compatible with later versions of the programmable operator facility. The routing tables must be converted to reflect this change. See Appendix C, "Converting Programmable Operator Routing Tables" on page 429 if your routing tables apply to Release 2.

Routing Table Entry Formats

Every routing table must have specific configuration information in the first records of the routing table file (filetype RTABLE) that are not comments or blank lines. These statements are in free format, meaning that they need not be positioned in any particular columns. See Figure 26 on page 345 for an example of a partial routing table. The statements and their parameters are as follows:

1. The **LGLOPR** statement identifies the default logical operator. The logical operator assignment can be changed using the LGLOPR command, but if a logical operator is released but not replaced the user specified in this statement resumes logical operator responsibilities. This userid or nickname **must be** on a VM system.

LGLOPR	{ *userid* [*nodeid*] } { *nickname* }

userid is a valid userid on the specified VM node.

nodeid is a valid id of a VM system in the network. If no nodeid is specified, the local system's nodeid is used.

nickname is a nickname defined in the programmable operator facility virtual machine CMS NAMES file.

Note: If a nickname is used to identify the logical operator, the nickname cannot be a list of nicknames. The programmable operator must have one nodeid to associate with the logical operator.

Either a nickname or a userid must be specified. If a userid is specified, a nodeid may be specified. If both a userid and a nodeid are specified, they must be separated by one or more blanks. If the name specified is both a local userid and a nickname, the programmable operator regards it as a nickname.

IMPORTANT NOTE: The programmable operator virtual machine should not be identified as the logical operator. This causes the programmable operator to go into a loop in the event it tries to do the routing. This includes specifying a userid of OPERATOR (or any abbreviation thereof) in the LGLOPR statement. This also causes the message to be sent to the system operator virtual machine, even if the system operator virtual machine has a different userid.

2. The optional **TEXTSYM** statement specifies the characters that the programmable operator facility interprets as special symbols in the text field of the routing table entries. All three parameters must be specified if the statement is specified.

TEXTSYM	*blank-sep arbchar-sep not-symbol*

blank-sep is a separator character indicating that **blanks** are to be skipped over when scanning the message. A message is scanned for the next non-blank character string. This non-blank character string is then compared to the text in the routing table entry following this separator character. The default character is "/".

arbchar-sep is a separator character indicating that **all non-matching characters** are to be skipped over when scanning the message. A message is scanned for the text specified in the routing table entry until it is found or until the end of the message is reached. The default character is "$".

not-symbol when it immediately follows a separator is the character indicating that the text should not be found in the message. If the text following the not-symbol is found in the message, then the message does not match that routing table entry. The default character is "¬".

See "Filtering Messages" on page 350 for the use of TEXTSYM characters in routing table entries.

3. The optional **PROPCHK** statement identifies the distributed nodes that the host system is to check on. The RSCS nodeids of these distributed systems must be specified in this statement. A programmable operator must be running in the system operator virtual machine on the distributed systems being checked. Specify as many nodes on one statement as fit in an 80-column record. The programmable operator facility only reads the first 80 columns. Enter any number of PROPCHK statements to specify different checking or response wait intervals for different RSCS nodes. The PROPCHK statement must be after the LGLOPR statement.

   Note: Nodes to be checked with PROPCHK must be systems running VM Release 3 or above, with a programmable operator in the system operator virtual machine.

PROPCHK	*ccc ww nodeid [nodeid ...]*

   ccc is the checking interval. This interval, in minutes, indicates how often acknowledgment requests are sent out to the specified nodes.

   ww is the response wait interval. This interval is the number of minutes permitted to pass before a response must be received from the specified node(s).

   nodeid is a valid id of a system in the network.

   Notes:

   a. *The checking interval specified must be greater than the response wait interval.*

   b. *The nodeid of the logical operator must not be specified as a nodeid on this statement.*

4. The optional **HOSTCHK** statement specifies the time interval for checking communication with the RSCS virtual machine at the logical operator node and the wait time for a response. The HOSTCHK statement must be after the LGLOPR statement.

HOSTCHK	*ccc ww*

cc is the checking interval. This interval, in minutes, indicates how often acknowledgment requests are sent out to the logical operator's node.

ww is the response wait interval. This interval is the number of minutes permitted to pass before a response must be received from the logical operator's node.

Notes:

a. *The checking interval specified must be greater than the response wait interval.*

b. *The HOSTCHK function is suspended when an NCCF or NetView operator or a local VM user is assigned as the logical operator. It is resumed when a remote VM user is assigned as the logical operator.*

5. The optional **LOGGING** statement specifies whether messages or messages and command responses are to be logged or not logged. If the LOGGING statement is not in the routing table, messages are logged and LOGGING is ON. If the LOGGING statement is in the routing table, one of the three operands must also be specified, because there is no default operand.

LOGGING	{ ON ALL OFF }

ON indicates that messages are to be logged while this RTABLE is active, unless it is explicitly turned off using the SET LOGGING command.

ALL indicates that messages and programmable operator command responses are to be logged while this RTABLE is active, unless it is explicitly turned off using the SET LOGGING command.

OFF indicates that messages are not to be logged while this RTABLE is active, unless it is explicitly turned on using the SET LOGGING command.

6. The **ROUTE** statement indicates the end of the configuration statements and the start of the routing entries.

ROUTE	

This statement **must** follow the other statements specified in this section.

The LGLOPR and ROUTE statements are required in every routing table. The TEXTSYM, LOGGING, PROPCHK, and HOSTCHK statements are optional. An example of these statements in a routing table is as follows:

```
LGLOPR OPERATNS HOSTNODE
TEXTSYM / $ ¬
PROPCHK 5 1 NODE1 NODE2 NODE3
PROPCHK 3 1 NODE4 NODE5
HOSTCHK 2 1
LOGGING ALL
ROUTE
```

These special statements may be specified for each routing table in any order (as long as LGLOPR is first and ROUTE is last) and with at least one blank separating each parameter. The statements depend on the installation and, therefore, must be supplied by the installation for each routing table. These entries are processed only when the routing table is loaded, so they are not searched during programmable operator message handling.

The configuration shown in Figure 25 can be described in a routing table with the first few lines like those in Figure 26.

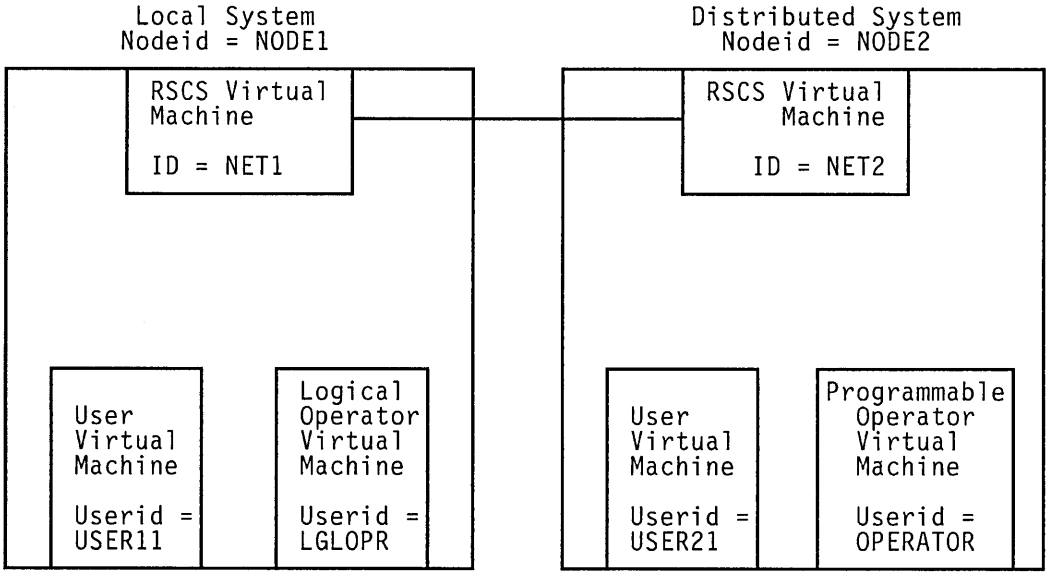


Figure 25. The Programmable Operator Facility in a Distributed System. The logical operator is situated at the Host system and the programmable operator is running in a different system at a distributed site.

The routing table entries to be searched must be in the following fixed format. The contents of the fields described below may appear anywhere in

the defined columns for that field (unaligned) unless the description for that field explicitly states that alignment is required.

Note: The words in parentheses correspond to the vertically aligned words in the comment records in Figure 26 through Figure 31, and also in the IBM sample routing table file.

FIELD	EXAMPLE	FIELD COLUMNS	LENGTH OF FIELD
COMPARISON TEXT (TEXT)	/FEEDBACK /	1-25	25
STARTING COLUMN (SCOL)	1	27-29	3
ENDING COLUMN (ECOL)	9	31-33	3
MESSAGE CLASS (TYPE)	1	35-36	2
USERID (USER)	USER21	38-45	8
NODEID (NODE)	NODE2	47-54	8
ACTION ROUTINE NAME (ACTN)	DMSPOR	56-63	8
PARAMETER TO ACTION ROUTINE (PARM)	TOFB	65-72	8

where:

COMPARISON TEXT
is a particular character string that the programmable operator facility searches for in the incoming message. If this field is left blank, any text compared with this field is considered a match. Multiple texts may be specified, with the capability to skip over intervening blank or non-blank characters.

STARTING COLUMN
is the column in the incoming message where the programmable operator facility starts looking for the character string mentioned in the COMPARISON TEXT field. If this field is left blank, the programmable operator facility starts scanning at the beginning of the message.

ENDING COLUMN
is the column in the incoming message where the programmable operator facility stops looking for the character string(s) mentioned in the COMPARISON TEXT field. If this field is left blank, the programmable operator facility continues scanning until the end of the message.

MESSAGE CLASS
identifies the origin of the incoming message according to the message type. Classes 1-9 are IUCV message types; class 30 is strictly a programmable operator facility message type for NCCF or NetView messages. For more details on IUCV message types, see the Message System Service section of this manual. If this field is left blank, any value compared with this field is considered a match.

The message types available to the programmable operator facility are:

Class	Message Types
1	Message sent using CP MESSAGE and CP MSGNOH.
2	Message sent using CP WARNING.
3	Asynchronous CP messages, CP responses to a CP command executed by the programmable operator facility virtual machine, and any other console I/O initiated by CP.
4	Message sent using CP SMSG command.
5	Any data directed to the virtual console by the virtual machine (WRTERM, LINEDIT, etc.).
6	Error messages from CP (EMSG).
7	Information messages from CP (IMSG).
8	Single Console Image Facility (SCIF) message from CP.
30	Message coming from the Network Communication Control Facility or NetView (*NCCF).

Note: CP responses that are trapped in a buffer using the extended DIAGNOSE code X'08' do not become type-3 messages. For example, CP responses from the programmable operator CMD command are not type-3 messages, and therefore are not logged when LOGGING is set to "ON".

USERID
is the character string compared to the userid of the user that sent the incoming message to the programmable operator facility. It determines the authority of the user to cause an action to be performed. The identifiers for all NCCF or NetView operators, who may use the programmable operator, must be unique. If this field is left blank, all userids compared with this field are assumed to match. If this field is not left blank, the userid must be left-justified in the field or you will receive an error message.

NODEID
is the character string compared to the nodeid of the user that sent the incoming message to the programmable operator facility. Again, it determines the authority of the user to cause an action to be performed. To authorize an NCCF or NetView operator, '*NCCF' or blanks must be in the nodeid field of the routing table. Also, the identifiers for all NCCF or NetView operators, who may use the programmable operator, must be unique. If this field is left blank, all nodeids compared with this field are assumed to match. If this field is not left blank, the nodeid must be left-justified in the field or you will receive an error message.

ACTION ROUTINE NAME
is the name of the LOADLIB member (a Basic Assembler Language routine) that the programmable operator facility is to NUCXLOAD when the LOADTBL function is performed, or the name of an EXEC, and subsequently, the routine that the programmable operator is to call when a match occurs on the entry in which the name is specified. If this field is left blank, no action is performed. If this field is not left blank, the action routine name must be left-justified in the field or you will receive an error message.

PARAMETER TO ACTION ROUTINE
is a character string of up to eight bytes passed as a parameter to the action routine by way of the programmable operator PARMLIST for a Basic Assembler Language routine or by way of a program stack for an EXEC. Often, this is used to specify a particular subroutine in the action routine. If this field is left blank, no parameter is passed. If this field is not left blank, the action routine parameter must be left-justified in the field or you will receive an error message.

Column 73 and beyond are reserved for future use.

```
* THIS IS THE DEFINITION OF THE PROP CONFIGURATION.
* LOGICAL OPERATOR IS NICKNAME "LOP".  SEE "OPERATOR NAMES" FILE.
* LOGICAL OPERATOR (NICKNAME "LOP") IS "OPERATOR" AT NODEID "NODE1".
LGLOPR   LOP
* THE TEXT SEPARATOR CHARACTERS.
TEXTSYM / $ ¬
* WHICH NODES TO CHECK, AND AT WHAT INTERVAL.
HOSTCHK  5  1
* THE ROUTING ENTRIES START
ROUTE
*----------------------- --- --- -- -------- -------- -------- --------
*T                       S   E   T  U        N        A        P
*E                       C   C   Y  S        O        C        A
*X                       O   O   P  E        D        T        R
*T                       L   L   E  R        E        N        M
*----------------------- --- --- -- -------- -------- -------- --------
* SEND PROP FEEDBACK COMMAND TO FEEDBACK ACTION ROUTINE
*----------------------- --- --- -- -------- -------- -------- --------
/FEEDBACK /                1   9  1 USER21   NODE2    DMSPOR   TOFB
*----------------------- --- --- -- -------- -------- -------- --------
* AUTHORIZE NCCF OR NETVIEW OPERATORS TO CHANGE LGLOPR
* WITH THE LGLOPR COMMAND
*----------------------- --- --- -- -------- -------- -------- --------
/LGLOPR /                  1   7 30                    DMSPOR   LGLOPR
*----------------------- --- --- -- -------- -------- -------- --------
* FILTER OUT LOGON AND LOGOFF MESSAGES SO OPERATOR NEEDN'T SEE THEM
* BUT LET "FORCED" LOGOFF MESSAGES THROUGH
*----------------------- --- --- -- -------- -------- -------- --------
$LOGON                    18  24  3
$LOGOFF$¬FORCED           18      3
*----------------------- --- --- -- -------- -------- -------- --------
* FILTER OUT COMMANDS THAT WE DON'T WANT ISSUED.
*----------------------- --- --- -- -------- -------- -------- --------
/CMD /SYSTEM                                          WARNING
/CMD /SET /EC                                         WARNING
*----------------------- --- --- -- -------- -------- -------- --------
* ALLOW ONLY OPERATOR ON HOST TO ISSUE SHUTDOWN.
*----------------------- --- --- -- -------- -------- -------- --------
/CMD /SHUTDOWN                        OPERATOR NODE1    DMSPOR   TOVM
*----------------------- --- --- -- -------- -------- -------- --------
* ROUTE ALL MESSAGES ABOUT DEVICE 00E TO THE SPOOL OPERATOR.
*----------------------- --- --- -- -------- -------- -------- --------
$ 00E                        3                        DMSPOS   SPOOLOP
```

where:

"WARNING"
represents a user action routine which may send a warning message to the user issuing that command.

"SPOOLOP"
represents a nickname or userid of the spool operator.

Figure 26. Partial Routing Table

Tailoring the Routing Table

Routing table entries determine what messages the programmable operator facility ignores (filtering), who is authorized to issue a particular command (authorization), and what action routines to invoke for a given circumstance. You can tailor the routing table to suit your system's individual needs by adding or changing entries in the routing table.

The programmable operator facility comes with a general purpose routing table named "PROP RTABLE". (See the section on "Installing the Programmable Operator Facility" on page 321 to locate the "PROP RTABLE".) You can only use this supplied routing table after the LGLOPR, HOSTCHK, and PROPCHK statements are modified. You may also have to modify the routing table entries. Make these changes using the VM System Product Editor. The programmable operator facility operates satisfactorily with no further changes. However, if you choose, you can modify the supplied routing table to change the operation of the programmable operator facility. You can also create different routing tables to cover varying circumstances. These tables can be dynamically loaded using the LOADTBL command. Only one routing table may be active at a time.

Specifying Routing Texts

Here are more examples of text comparisons for the programmable operator facility. "$" is the arbitrary character separator, "/" is the blank separator, and "¬" is the not-symbol. In all of these examples, it is assumed that the starting and ending columns do not interfere with the matching.

1. The RTABLE entry

   ```
   $LOGOFF
   ```

   is matched by any message containing the word "LOGOFF". If one text is preceded by the arbitrary character separator ($), the text can appear anywhere in the message to be a match.

2. The RTABLE entry

   ```
   /LOGOFF
   ```

   matches the message

   ```
   LOGOFF USER1 IN 5 minutes
   ```

   as there are no non-blank characters preceding the word "LOGOFF", but the entry does not match the message

   ```
   11:20:15 GRAF 0A0 LOGOFF AS USER1     USERS = 020
   ```

   If only one text is preceded by the blank character separator (/), the text must be the first non-blank string in the message order to be a match.

3. The RTABLE entry

   ```
   $AUTO$LOGON$AUTOLOG
   ```

   matches the message

   ```
   11:09:02 AUTO LOGON    ***    USER2      USERS = 021 BY AUTOLOG1
   ```

   and the message

   ```
   11:09:02 AUTO LOGON    ***    AUTOLOG2   USERS = 021 BY SYSTEM
   ```

   but not the message

   ```
   11:09:02 GRAF 0A0 LOGON AS AUTOLOG1    USERS = 023
   ```

   or the message

   ```
   AUTOLOG WON'T LOGON TOMORROW
   ```

A text with two or more texts preceded by arbitrary character separators ($), is matched by a message with all those texts appearing in that order.

The texts in the message are scanned in the order that they appear in the routing table entry. One text is searched for at a time. If the arbitrary character separator ($) precedes the text in the entry, a message is scanned until a match is found or the end of the message is reached. If the blank character separator (/) precedes the comparison text, blanks are skipped over and the first non-blank string of characters is compared to the comparison text, which may or may not match.

Routing table entries and messages are also affected if the text is preceded by a not-symbol (¬). The not-symbol is always used with one of the other separator characters; it never stands alone. If matching text is found and the text in the routing table is preceded only by a "$" or a "/", the position following the last matched text is remembered. If there are no more RTABLE texts to be searched for, the entry is a match. If there is another text in that RTABLE entry to be searched for, the scan continues from the position following the last matched text. A match depends on the rest of the message text and the routing table entry. If matching text *is* found but the text in the routing table *is* preceded by the not-symbol (¬), the entry *is not* a match and checking goes no further. Similarly, if a matching text **is not** found but the text in the routing table **is not** preceded by the not-symbol, the entry **is not** a match. If a match *is not* found and the text *is* preceded by the not-symbol (¬), and if there is no more text, the entry matches the message. If there is more text to scan for, the scan continues as above starting with the character following the last match. A match depends on the rest of the message text and the routing table entry.

Consider the following example:

4. The RTABLE entry

   ```
   $¬AUTO$LOGON
   ```

does match the message

```
12:04:28 GRAF 0A0 LOGON AS USER1      USERS = 027
```

Because the first text in the RTABLE entry (AUTO) is preceded by the arbitrary character separator ($), the entire text is searched for "AUTO". No match is found. Because the text is preceded by the not-symbol (¬), the text is still a match at this point. The scan for the next text (LOGON) begins at the end of the last match. Because there was no previous match, the scan begins again at the start of the message. The LOGON text is preceded by the arbitrary character separator ($), so the search proceeds through the message until "LOGON" is matched. Because "LOGON" appears in the message, this RTABLE entry and message *do* match.

Now consider this example:

5. The RTABLE entry

   ```
   $¬AUTO/LOGON
   ```

   does not match the same message

   ```
   12:04:28 GRAF 0A0 LOGON AS USER1      USERS = 027
   ```

   The message is scanned for "AUTO" as above. The search for "LOGON" again begins at the beginning of the message. In this case, however, the LOGON text is preceded by the blank separator (/), so only blanks are skipped prior to the comparison. No blanks are found, so the comparison is made at the beginning of the text and "LOGON" is compared with "12:04:". This is not a match. Because this text was not preceded by the not-symbol, this RTABLE entry and message *do not* match.

Another example:

6. The RTABLE entry

   ```
   $¬AUTO/LOGON
   ```

   does not match the message

   ```
   12:04:28 AUTO LOGON *** USER1    USERS = 027 BY AUTOLOG1
   ```

   Because "AUTO" is found in the message and is preceded in the RTABLE entry by the arbitrary character separator ($) and the not-symbol (¬), the RTABLE entry and message *do not* match.

Here is another example:

7. The RTABLE entry

   ```
   $LOGOFF$¬030/FORCED
   ```

   does not match the message

```
12:04:28 USER DSC LOGOFF AS USER1      USERS = 026 FORCED
```

The first text, "LOGOFF", is preceded by the arbitrary character separator ($) and is scanned for through the text. "LOGOFF" is found. Because "LOGOFF" is not preceded by the not-symbol, the next text is scanned. The scan continues from the end of the previous match, which is the character following the LOGOFF text. Since the arbitrary character separator ($), precedes "030", the entire remaining text is searched for "030". It is not found but because "030" is preceded by the not-symbol, the message and RTABLE entry still match. Finally, "FORCED" is scanned for. It is preceded by the blank separator (/). Blanks are skipped, and starting with the character following the last matched string (which was "LOGOFF"), "FORCED" is compared to "AS USE". This is not a match. Because "FORCED" is not matched and is not preceded by the not-symbol, this RTABLE entry and message *do not* match.

Here are routing table entries that *do* match this message:

```
$LOGOFF$¬030$FORCED
```

would match because arbitrary characters would be skipped before comparison for "FORCED" and

```
$LOGOFF$¬030/¬FORCED
```

would match because the first non-blank string after "LOGOFF" is not "FORCED".

Specifying Routing Texts to an NCCF or NetView Operator

To route a message to an NCCF or NetView logical operator, that NCCF or NetView operator must have issued a LGLOPR ASN or LGLOPR RPL command. Be sure that the appropriate NCCF or NetView operators are authorized in the active routing table to issue the command.

For example, to route the message

```
12:04:15 GRAF 0A0 LOGOFF AS USER1     USERS = 020  FORCED
```

to the NCCF or NetView logical operator, the routing table should have the following entries:

```
:
:
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* AUTHORIZE OPERATORS TO CHANGE LGLOPR ASSIGNMENT
*------------------------ --- --- -- -------- -------- -------- --------
/LGLOPR /                   1   7 30                   DMSPOR   LGLOPR
*------------------------ --- --- -- -------- -------- -------- --------
* FILTER OUT SPECIFIC MESSAGES
*------------------------ --- --- -- -------- -------- -------- --------
$GRAF$LOGOFF AS$FORCED$              3                DMSPOS   LGLOPR
*------------------------ --- --- -- -------- -------- -------- --------
:
:
```

Figure 27. Routing Entries to Send Messages to an NCCF or NetView Operator

DMSPOR LGLOPR specifies that the programmable operator LGLOPR command can be executed. Message type 30 says that the user executing the LGLOPR command must be an NCCF or NetView operator. DMSPOS LGLOPR specifies that the desired action is to send the matching message to the logical operator. First, the NCCF or NetView logical operator must have issued LGLOPR ASN or LGLOPR RPL command to the programmable operator facility. Then, when the above message arrives at the programmable operator virtual machine, it is routed to the assigned NCCF or NetView logical operator.

If you wish to put an id in the nodeid field of the routing table entries for the NCCF or NetView operator, it must be "*NCCF".

Filtering Messages

This is the simplest application of the programmable operator facility. Entries can be placed in the routing table to filter informational messages. The messages are filtered because no action routine is specified in the routing table entries. For example, when the programmable operator facility is running in the system operator's virtual machine, informational messages resulting from commands such as, LOGON, LOGOFF, and DISCONN, can be prevented from being displayed at the logical operator's console. Although the messages are not displayed at the operator's console, they can be logged in the current day's log file. The routing table entries must identify the text(s) in the message that makes it unique and identify the columns between which the text(s) should be found in the message. With single or multiple texts, TEXTSYM characters should be selected accordingly. Figure 28 shows an example of how entries may be placed in the routing table to filter unwanted responses directed to the logical operator. For example, using Figure 28 below as the routing table, the message

```
12:04:50 GRAF 055 LOGON AS USER1
```

would match the second routing table entry (/¬AUTO$LOGON). This RTABLE specification means that, starting in column 9 (SCOL) of the message, "AUTO" cannot be the first non-blank string and that "LOGON" must appear somewhere in the message. The message would be filtered out but logged in the current day's log file.

However, the message

```
12:04:28 AUTO LOGON *** USER BY AUTOLOG1
```

would not match the second routing table entry (/¬AUTO$LOGON) because "AUTO" is the first non-blank string in the message appearing in the columns between the SCOL and ECOL fields. Thus, the message would not be filtered out and would be routed to the logical operator, as specified in the last entry in Figure 28.

```
* THIS IS THE DEFINITION OF THE PROP CONFIGURATION.
* LOGICAL OPERATOR IS NICKNAME "LOP".  SEE "OPERATOR NAMES" FILE.
LGLOPR   LOP
* THE TEXT SEPARATOR CHARACTERS.
TEXTSYM / $ ¬
* WHICH NODES TO CHECK, AND AT WHAT INTERVAL.
PROPCHK  5  1  NODE2A    NODE2B
PROPCHK  2  1  NODE1A    NODE1B
* THE ROUTING ENTRIES START
ROUTE
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* FILTER OUT LOGON AND LOGOFF MESSAGES SO OPERATOR NEEDN'T SEE THEM
*------------------------ --- --- -- -------- -------- -------- --------
$OUTPUT/OF                 19  36  3
/¬AUTO$LOGON                9  33  3
$LOGOFF                    19  34  3
$DSCONNECT                 19  36  3
$RECONNECT                 19  36  3
$DIAL                      19  32  3
$DROP                      19  32  3
*------------------------ --- --- -- -------- -------- -------- --------
* SEND REMAINING ASYNCHRONOUS CP MESSAGES TO LOGICAL OPERATOR
*------------------------ --- --- -- -------- -------- -------- --------
                                   3                  DMSPOS   LGLOPR
```

Figure 28. Routing Entries to Filter Responses to Routine Commands

In Figure 28, the entries that appear in the "TEXT" field (OUTPUT OF, LOGON, etc.) are the texts contained in the messages that are to be trapped by the programmable operator facility when they are issued by CP.

No userids and nodeids are specified for these entries because they are issued by CP. Because no action routine is specified, the only action taken is the logging of the messages in the current day's log file.

Looking at the last line in Figure 28, you can see that if a type-3 IUCV message is received that does not have a corresponding entry in the routing table, action routine DMSPOS together with the LGLOPR parameter routes the message to the logical operator. In this case, this entry has to be placed after the specific text entries that you want filtered from the message stream. If this entry appeared before the text entries in Figure 28, all type-3 IUCV messages would be routed to the logical operator.

Controlling Authorization

The routing table determines who is authorized to issue specific commands in the programmable operator facility. Programmable operator authorization is based entirely on the contents of the routing table. Therefore, controlling authorization is a relatively simple procedure. Authorization checking uses either the userid, nodeid, the command text, or any combination of these fields in a routing table entry. A change to any of these fields can result in a change in authorization. You can easily tailor the authorization structure to your particular needs by changing only these fields in the routing table entries, without changing the action routines.

When a userid and nodeid *are not* specified for a routing table entry, all users are authorized to match that entry and to use the function that it describes. Figure 29 shows an example of unrestricted authorization for the FEEDBACK command. A message sent with the FEEDBACK command is passed to module DMSPOR, which supports most of the programmable operator commands. The TOFB parameter invokes the proper action routine contained in module DMSPOR that writes the message to the FEEDBACK file. (See "The Feedback File" on page 358 or"FEEDBACK Command" on page 371 .

Note: In the following examples, the TYPE (message class) field is left blank to allow the FEEDBACK (or FB) command to be issued with any class of IUCV message. The ECOL fields are 9 and 3 because the character string being looked for is FEEDBACK or FB followed by a blank, for example, "FEEDBACK " or "FB " would match.

```
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* PLACE A FEEDBACK MESSAGE IN THE PROP FEEDBACK FILE
*------------------------ --- --- -- -------- -------- -------- --------
/FEEDBACK /                 1   9                      DMSPOR   TOFB
/FB /                       1   3                      DMSPOR   TOFB
   :                        :   :                        :        :
   :                        :   :                        :        :
```

Figure 29. Uncontrolled Authorization

Authorization can be restricted to users at a particular network node by specifying only the nodeid. In Figure 30, only users at NODE1 are authorized to issue the FEEDBACK command.

```
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* PLACE A FEEDBACK MESSAGE IN THE PROP FEEDBACK FILE
*------------------------ --- --- -- -------- -------- -------- --------
/FEEDBACK /                 1   9              NODE1    DMSPOR   TOFB
/FB /                       1   3              NODE1    DMSPOR   TOFB
   :                        :   :                :        :        :
   :                        :   :                :        :        :
```

Figure 30. Restricting Authorization by Nodeid

When a userid and nodeid are specified, only that user at the specified node is authorized to match that entry. In Figure 31, only JOHNDOE at NODE1 and JANEDOE at NODE2 are authorized to place messages in the feedback file.

```
*------------------------ --- --- -- -------- -------- -------- --------
*T                        S   E   T  U        N        A        P
*E                        C   C   Y  S        O        C        A
*X                        O   O   P  E        D        T        R
*T                        L   L   E  R        E        N        M
*------------------------ --- --- -- -------- -------- -------- --------
* PLACE A FEEDBACK MESSAGE IN THE PROP FEEDBACK FILE
*------------------------ --- --- -- -------- -------- -------- --------
/FEEDBACK /                 1   9    JOHNDOE  NODE1    DMSPOR   TOFB
/FEEDBACK /                 1   9    JANEDOE  NODE2    DMSPOR   TOFB
/FB /                       1   3    JOHNDOE  NODE1    DMSPOR   TOFB
/FB /                       1   3    JANEDOE  NODE2    DMSPOR   TOFB
  :                         :   :      :        :        :        :
  :                         :   :      :        :        :        :
*------------------------ --- --- -- -------- -------- -------- --------
* SEND REMAINING REQUESTS AND COMMANDS TO THE LOGICAL OPERATOR
*------------------------ --- --- -- -------- -------- -------- --------
                                                        DMSPOS   LGLOPR
```

Figure 31. Restricting Authorization by Userid and Nodeid

Since the user must explicitly issue the FEEDBACK or FB command to have a message placed in the feedback file, action routine DMSPOR TOFB must be specified in the routing table to carry out the required action. Any user trying to issue the FEEDBACK command that is not authorized by the routing table in Figure 31 will have their command sent to the logical operator as a message via action routine DMSPOS with parameter LGLOPR, as specified by the last record of the routing table.

Additional userids and nodeids may be added to the table to grant authorization to issue these commands. Conversely, userids and nodeids may be removed to revoke authorization.

Restricting Command Use

You can easily restrict command use to a specific group of users. You can specify first entries for those users who should be allowed to use the command, then specify an entry to trap the use of that command by any other user. Finally have an entry for any users who should have general command usage except for the restriction. The following example demonstrates command restriction.

```
*----------------------- --- --- -- -------- -------- -------- --------
*T                       S   E   T  U        N        A        P
*E                       C   C   Y  S        O        C        A
*X                       O   O   P  E        D        T        R
*T                       L   L   E  R        E        N        M
*----------------------- --- --- -- -------- -------- -------- --------
/CMD /SHUTDOWN /           1  12    LGLOPR   HOSTNODE DMSPOR   TOVM
/CMD /SHUTDOWN /                                     ACTIONX
/CMD /NETWORK /SHUTDOWN /  1  25    NETOP1   HOSTNODE DMSPOR   TOVM
/CMD /NETWORK /SHUTDOWN /                            ACTIONX
/CMD /NETWORK /            1  20    NETOP1   HOSTNODE DMSPOR   TOVM
/CMD /NETWORK /            1  20    NETOP    RMTNODE  DMSPOR   TOVM
/CMD /NETWORK /                                      ACTIONX
/CMD /                     1   4    LGLOPR   HOSTNODE DMSPOR   TOVM
/CMD /                     1   4    MAINT    HOSTNODE DMSPOR   TOVM
/CMD /                     1   4    MAINT    PROPNODE DMSPOR   TOVM
*----------------------- --- --- -- -------- -------- -------- --------
```

Figure 32. Restricting Command Use to Specific Users

An RTABLE containing these entries allows LGLOPR at HOSTNODE to use the SHUTDOWN command, but anyone else trying to use it invokes ACTIONX. ACTIONX could be DMSPOS sending the offending command to the logical operator or it could be a user-written action routine taking some other action against the issuer, depending of the severity of the offense. The same description applies to the NETWORK SHUTDOWN and NETWORK commands. So, routing table coding provides considerable flexibility in granting or restricting command authority.

The Log File

If LOGGING is set to ON or ALL, every incoming message that the programmable operator facility receives is put into a CMS file referred to as the **log file**. If LOGGING is set to ALL, all error messages and command responses generated by the programmable operator facility are also put in the log file. If LOGGING is set ON, responses from CP, CMS, and programmable operator facility commands are not logged; messages are.

Each message is identified by the date and time received. The userid and nodeid appear only if the text was sent by a CP MSG, SMSG, WNG, or sent using SCIF (Single Console Image Facility). The userid and nodeid are blank for a message sent by CP. A message sent by a remote RSCS network virtual machine has a nodeid, but no userid. A message sent from an NCCF or NetView operator console has '*NCCF' as the nodeid.

Log entries generated and logged by the programmable operator have a userid of PROP. The log file has the following format:

```
col 1     col 10    col 19    col 28      col 39
|         |         |         |           |
V         V         V         V           V
yy/mm/dd hh:mm:ssffluserid    nodeid":    text
```

The log file contains variable length records. The maximum record length that the programmable operator facility can place in the log file is 132 characters. Because the prefix uses 38 of the 132 characters, the text can be only 94 characters long. Therefore if the text of a message exceeds the maximum length of 94 characters the overflow is continued on the next record. This continued record has the same prefix as the preceding record, with no colon (:) preceding the text in column 37.

A separate log file is started for each day. The name of the file is:

```
LGyymmdd nodeid A5
```

where:

yy is the current year

mm is the current month

dd is the current day

nodeid is the current RSCS nodeid of the system on which the programmable operator facility is running.

When the programmable operator facility is started, stopped, or debug mode is changed, a record is written to the log file. The messages written to the file have the normal log prefix and a text corresponding to the changed function. Generally, responses to the programmable operator *console* commands are written to the log file when LOGGING is set to ON or ALL. It is also possible to have responses from the programmable operator commands written to the log file. See the LOGGING statement of the routing table or the programmable operator SET command for more information. Note that when LOGGING is set to ALL, the log file may be used as an alternative to spooling the virtual console. When node-checking is in effect, by having PROPCHK or HOSTCHK statements in the RTABLE, if a node changes status from UP to DOWN or vice versa, a message is also written to the log file.

If a virtual machine resource limit is reached, such as "disk-full", it may not be possible to write another record to the programmable operator

facility log file. If this happens, a user-written EXEC is invoked to perform whatever recovery action the user thinks is desirable or necessary. The user EXEC must have the filename of PROPLGER. See "LOG Error Exit" on page 397 later in this section.

Any VM user authorized in the active routing table[17] can obtain the log file as a reader spool file by using the programmable operator GET LOG command (this does not include NCCF or NetView users). Messages can be placed in the log file by authorized users by using the programmable operator LOG command with no other action being taken.

An old log file can be purged by any user authorized in the active routing table to use the programmable operator CMD command by issuing the CMS ERASE command.

Logging NCCF or NetView messages in the Log File

A message from an NCCF or NetView operator is logged just as any other message sent to the programmable operator. When logging a message or command from an NCCF or NetView operator, the keyword "*NCCF" is placed in the nodeid field of the log record. You must ensure that the identifiers for all of the NCCF or NetView operators who may access the programmable operator facility are unique. This is also an NCCF or NetView requirement. As a result, the operator identifier with the "*NCCF" nodeid is sufficient to identify the issuer of the programmable operator command. Also, messages received by the programmable operator facility from NCCF or NetView have the message type of '30'.

Ensuring a Complete Log

When the programmable operator facility routes a message to the logical operator, the message contains the userid of the sender. The operator, in responding to the message, may choose to send a message directly to the user without going through the programmable operator facility. However, if this is done, the message is not logged in the log file. To ensure that these messages are logged, the operator should send the message to the user through the programmable operator facility by using the programmable operator facility CMD command. For information about the programmable operator facility CMD command, See the "CMD Command" on page 369.

Whether the message was sent through the programmable operator or not has little significance to the user. However, so that the messages received by the user always have the same id (the programmable operator facility id), the message should always be sent from the logical operator through the programmable operator facility.

---

[17] An NCCF or NetView operator cannot use the GET command to obtain the log file. Use CMS commands (and the programmable operator CMD command) to type the file or portions of the file or to send the file to a userid where you can process it.

In a Single System: To route a message through the programmable operator facility where the operator and user are on the same physical system, use the MSG command:

MSG *operator* **CMD MSG** *user1* **- RESPONDING TO YOUR REQUEST**

operator
is the userid of the programmable operator facility virtual machine.

CMD
is the programmable operator facility CMD command.

MSG
is the VM command that the programmable operator facility will execute.

user1
is the userid of the user who will receive the message.

RESPONDING TO YOUR REQUEST
is the message text sent to the user.

In VM Distributed Systems: To route a message through the programmable operator facility where the operator and user are not on the same physical VM system, use the SMSG command:

SMSG *net1* MSG *node1 operator* **CMD MSG** *user1* **- RESPONDING TO YOUR REQUEST**

net1
is the userid of the network machine at the user's node.

MSG
(first appearance) is the RSCS message command.

node1
is the nodeid of the programmable operator facility virtual machine.

operator
is the userid of the programmable operator facility virtual machine.

CMD
is the programmable operator facility CMD command.

MSG
(second appearance) is the VM command that the programmable operator facility executes.

user1
is the userid of the user to receive the message.

RESPONDING TO YOUR REQUEST
is the message text sent to the user.

In a Mixed Environment: To route a message through the programmable operator facility from the logical operator who is an NCCF or NetView operator to the user on a VM system, use PROP, an NCCF and NetView command as follows:

PROP CMD MSG *user1* **- RESPONDING TO YOUR REQUEST**

PROP
is the NCCF or NetView command that sends the message from an NCCF or NetView operator to the programmable operator facility.

CMD
is the programmable operator CMD command.

MSG
is the VM command that the programmable operator facility will execute.

user1
is the userid of the VM user who will receive the message.

RESPONDING TO YOUR REQUEST
is the message text sent to the user.

Note: Refer to "Helpful Hints" on page 367 for ways to reduce typing of long text strings.

The Feedback File

The feedback file is another CMS disk file (named FEEDBACK nodeid A5) that the programmable operator facility manages. The feedback file, unlike the log file, is not automatically written by the programmable operator facility. Authorized users can write time stamped notes and complaints about the operation of the system to this feedback file. To write a notice to the feedback file, you, as a user, must explicitly use the FEEDBACK (or FB) command. An example of such a message is

```
M OP FEEDBACK RESPONSE TIME WAS SLOW DURING MORNING SHIFT.
```

Because the feedback file is normally smaller than the log file, it is easier for the personnel in charge of the programmable operator facility's maintenance to review the users' comments and identify when and where particular problems occurred.

Each record in the feedback file is prefixed with the date and time the message was logged along with the sender's userid and nodeid. The feedback file has the following format:

```
col 1     col 10    col 19    col 28      col 39
|         |         |         |           |
V         V         V         V           V
yy/mm/dd hh:mm:ss userid    nodeid:     text
```

The feedback file contains variable length records. The maximum record length that the programmable operator facility can place in the feedback file is 132 characters. Because the prefix uses 38 of the 132 characters, the text can be only 94 characters long. Therefore if the text of a message exceeds the maximum length of 94 characters the overflow is continued on the next record. This continued record has the same prefix as the preceding record, with no colon preceding the text.

Any user authorized in the active routing table except an NCCF or NetView operator[18] can obtain the feedback file as a reader spool file by using the programmable operator GET FEEDBACK or GET FB command.

An old feedback file can be purged by any user authorized in the active routing table to use the programmable operator CMD command by issuing the CMS ERASE command.

Communications Checking

In a VM-only environment, the programmable operator facility can operate either from the host system or from a distributed system in a network or from both sides. Special functions can be performed depending on the ability of the programmable operators to communicate through RSCS Networking. The purpose of these functions is:

- To provide the host operator (logical operator) with timely information and/or action in the event of a break in communication with the programmable operator on one of the network's distributed systems.
- To provide a distributed system with timely information and/or capability for action in the event of a break in communication with the host system.

A programmable operator can periodically check on the link with another system to determine whether it is possible to communicate with that system. The systems to be checked must be identified in the routing table of the programmable operator doing the checking. This may be either the programmable operator at the host system checking on specified distributed systems or a distributed programmable operator checking on communications with its host system. When the programmable operator at the host system is checking on the distributed systems, the programmable operator needs another programmable operator running in the system

---

[18] An NCCF or NetView operator cannot use the GET command to obtain the feedback file. Use CMS commands (and the programmable operator CMD command) to type the file or portions of the file or to send the file to a userid where you can process it.

operator virtual machine on the distributed system. This is not required when a distributed system is checking on the host system. In other words, for "host checking", no programmable operator is required at the host system, but for "distributed system checking" programmable operators must be running at the distributed systems. These various types of checking may be collectively referred to as "node-checking".

Note that the roles of the 'host' and 'distributed' systems need not be strictly defined. For example, a programmable operator may use the PROPCHK function to check communication with *any* other system (node) running a programmable operator in its system operator virtual machine in the network. With the HOSTCHK function, the system being checked is simply the system defined as the checking system's logical operator, and not necessarily 'the' host system for the network.

The programmable operator facility that has been instructed to check on its distributed system(s) periodically tries to communicate with those systems by sending a message that causes a response. The programmable operator then waits a specified time for a response. For checking the host system (HOSTCHK), the acknowledgement request goes to the RSCS on the logical operator node. For checking the distributed systems (PROPCHK), it goes to the programmable operator on the distributed system. No response indicates that something has prevented communication between the host and the distributed system(s). Getting a response after being delinquent for a time indicates that communication between the programmable operators has been restored. With the SET command, the user is able to set the checking function ON or OFF.

Any time that the programmable operator detects that a node has exceeded the time allowed for responding, that fact is recorded in the programmable operator log. Also logged is the fact that a node has resumed responding.

When one of these conditions, no response or a late response, is detected, the programmable operator facility invokes one of two EXECs supplied by IBM for this purpose. For checking a distributed system, the PROPPCHK EXEC is invoked. For checking on the host, the PROPHCHK EXEC is invoked.

The programmable operator doing the checking invokes the EXEC. For example, if a programmable operator on a distributed system has a HOSTCHK statement in its routing table, the PROPHCHK EXEC would be invoked if communication with the host system were lost for a long enough period that the request or the response were prevented from getting through. Similarly, if a programmable operator on a host system has a PROPCHK statement in its routing table, that programmable operator would invoke the PROPPCHK EXEC if communication with one of the specified distributed nodes were lost for such a period. These EXECs are supplied as samples only and may be modified or replaced with user-written EXECs, allowing the user to tailor the resulting action(s).

The IBM-supplied PROPHCHK EXEC operates as follows:

- When the logical operator's node fails to respond or resumes responding, type a message on the programmable operator console and send a message to the userid MAINT indicating that communication with the host system has been broken or restored.

The IBM-supplied PROPPCHK EXEC operates as follows:

- For each node that has failed to respond, notify the logical operator that the programmable operator facility is unable to communicate with that particular node (distributed system).

- For each node that has resumed responding from a failed state, notify the logical operator that communication with that node (distributed system) has been reestablished.

Note: PROPPCHK does not try to send a message to the logical operator if the logical operator is an NCCF or NetView operator.

Invoking Programmable Operator Facility Commands

The commands used with the Programmable Operator Facility can be executed by anyone who is authorized by the active routing table. If an unauthorized user issues a programmable operator facility command, the command is not executed but is routed to the logical operator. To send a command to the programmable operator facility you must send a message to the programmable operator facility virtual machine. The text of the message is the command to be issued. Unless otherwise noted, the response from the command is returned to the user who sent the command to the programmable operator facility virtual machine. Responses are sent one line at a time via the CP MSGNOH or MESSAGE commands. For NCCF or NetView operators, the responses are sent via the Programmable Operator/NCCF Message Exchange (PMX). These responses are displayed in the language in which the programmable operator virtual machine is set.

The format of the message sent to the programmable operator facility virtual machine is the same as used with the CP MESSAGE command for local use and the CP SMSG command for distributed (network) use. In a mixed environment, an NCCF or NetView operator uses PROP, an NCCF or NetView command, to send messages to the programmable operator facility virtual machine.

The local format is:

Message **MSG**	*userid* propcmd [parameters]

The distributed (network) format is:

SMsg	*netid* **Msg** *nodeid userid* propcmd **[** parameters **]**

netid
is the userid of the RSCS network machine at the user's node.

Msg
is the required RSCS message command. It can be entered as M or MSG.

nodeid
is the nodeid of the programmable operator facility virtual machine.

userid
is the userid of the programmable operator facility virtual machine.

propcmd
is the command to be executed by the programmable operator facility. See "Programmable Operator Facility Command Descriptions" for information about the commands.

parameters
are the parameters associated with the command to be executed.

The format for an NCCF or NetView operator station is:

PROP	propcmd [parameters]

propcmd
is the command that the programmable operator facility will execute

parameters
are the parameters associated with the command to be executed.

Notes:

1. To use this format, you must be authorized by NCCF or NetView
2. If you (NCCF or NetView operator) are not attached to the local system, use the NCCF or NetView ROUTE command to specify the domain where you want to execute the command.

Issuing Commands in the Single System Environment

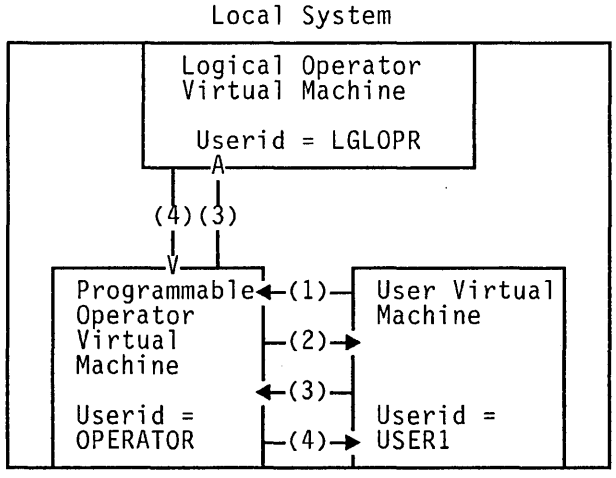
If the Programmable Operator Facility is running in the operator's virtual machine, a user on the same physical machine might send the messages:

```
MESSAGE OPERATOR QUERY RTABLE

MESSAGE OPERATOR WHAT TIME IS SHUT DOWN?
```

where "QUERY" is a programmable operator facility command. See "QUERY Command" on page 379 for information about the programmable operator facility QUERY command.

Figure 33 shows how messages are transferred between a user, the programmable operator facility, and the logical operator when all are located on the same physical system.



(1) USER1 issues the message:

```
MESSAGE OPERATOR QUERY RTABLE
```

(2) The programmable operator facility responds by sending the following message to USER1:

```
PROP running with routing table 'PROP RTABLE A5'
```

(3) USER1 issues the message:

```
MESSAGE OPERATOR WHAT TIME IS SHUT DOWN?
```

Because this message cannot be processed by the programmable operator facility, the message is routed to the logical operator.

(4) The logical operator responds to USER1's message by sending a message to USER1 by way of the programmable operator facility. (See 'Ensuring a Complete Log').

Figure 33. Example of Communication in the Single System Environment

Issuing Commands In The Distributed VM Environment

A user (USER21) on another VM system (assuming a remote system with the nodeid of "NODE2", a network machine with the id of "NET2", and that the programmable operator facility is running in the operator's virtual machine with a userid of "OPERATOR") might send the messages:

```
MESSAGE OPERATOR QUERY RTABLE

MESSAGE OPERATOR WHAT TIME IS SHUT DOWN?
```

where "QUERY" is a programmable operator facility command. See "QUERY Command" on page 379 for information about the programmable operator facility QUERY command.

Figure 34 shows how a user on one physical system exchanges messages with the programmable operator facility on a different physical system and how the programmable operator facility responds directly to a user on the same physical system or routes messages to the logical operator who is located on a different physical system.

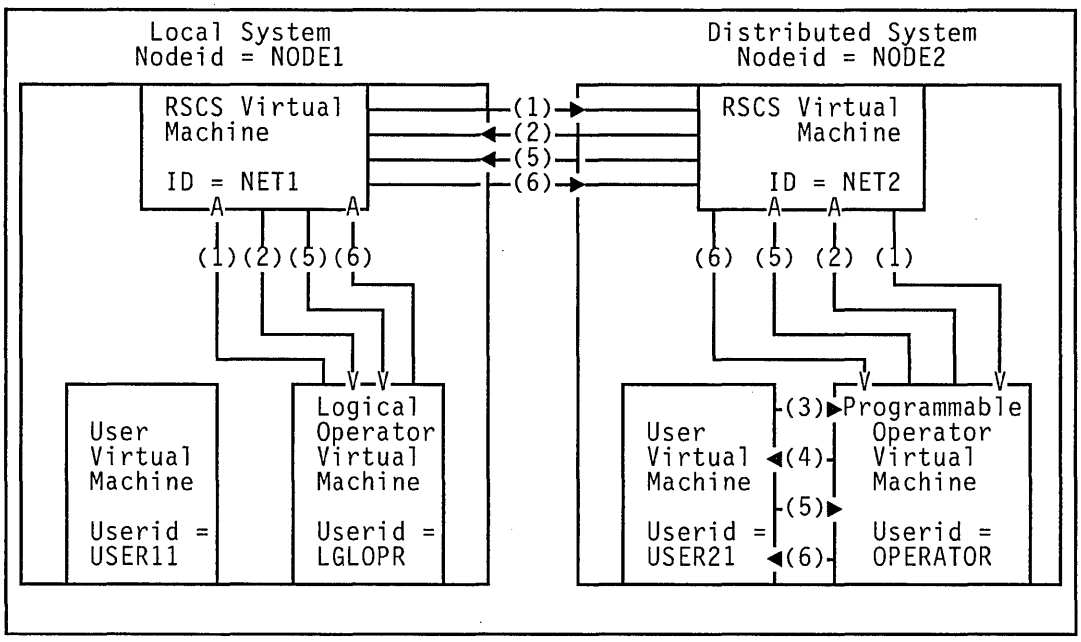


Figure 34 (Part 1 of 2). Example of Communication in the Distributed Environment

```
(1)  LGLOPR issues the message:

     CP SMSG NET1 MSG NODE2 OPERATOR QUERY RTABLE

(2)  The programmable operator facility responds by sending the
     following message to LGLOPR:

     PROP running with routing table 'PROP RTABLE A5'

(3)  USER21 issues the message:

     MESSAGE OPERATOR QUERY RTABLE

(4)  The programmable operator facility responds by sending the
     following message to USER21:

     PROP running with routing table 'PROP RTABLE A5'

(5)  USER21 issues the message:

     MESSAGE OPERATOR WHAT TIME IS SHUT DOWN?

     Because this message cannot be processed by the programmable
     operator facility, the message is routed to the logical
     operator.

(6)  The logical operator responds to USER21's message by sending a
     message to USER21 by way of the programmable operator facility.
     (See 'Ensuring a Complete Log').
```

Figure 34 (Part 2 of 2). Example of Communication in the Distributed Environment

Issuing Commands in a Mixed Environment

In a mixed environment, message paths are more complex. A user on a VM system (VMUSER3) might send the messages:

```
MESSAGE OPERATOR QUERY RTABLE

MESSAGE OPERATOR WHAT TIME IS SHUT DOWN?
```

where "QUERY" is a programmable operator facility command. See "QUERY Command" on page 379 for information about the programmable operator QUERY command.

The message sent containing the QUERY command simply follows the paths of a message sent by a user to the programmable operator facility in the same system. The message is processed by the programmable operator facility and a response is returned to the user (VMUSER3).

Figure 35 on page 366 shows how messages are transferred between a user, the programmable operator facility, and an NCCF or NetView logical operator. It also shows how messages and commands are transferred from an NCCF or NetView operator to the programmable operator facility and its VM system.

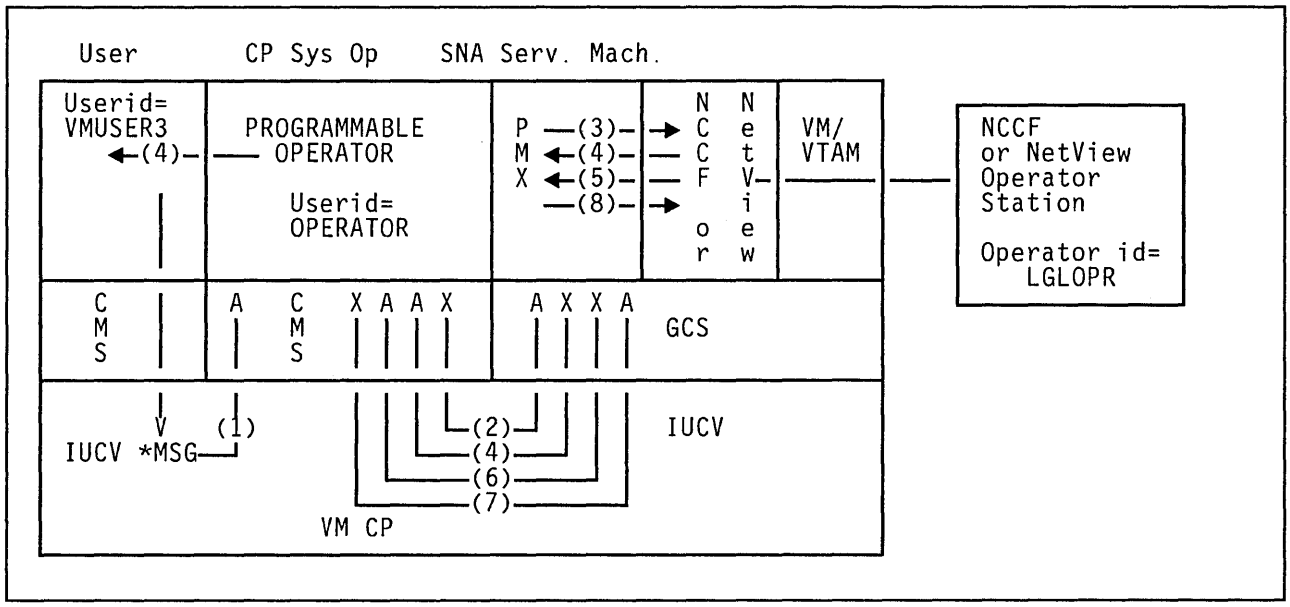


Figure 35 (Part 1 of 2). Example of Communication in the Mixed Environment

(1) VMUSER3 issues the message

MESSAGE OPERATOR WHAT TIME IS SHUT DOWN?

that reaches the programmable operator facility through the IUCV *MSG service. The programmable operator facility determines that the message is to be routed to the logical operator.

(2) The programmable operator determines that the logical operator is an NCCF or NetView operator and sends the message though IUCV to the PMX.

(3) The PMX, upon receiving the message from IUCV, queues (through the NCCF or NetView Message Queueing Service) the message for display on the specified NCCF or NetView operator's console.

(4) Upon receiving the message from VMUSER3, the NCCF or NetView logical operator issues the command

PROP CMD MSG VMUSER3 SHUTDOWN IS AT 10:00 PM.

which travels back through the PMX, IUCV, and the programmable operator facility, before the response reaches VMUSER3's screen.

(5) An NCCF or NetView operator issues a programmable operator command

PROP QUERY RTABLE

which is queued for the PMX.

(6) The PMX sends the command as an IUCV message to the programmable operator.

(7) The programmable operator receives the message from IUCV and processes it. Upon determining that the requester was an NCCF or NetView operator, the programmable operator sends any responses through IUCV to the PMX.

(8) The PMX receives the response:

PROP running with routing table 'PROP RTABLE A5'

from IUCV and queues the response for display on the specified NCCF or NetView operator's terminal.

Figure 35 (Part 2 of 2). Example of Communication in the Mixed Environment

Helpful Hints

The typing of large text strings, such as the message command string preceding the message text, can be minimized by assigning the string, up to the userid, to a PF key. An EXEC can also be used that prompts for, or accepts as parameters, that part of the message command that the user must type in.

For example:

SET PF01 IMMED SMSG NET1 MSG NODE2 OPERATOR QUERY RTABLE

will allow the logical operator to press PF01 to find out the name of the active routing table when the logical operator and programmable operator facility are located on different physical systems.

The CMS TELL and NAMES exec procedures may also be used by the logical operator to send a message to a user. If the logical operator has a NAMES file entry assigning the nickname PROP1 to OP at node NODE1, the logical operator could send a message in the following manner:

```
TELL PROP1 QUERY RTABLE

TELL PROP1 CMD WNG ALL ...WILL RE-IPL IN 5 MIN - PLS LOGOFF.
```

These exec procedures are described in *VM/SP CMS Command Reference* and *VM/SP CMS User's Guide.*

Programmable Operator Facility Command Descriptions

The routing table controls authorization for the programmable operator facility. So, in the following command descriptions, an *authorized user* must have a routing table entry for the command he or she wants to issue. All programmable operator facility commands are sent by users as messages; these messages should be in English to ensure that they get routed properly.

CMD Command

Use the CMD command to execute CP or CMS commands in the programmable operator facility virtual machine. The command is accepted or rejected by CP based on the CP user class defined for the programmable operator facility virtual machine. Also, the programmable operator facility may reject or accept the command based on the authorization granted in the active routing table. The user class assigned to the programmable operator facility virtual machine is determined by the installation. If accepted, the response from the command is returned to the issuer of the command.

The format of the CMD command is:

CMD	*vmcmd*

vmcmd
is the VM command sent to the programmable operator facility virtual machine for execution according to the CMS IMPCP and IMPEX settings.

Usage Notes

1. Any commands that alter or overlay CMS storage (CP DEFINE STORAGE, CP IPL CMS, CP SHUTDOWN, etc.) will have an adverse effect on the operation of the programmable operator facility and should not be issued under the programmable operator facility.

2. Reissuing the PROP command once the programmable operator facility is running will cause it to stop operating correctly. The user must then re-IPL CMS and restart the programmable operator facility using the procedure described under "Invoking the Programmable Operator Facility" on page 322.

3. Issuing commands that cause a VM READ or a CP READ (such as the DDR command) will stop the programmable operator facility. It must then be restarted using the procedure described under "Invoking the Programmable Operator Facility," or the read must be answered from the console of the programmable operator facility virtual machine. Commands of this type should not be sent to the programmable operator facility.

4. Line editing characters (CHARDEL, LINEDEL, LINEND, and ESCAPE), although interpreted by your terminal when you have SET LINEDIT ON in effect, are not interpreted as line editing characters when sent to the programmable operator facility. In other words, they are interpreted as the characters they are (that is, @, ¢, #, :). For example, the string:

M OP CMD ACCESS 191 A"#RECEIVE

will cause an "INVALID MODE " A#RECEIV "" message to be returned to the issuer. The default line editing characters may be defined by the installation. You may also define your own line editing characters by using the CP TERMINAL command.

5. The programmable operator facility does not recognize the CMS immediate commands (HB, HI, HO, HT, HX, RO, RT, SO, TE, and TS). If you issue any of these commands, the programmable operator facility issues "Unknown CP/CMS command".

Examples

The command issued at a virtual machine console:

M OP CMD INDICATE

M OP CMD QUERY FILES

M OP CMD QUERY PRINTER ALL

M OP CMD ERASE LG861015 NODE1 A5

M OP CMD QUERY SEARCH

M OP CMD LISTFILE

The command issued at an NCCF or NetView operator terminal:

PROP CMD INDICATE

PROP CMD QUERY FILES

PROP CMD QUERY PRINTER ALL

PROP CMD ERASE LG861015 NODE1 A5

PROP CMD QUERY SEARCH

PROP CMD LISTFILE

Responses

The response returned by CP or CMS is sent to the issuer of the command. After the response from CP or CMS, the programmable operator facility responds with:

Command complete

FEEDBACK Command

Use the FEEDBACK command to place comments about the operation of the system and/or the programmable operator facility in the feedback file. These comments are available for review by personnel responsible for maintenance of the programmable operator facility. The comment (preceded by the date and time it was received, and the sender's userid and nodeid) is placed in a file named "FEEDBACK nodeid A5,".

The format of the FEEDBACK command is:

FEEDBACK **FB**	*text...*

text...
is the user's comments placed in the feedback file (FEEDBACK nodeid A5).

Usage Notes

1. Authorized VM users can retrieve the feedback file using thee GET FEEDBACK command (this does not include NCCF or NetView users).

2. The length of the message is limited by the maximum length of the command used to send the message to the programmable operator facility. If the user desires to send a longer message, the command must be used multiple times.

Example

The command issued at a virtual machine console:

M OP FEEDBACK SYSTEM RESPONSE WAS SLOW DURING THE MORNING SHIFT

The command issued at an NCCF or NetView operator terminal:

PROP FEEDBACK SYSTEM RESPONSE WAS SLOW DURING THE MORNING SHIFT

Response

Command complete

GET Command

Use the GET command to retrieve one of the programmable operator facility files; the feedback file (FB or FEEDBACK) or the log file (LOG). The file is sent to the requesting user if he is authorized in the active routing table to receive it. If LOG is specified, the user will receive the log file for either the current day or the specified day.

The format of the GET command is:

GET	{ **FEEDBACK** / **FB** / **LOG** [*yymmdd*] }

FEEDBACK or FB
indicates that the feedback file is to be retrieved.

LOG [*yymmdd*]
indicates that the log file for date "yymmdd" is to be retrieved. If no date is given, the log file for the current day is retrieved.

Usage Notes

1. An NCCF or Netview operator cannot use the GET command to retrieve the LOG and FEEDBACK files. Use CMS commands (and the programmable operator CMD command) to type the file(s) or portions of the files, or send the file(s) to some userid where you can process them.

2. The file appears in the requesting user's virtual reader in DISK DUMP format. The user must execute a DISK LOAD or RECEIVE command to read the file.

Examples

M OP GET FB

M OP GET LOG 861030

Response

Command complete

LGLOPR Command

Use the LGLOPR command to assign or release yourself as the logical operator for the programmable operator facility under which the command is executed. This programmable operator command can be used by a VM user or an NCCF or NetView operator.

Users are authorized in the active programmable operator routing table.

LGLOPR	{ **ASN** } { **RLS** } { **RPL** }

ASN
assigns the issuer of the command as the logical operator, if a logical operator is not currently assigned (i.e. the current LGLOPR is the default). If a logical operator is already assigned to the programmable operator facility, an error response is given.

RLS
releases the issuer from being the logical operator, if he currently is the logical operator and assigns the default LGLOPR. If the issuer is not the logical operator, no operation is performed and the system gives the following response:

```
DMSPOR763E Not currently assigned as LGLOPR,
cannot be released
```

RPL
replaces the current logical operator with the issuer of the command. The programmable operator determines if a logical operator is currently assigned. If there is, an implicit release is done. Then the issuer of the command becomes the logical operator.

Usage Notes

1. The system keeps the logical operator that is specified on the LGLOPR statement in the routing table as a default. You cannot release this logical operator. The LGLOPR ASN and LGLOPR RPL commands override the default.

2. To ensure that messages are not lost when changing logical operators, the new logical operator should issue a "LGLOPR RPL" command, rather than the current logical operator issuing a "LGLOPR RLS" command and the new logical operator issuing a "LGLOPR ASN" command.

3. If the default logical operator is the current logical operator and he issues the LGLOPR ASN command, CP will send a message stating that he already assigned as the logical operator

4. The HOSTCHK function is suspended when an NCCF or NetView operator or a local VM user is assigned as the logical operator. It is resumed when a remote VM user is assigned as the logical operator.

Examples

The command issued at a virtual machine console:

MSG OP LGLOPR ASN

The command issued at an NCCF or NetView operator terminal:

PROP LGLOPR ASN

Responses

Both the new and old logical operators receive the message:

DMSPOR758I {NCCF|VM} user *userid* [*nodeid*] is now LGLOPR for PROP on node *nodeid*

to notify them of the change of logical operators. The requester also receives the response:

Command complete

LOADTBL Command

Use the LOADTBL command to dynamically load a new routing table. You, as a VM user or an NCCF or NetView operator, can specify whether or not the currently assigned logical operator should be replaced by the logical operator specified in the new routing table.

LOADTBL	*[filename]* **[(RPL [ )]]**

filename
is the name of the routing table to be loaded. If filename is not given, the default routing table (filename = PROP) is used.

RPL
replaces the currently assigned logical operator with the logical operator specified in the new routing table. The logical operator is replaced only after the new routing table has been successfully loaded.

If RPL is not specified, the logical operator in the new routing table simply becomes the new default logical operator, and any explicitly assigned logical operator (i.e. a logical operator assigned by the LGLOPR command) remains the logical operator.

Usage Notes

1. If any action routines named in the specified routing table can not be located or loaded, an error message is issued, and the programmable operator facility drops any action routine modules associated with the specified routing table. It then tries to reload the action routine modules associated with the routing table that was active before the LOADTBL command was issued. If it cannot load these modules, the programmable operator facility terminates operation.

   Note: If any of the action routines associated with the previous routing table were modified (that is, replaced in LOADLIB) between the time the programmable operator facility dropped the specified routing table modules and reloaded the previously active routing table modules, the modified version of the action routines are used when the previous routing table is reloaded.

2. Only the filename is used to identify the routing table file to the LOADTBL command.

3. Because DMSPOL is the action routine module executing the LOADTBL command, it is not dropped and reloaded as are the other action routine modules listed in the routing table. If you want to replace this module, you must stop the programmable operator facility (using the STOP command), make the desired modifications or replacement, and invoke the programmable operator facility again as

described under "Invoking the Programmable Operator Facility" on page 322.

If the programmable operator virtual machine is set up so that the programmable operator is started automatically when CMS is IPLed in that virtual machine, it is sufficient to do the replacement and then IPL CMS again.

4. With the loading done by DMSPOL, it is possible for the other routines in DMSPOR to be replaced when a LOADTBL occurs. This permits changes to action routines other than DMSPOL to be made dynamically, without stopping the programmable operator. It is also possible to specify the name of a table to be loaded as a parameter to the action routine. The logical operator will be notified of the loading.

5. If the current logical operator is the default logical operator (not explicitly assigned), then the current logical operator will be replaced even if the RPL option is not specified.

6. When you issue LOADTBL and replace the logical operator, both the old and new logical operators receive the following message:

   `DMSPOR758I {NCCF|VM} user` *`userid`* `[`*`nodeid`*`] is now LGLOPR for PROP on node` *`nodeid`*

   Both operators receive this message, even if you have not specified the RPL option.

Example

The command issued at a virtual machine console:

MSG OP LOADTBL ROUTE3 (RPL

The command issued at an NCCF or NetView operator terminal:

PROP LOADTBL ROUTE3 (RPL

Responses

New RTABLE not loaded

PROP terminated

PROP running with routing table *fn ft fm*

LOG Command

Use the LOG command to write a message to the log file. Use of the LOG command allows messages to be placed in the log file with no action taken by the programmable operator facility.

The format of the LOG command is:

LOG	*text...*

text...
is the message text to be placed in the log file.

Usage Notes

1. All messages are logged whether or not the LOG command is explicitly used.

2. Authorized VM users can retrieve the log file using the GET LOG command (this does not include NCCF or NetView users). The log file has a fileid of "LGyymmdd nodeid A5" where "yy" is the year, "mm" is the month, "dd" is the day, and "nodeid" is the nodeid of the system on which the programmable operator facility is running.

   The userid and nodeid of the sender is recorded along with the message.

3. The length of the message is limited by the maximum length of the command used to send the message to the programmable operator facility. If you want to send a longer message, you must use the command multiple times.

Example

The command issued at a virtual machine console:

```
M OP LOG THIS MESSAGE IS TO BE LOGGED.
```

The command issued at an NCCF or NetView operator terminal:

```
PROP LOG THIS MESSAGE IS TO BE LOGGED.
```

Response

Command complete

QUERY Command

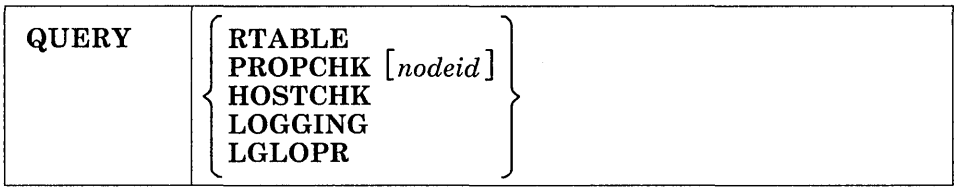
Use QUERY RTABLE to find the name of the active routing table.

Use QUERY PROPCHK and QUERY HOSTCHK to query the status of the programmable operator node-checking.

Use QUERY LOGGING to query the status of the logging messages.

Use QUERY LGLOPR to find the name of the currently assigned logical operator.

The format of the QUERY command is:



RTABLE
displays the name of the active routing table.

PROPCHK
displays a message with node-checking status. The message will state whether PROPCHK is set ON or OFF. If a nodeid is supplied, only the node specified is checked.

HOSTCHK
displays a message with node-checking status. The message will state whether HOSTCHK is set ON or OFF.

LOGGING
displays a message with logging status. The message will state whether messages and responses are being logged (LOGGING ALL), incoming messages and special programmable operator messages are being logged (LOGGING ON), or there is no log (LOGGING OFF).

LGLOPR
for a VM logical operator, displays a message with the userid and nodeid of the operator. For an NCCF or NetView logical operator, LGLOPR displays only the userid (i.e. operator-id) of the operator.

Example

The command issued at a virtual machine console:

M OP QUERY RTABLE

M OP QUERY PROPCHK NODE1

M OP QUERY HOSTCHK

M OP QUERY LOGGING

M OP QUERY LGLOPR

The command issued at an NCCF or NetView operator terminal:

PROP QUERY RTABLE

PROP QUERY PROPCHK VMSYS1

PROP QUERY HOSTCHK

PROP QUERY LOGGING

PROP QUERY LGLOPR

Responses

PROP running with routing table *fn ft fm*
: is received if the programmable operator facility is running.

  fn
  : is the filename of the active routing table.

  ft
  : is the filetype of the active routing table.

  fm
  : is the filemode of the active routing table.

{PROPCHK|HOSTCHK} is ON
: is received if node-checking is in effect.

{PROPCHK|HOSTCHK} is OFF
: is received if node-checking was specified in the RTABLE but is currently off.

PROPCHK is {ON|OFF} for nodeid *nodeid*
: is received if a specified nodeid is being queried.

DMSPOR762E Host checking is suspended -- LGLOPR not on a checkable node
: is received if the logical operator is an NCCF or NetView operator or a local VM user.

DMSPOR690E {PROPCHK|HOSTCHK} not specified in RTABLE
is received if node-checking was not specified in the current RTABLE.

DMSPOR709E PROPCHK not specified in RTABLE for nodeid *nodeid*
is received if a nodeid was specified on the QUERY command and node-checking was not specified in the current RTABLE for that node.

Logging ALL
is received if incoming messages and all programmable operator responses are being logged.

Logging ON
is received if incoming messages and special programmable operator responses are being logged.

Logging OFF
is received if no logging is being done.

DMSPOR758I {NCCF|VM} user *userid* {*nodeid*} is now LGLOPR for PROP on node *nodeid*
is received when the logical operator is being queried.

SET Command

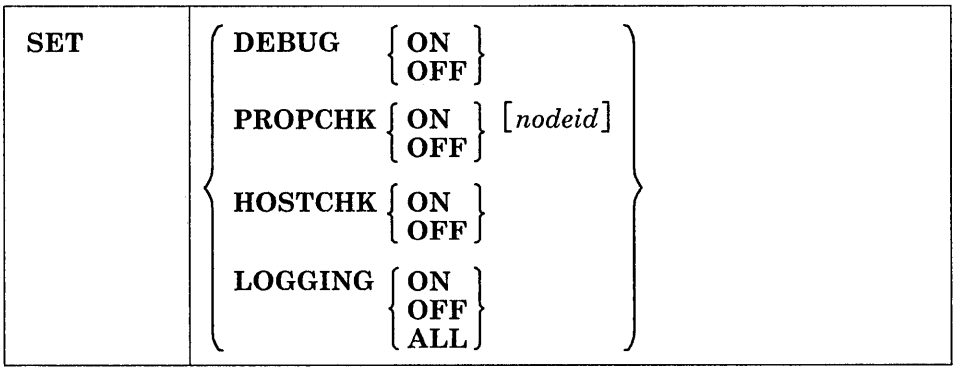
Use SET DEBUG to enter or exit the programmable operator facility DEBUG mode. DEBUG mode is used to do problem determination on the programmable operator facility.

Use SET PROPCHK to set the periodic checking of the programmable operator on the distributed systems ON or OFF. The distributed systems are identified by the PROPCHK statements in the routing table of the host programmable operator. The programmable operator facility with the PROPCHK statement (e.g. the host system) does the checking.

Use SET HOSTCHK to set the periodic checking of the link to the host system ON or OFF. HOSTCHK must be specified in the routing table of the programmable operator at the distributed system. The programmable operator facility with the HOSTCHK statement (e.g. the distributed system) does the checking.

Use SET LOGGING to control messages going to the programmable operator log file. SET LOGGING allows the message sender control the logging level: no logging, logging incoming messages and special programmable operator messages, or incoming messages plus response messages.

The format of the SET command is:



DEBUG {**ON** | **OFF**}

SET DEBUG ON stops the programmable operator facility from intercepting responses to CP commands. SET DEBUG OFF allows the programmable operator facility to return to its normal function of intercepting messages and responses from CP. SET DEBUG OFF is the initial setting.

PROPCHK {**ON** | **OFF**} [*nodeid*]

SET PROPCHK OFF halts checking of the programmable operators on the distributed systems until this command is reissued to set the checking back on, or until the programmable operator is stopped and restarted, or the PROP LOADTBL command is issued. SET PROPCHK ON restarts the checking. If nodeid is specified, PROPCHK applies to the specified node only. The initial setting is determined by the existence of the PROPCHK statement(s) in the RTABLE (ON if they exist, OFF if they do not).

An error message is received if the SET PROPCHK command is given and the routing table does not contain a PROPCHK statement or if the nodeid specified in the SET PROPCHK command is not found in a PROPCHK statement in the routing table.

HOSTCHK {**ON** | **OFF**}

SET HOSTCHK OFF halts checking of the host system by the distributed system. SET HOSTCHK ON restarts the checking. The initial setting is determined by the existence of a HOSTCHK statement in the RTABLE (ON if it exists, OFF if it does not.)

LOGGING {**ON** | **OFF** | **ALL**}

SET LOGGING OFF causes the programmable operator facility to stop writing any messages to the log file. SET LOGGING ON or ALL causes logging to be resumed. SET LOGGING ALL causes logging of all programmable operator command responses, such as virtual machine console I/O generated by action routines. The LOGGING statement in the configuration portion of the routing table determines the initial setting. If no LOGGING statement appears in the routing table, the default is ON.

Usage Notes

1. The SET DEBUG command is only valid from the console of the programmable operator facility virtual machine.

2. The SET DEBUG ON command permits an authorized user to stop the programmable operator facility from intercepting messages associated with CP commands entered from the console of the programmable operator facility virtual machine. The programmable operator facility responds to CP messages (MSG), warnings (WNG), and special messages (SMSG), and messages sent using the Single Console Image Facility (SCIF), but not to responses from CP.

3. When SET DEBUG OFF is in effect, all responses to CP commands are intercepted by the programmable operator facility. SET DEBUG OFF is in effect when the programmable operator facility is initialized.

4. If the logical operator is on a non-checkable node (for example an NCCF or NetView operator) when he turns HOSTCHK ON, the issuer of the SET command receives an error message.

Example

The command issued at the programmable operator virtual machine console:

```
SET DEBUG ON

SET DEBUG OFF
```

The command issued at a virtual machine console:

```
M OP SET HOSTCHK OFF

M OP SET PROPCHK ON SYS2

M OP SET LOGGING OFF
```

The command issued at an NCCF or NetView operator terminal:

```
PROP SET HOSTCHK OFF

PROP SET PROPCHK ON SYS2

PROP SET LOGGING OFF
```

Responses

For commands sent by message:

```
Command complete
```

For commands issued at the programmable operator console:

```
{PROPCHK|HOSTCHK} has been started

{PROPCHK|HOSTCHK} has been stopped

Logging has been started

Logging has been stopped

PROP running in DEBUG mode

PROP has exited DEBUG mode
```

STOP Command

Use the STOP command to stop the operation of the programmable operator facility. When the STOP command is issued, the programmable operator facility processes all outstanding messages, closes files, stops operation, and returns control to CMS. The STOP command is logged in the log file for the current day.

The format of the STOP command is:

STOP	

Usage Note

The STOP command is also valid from the console of the programmable operator facility virtual machine if an operator is logged on to the programmable operator facility virtual machine.

Response

PROP has terminated

Action Routines

Action routines are programs or EXECs that receive control in response to the match of a message and a routing table entry. They handle a particular type of message or command intercepted by the programmable operator facility. A set of action routines is provided with the programmable operator facility. These need no tailoring to provide you with the control and function needed to operate the programmable operator facility. You can extend the programmable operator facility by writing a new action routine and adding it to the appropriate routing table. Action routines can be EXECs or written in Basic Assembler Language. (Basic Assembler Language action routines must also be added to the PROPLIB LOADLIB. You can do this by invoking CMSGEND PROP.)

When the programmable operator facility invokes an action routine, it assumes that the originator of the message (the requester) is authorized to receive any responses or error messages issued by the action routine. Therefore, all VMCONIO responses (console I/O generated by LINEDIT, WRTERM, DMSERR, &TYPE, SAY, etc.) or error messages resulting from CP commands issued by the action routine are sent back to the requester. However, if the action routine uses the extended DIAGNOSE code X'08' to issue a CP command and receive the responses in a buffer, the programmable operator facility never sees these responses, and therefore, cannot send them to the requester.

An action routine is provided with the programmable operator facility that uses the extended DIAGNOSE code X'08' to issue a CP command and receive the responses in a buffer. This method ensures that the responses, error messages, or informational messages from the CP command are NOT presented to the programmable operator facility as IUCV message types 3, 4, or 7, respectively. Instead, the programmable operator will send the buffered messages or responses to the requester.

If an action routine abends, abend error messages are sent to the logical operator and the requester (if any). Control is returned to the point in the programmable operator facility immediately following the action routine call.

Note: Programs written in Basic Assembler Language can access the parameter list built by the programmable operator facility. The programmable operator parameters are available in a different fashion for EXEC action routines[19].

---

[19] EXECs may be written using the System Product Interpreter, EXEC 2, or CMS EXEC languages.

The Action Routine Interface

Action Routine Call Interface

Action routines are loaded by the programmable operator facility as CMS nucleus extensions. As a result, they must be invoked by the programmable operator facility as CMS commands via SVC 202. Also, addresses cannot be resolved between separate nucleus extensions; they must be passed dynamically if they are desired.

Action Routine Parameter Interface

An installation can write additional action routines in Basic Assembler Language. Action routines may also be written as EXECs. Programs written in Basic Assembler Language can access the parameter list built by the programmable operator facility. (The programmable operator parameters are available in a different fashion for EXEC action routines--see below.) The parameter list contains a list of addresses pointing to data that may be significant to the action routine invoked. The programmable operator facility then passes the address of the list as a parameter when it invokes the required action routine. See *VM/SP Data Areas and Control Block Logic Volume 2 (CMS)* for descriptions of the DSECTs mentioned below.

The register conventions used for invoking an action routine are:

- Register 1 points to a list of eight-byte tokens (CMS PLIST) containing the following information:

  TOKEN 1 Contains the command name (action routine name).

  TOKEN 2 Contains two fullwords. These fullwords contain the following:

  Fullword 1 - Contains the address of the PROP common area as described by the PROPCOM DSECT.

  Fullword 2 - Contains the address of a list of addresses that point to data that may be needed by the action routine. This list is described by the PARMLIST DSECT.

  TOKEN 3 Contains eight X'FF's to mark the end of the parameter list.

- Register 13 points to a standard OS eighteen word save area.

- Register 14 points to the address that receives control when the action routine completes processing, that is, the address to which the action routine must return control.

- Register 15 points to the action routine entry point and may be used as a base register.

Figure 36 offers a graphic representation of the previous discussion. It illustrates the data areas that can be accessed through Register 1.

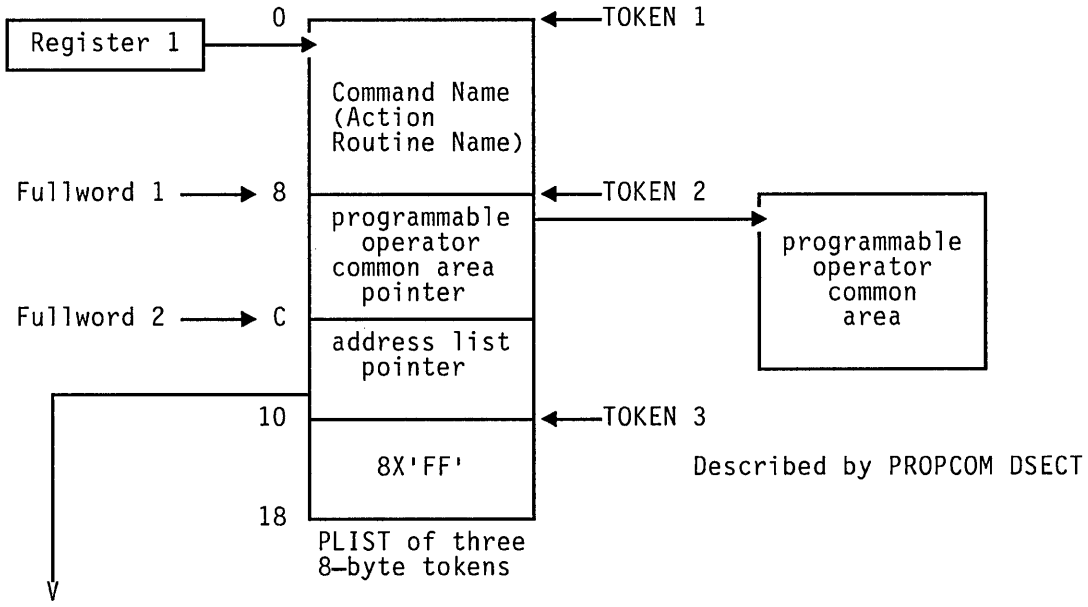
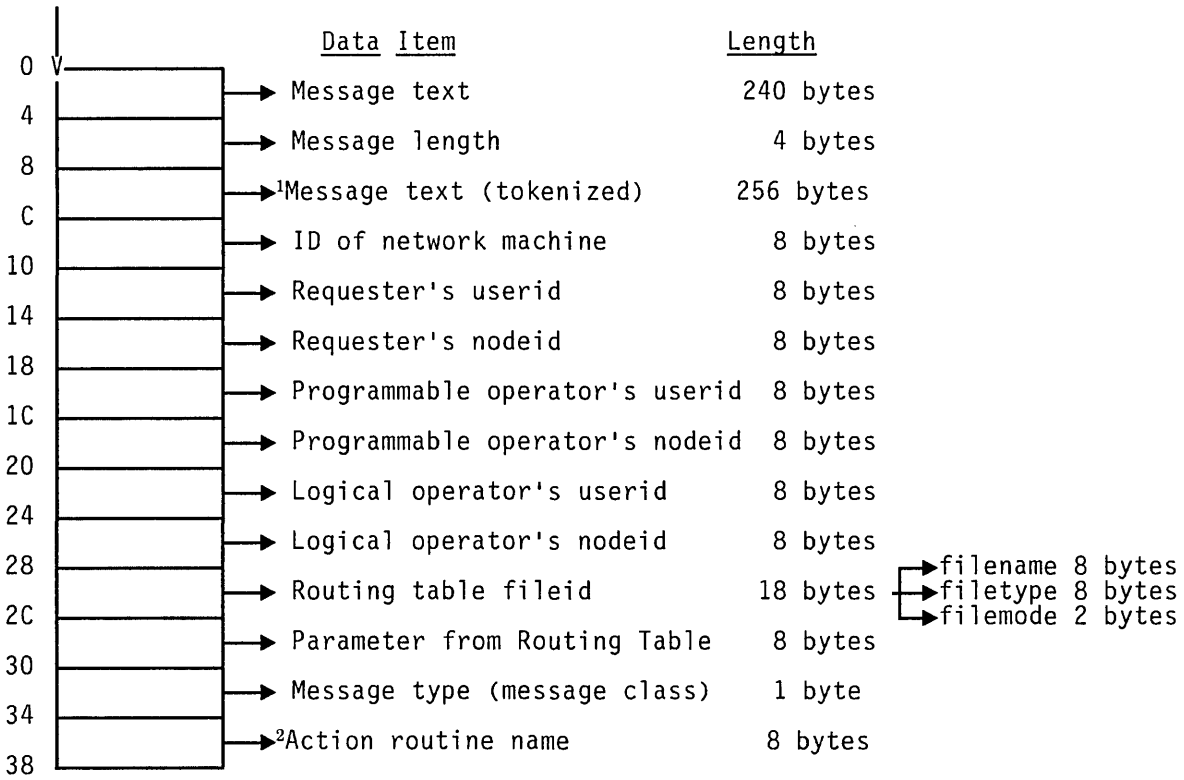


Figure 36 (Part 1 of 2). Register Conventions for Invoking an Action Routine



Address List described by PARMLIST DSECT

[1]In addition to the original message text, the message text is also provided in CMS tokenized form (eight-byte tokens followed by 8X'FF').

[2]The high order bit (X'80') of the last fullword of this list of addresses is set to one to indicate that it is the last entry in the list according to standard OS linkage conventions.

Figure 36 (Part 2 of 2). Register Conventions for Invoking an Action Routine

EXEC Action Routines

Action routines may also be written as EXECs[20]. The programmable operator parameters are available in a different fashion for EXEC action routines. The method is described below.

1. Having determined that the action routine is an EXEC, the programmable operator facility calls the action routine accordingly.

2. The following information is passed as parameters (arguments) on the EXEC invocation, in this order:

   - Requester's userid
   - Requester's nodeid
   - Logical operator's userid
   - Logical operator's nodeid
   - Message type code
   - The programmable operator facility's userid
   - The programmable operator facility's nodeid
   - Networking machine userid
   - RTABLE filename.

3. The following parameters are stacked LIFO for the EXEC in this order:

   - RTABLE PARAMETER field contents
   - Message text.

To ease handling by an EXEC, if the requester is CP, the requester's userid and nodeid are "CP".

Writing Action Routines

How an action routine is written using Basic Assembler Language depends on the function(s) that the action routine performs and the conditions under which it runs. Since the programmable operator can use message content and message origin to determine which action routine to call, it may not be necessary for the action routine to check any further conditions. However, by using the PARMLIST supplied by the programmable operator facility, the action routine may obtain additional information about the message. Each entry in the PARMLIST points to

---

[20] EXECs may be written using the System Product Interpreter, EXEC 2, or CMS EXEC languages.

some item of data about the message just received or about the programmable operator environment.

As described in the section, "Action Routine Parameter Interface" on page 387, the information initially provided to the action routine is in the form of a CMS tokenized PLIST. By loading the second fullword of the second token of that PLIST into a register, the user can establish addressability to the PROP PARMLIST. For example,

```
SAVE   (14,12)              SAVE REGISTERS
LR     R12,R15              LOAD BASE REGISTER
USING  ROUTINEX,R12         ESTABLISH ADDRESSABILITY
L      R2,12(,R1)           LOAD PROP PARMLIST ADDRESS
USING  PARMLIST,R2          PARM ADDRESSABILITY
```

These instructions would be sufficient for many action routines to establish addressability for the action routine and the PROP PARMLIST. The following instructions could then be used to obtain the addresses of the requester's (message originator's) userid and nodeid.

```
L      R4,PARMRUSR          GET REQUESTER'S USERID ADDRESS
L      R6,PARMRNOD          GET REQUESTER'S NODEID ADDRESS
```

The Programmable Operator DSECTs, such as PARMLIST, define the above labels. To include the PROP DSECT in the action routine insert the following assembler instruction in the source file for the routine:

```
COPY   PROP
```

In addition, it may be desirable to include the CMS REGEQU macro instruction for register equates. When the action routine is complete, it is necessary to restore registers and branch to the address in register 14, or use the OS RETURN macro.

Action Routine Error Message and Response Handling

It is sometimes necessary to issue error messages and/or responses from an action routine that are to be sent to the message originator (requester). To do this, the action routine should simply TYPE the messages/responses (via LINEDIT, DMSERR, or WRTERM for action routines written in BAL, and &TYPE or SAY for EXEC action routines). Upon completion of the action routine, the programmable operator facility collects these VMCONIO messages (IUCV type 5 messages), as well as any error messages (IUCV type 6 messages), and sends them back to the requester. To prevent the programmable operator facility from sending error messages from CP commands back to the requester, the action routine should be coded to use the extended DIAGNOSE code X'08' to obtain the responses in a buffer.

Handling Console I/O in an Action Routine

The installation must determine how an action routine is to handle console I/O generated by the virtual machine and CP. Normal operation of the programmable operator facility sets VMCONIO to IUCV (by default) before calling an action routine and sends the VMCONIO to the message originator (requester). If it is desired that console I/O (VMCONIO) produced by the action routine be typed on the programmable operator virtual machine console, the action routine must SET VMCONIO OFF. However, because there would not normally be an operator at the programmable operator virtual machine console, an installation can code an action routine to receive and handle VMCONIO instead of allowing the programmable operator to receive it and send it to the requester (message originator). To accomplish this, the action routine can receive the VMCONIO that was generated by using the IUCV RECEIVE function with message type 5 specified as the IUCV target class (TRGCLS). For details on using IUCV, refer to the section on IUCV earlier in this manual.

An example of this may be found in the subroutine CALLARTN of the IBM-supplied module DMSPOA. To use IUCV, it is necessary to include the CP COPY files, IPARML and EQU.

CP generated console I/O (CPCONIO) should be handled differently than above. The CPCONIO setting should not be changed because this could cause the programmable operator facility to miss some asynchronous CP messages.

If the action routine is to receive the responses from CP commands that it issues, it should use the DIAGNOSE code X'08' support with a command response buffer, rather than trying to receive it with IUCV. (See "DIAGNOSE Code X'08' -- Virtual Console Function" on page 9.) The reason for this is that other CP messages can be mixed in with the command response, and therefore the program cannot be assured of receiving its response in consecutive IUCV messages.

If the CP command response is to be typed on the programmable operator's virtual machine console, the action routine should use a CMS function, such as WRTERM, to write the lines in the program's CP command response buffer to the terminal.

Description of Supplied Action Routines

The action routines supplied with the programmable operator facility are DMSPOR, DMSPOS, and DMSPOL. A parameter must be supplied for module DMSPOR. This parameter is the name of the function or action that DMSPOR is to perform. DMSPOS may be invoked along with a parameter, which, in this case, is a userid or nickname. DMSPOL may be invoked with a parameter, which is a routing table name.

Note that new action routines are not required to be in this format. The programmable operator facility supports any desired number of action

routines. Each one is loaded separately when the programmable operator facility is initialized, or when a LOADTBL command is issued.

The following sections describe the action routines that are supplied with the programmable operator facility. These action routines (or subroutines in the case of DMSPOR) correspond to the programmable operator commands described later in this chapter.

DMSPOR - Miscellaneous supplied action routines

GET - Send the indicated file to an authorized user

This routine sends programmable operator files, such as log and feedback files, to requesting user. The files are sent using the CMS DISK DUMP command.

LGLOPR - Process the LGLOPR command

This routine processes the LGLOPR command to assign (ASN), release (RLS), or replace (RPL) the logical operator as specified by an authorized user of the LGLOPR command.

QUERY - Return a response to a user query

This routine returns the fileid of the currently active routing table, returns the userid of the current logical operator, or returns the status of programmable operator node-checking or logging to the user who issued the command.

SET - Change the status of specific functions

This routine stops or resumes the periodic checking of the distributed systems or the host system, or the logging of messages in the log file.

STOP - Stop the programmable operator facility

This routine stops the programmable operator operation after processing currently queued messages. The programmable operator virtual machine returns control to CMS.

TOFB - Write a message to the feedback file

This routine attaches the date and time received to the head of the incoming message and writes it to the feedback file. See "The Feedback File" on page 358 for more information.

TOVM - Execute a CP/CMS command

This routine is invoked when the programmable operator CMD command is issued. The text following "CMD" is regarded as the CP or CMS command to be executed in the programmable operator virtual machine. The command is treated as if it were entered at a CMS console (that is, such things as synonyms and the IMPCP and IMPEX

settings apply to its interpretation as supported by the CMS COMMAND SUBCOMM environment). The response to the executed CP or CMS command is returned to the authorized user who invoked the CMD command.

Authorized users of the CMD command should be aware of the following:

- Issuing commands that alter or overlay CMS storage, such as CP DEFINE STORAGE, CP IPL CMS, CP SHUTDOWN, and so on, has an adverse effect on the operation of the programmable operator facility.
- Reissuing the PROP command once the programmable operator facility is running causes the programmable operator facility to stop operating correctly. The user must re-IPL CMS and restart the programmable operator facility using the procedure described under "Invoking the Programmable Operator Facility" on page 322.
- Issuing commands that cause a VM READ or CP READ (interactive commands such as the DDR command) stop the operation of the programmable operator facility. The programmable operator facility must then be restarted in the manner described under "Invoking the Programmable Operator Facility" on page 322.
- Line editing characters (pound sign (#), for example), as defined by the CP TERMINAL command, are not recognized as line editing characters by the programmable operator facility.
- The CMS immediate commands (e.g. HB, HI, HO, HT, HX, RO, RT, SO, TE, and TS) are not recognized by the programmable operator facility. If a user issues any of these commands, he receives an "Unknown CP/CMS command" response from the programmable operator facility.

In general, the programmable operator facility does no checking to ensure or prevent any of the above circumstances from occurring.

DMSPOS - Route a message

DMSPOS sends (routes) a message to the user specified in the RTABLE PARAMETER field. The user is identified by a nickname from the CMS userid NAMES file or by a userid. If the user is on another system, identification must be through a nickname. LGLOPR may be specified in the PARAMETER field of the routing table, which would indicate that DMSPOS uses the value of the currently assigned logical operator. If no logical operator has been explicitly assigned, DMSPOS uses the value specified in the LGLOPR statement in the routing table. This is the default if the parameter field is left blank.

A message longer than 94 characters (including the 19-character programmable operator origin id) is split and sent as multiple messages.

The first piece is no more than 94 characters. The remaining pieces are no longer than 91 characters, and preceded by a continuation mark (".. "). This splitting ensures that the message is small enough to be sent through an RSCS network.

If an error occurs because of an invalid target id, for example, the nickname was not in the "userid NAMES" file, the programmable operator tries to send the message to the logical operator.

Messages are sent with the CMS TELL command. If the programmable operator virtual machine is authorized (class B), the CP MSGNOH command is used. If the virtual machine is not authorized to use the CP MSGNOH command, then the CP MESSAGE command is used. For more information on the CMS TELL command, see the *VM/SP CMS Command Reference.*

Notes:

1. *Using the CMS TELL command requires the user to have a SYSTEM NETID file set up.*

2. *DMSPOS must not be invoked if the logical operator virtual machine is the same as the programmable operator virtual machine. Also, a parameter should not be specified that directs the message to the programmable operator virtual machine.*

To prevent a LOOP condition, a message being handled may not be sent on to the routing target by the DMSPOS routine. A message is not sent if it falls into any one of the following categories:

1. The preceding message could not be sent, and the current message is the same as the preceding message. In other words, the programmable operator receives an error return code when trying to send consecutive identical messages.

2. The programmable operator tries to route a message that originated from the networking virtual machine on the programmable operator's node. The message is identical to the last message from that virtual machine that the programmable operator tried to route. For example, a local network machine detects that a link is down.

3. The programmable operator tries to route a message that originated from the networking virtual machine on another system (node), and that message is identical to the last message that the programmable operator tried to route. For example, messages are being routed from system A to the logical operator who is supposed to be on another system (B), but is not logged on. The networking virtual machine on system B sends the programmable operator an error message each time it tries to route a message to the logical operator. This could cause a loop if not detected.

If the DMSPOS routine tries to send a message to the logical operator, but for some reason the logical operator's network node is unavailable for

messages (not logged on, SMSG and/or MSG off), DMSPOS detects this condition and stops any further attempts to send that message. The unsent message, although logged in the current day's log file, is not displayed at the logical operator's console.

DMSPOL - Load a routing table

This routine dynamically loads the routing table indicated by the programmable operator LOADTBL command. The routing table name must be "filename RTABLE", where "filename" can be any name that conforms to CMS file naming conventions. Although the routing table name specified with the LOADTBL command takes precedence, it is also possible to specify in a routing table the filename of the table to be loaded as a parameter to the action routine. (This can be used as a default.) Therefore, any message selected by the system programmer can cause a new RTABLE to be loaded. Also, the programmer can change the LOADTBL default of "PROP" to whatever is desired without changing the LOADTBL action routine.

Note: With the loading of the routing table done by a separate action routine, it is possible for the other routines, DMSPOR and DMSPOS and any user-written routines, to be replaced when a LOADTBL occurs. This permits changes to action routines other than DMSPOL to be made dynamically without stopping the programmable operator.

Exit EXECs

Exit EXEC Interface

The programmable operator facility exit EXECs have the same parameter list provided as an EXEC action routine, with the exception that no RTABLE parameter field value and no message text are stacked for the EXEC. When an exit EXEC is called, contents of the program stack depend upon which exit is being invoked. Descriptions of the stack contents for the different types of exits follow:

Notes:

1. *Some of the parameter values have no meaning for a particular exit EXEC and their use is left to the discretion of the EXEC writer. For example, the requester's userid and nodeid have no meaning for the communication error EXECs, PROPPCHK and PROPHCHK.*

2. *The programmable operator facility does not trap VMCONIO-type or CP EMSGs produced by exit EXECs as it does for action routines.*

Supplied Error Exit EXECs

Communication Error Exit

The PROPPCHK EXEC is invoked when the programmable operator facility determines that communication with a node that is being checked has changed status. When this occurs, the following information is stacked, LIFO, for the EXEC.

1. Entries having the format

   `"nodeid UP"  or  "nodeid DOWN"`

   where:

   nodeid is the RSCS nodeid of a node that has changed communication status.

   UP indicates that the node had not been responding and has resumed responding to acknowledgement requests.

   DOWN indicates that the node had been responding and has ceased responding.

2. Total number of nodeid entries stacked.

The PROPHCHK EXEC is invoked when the programmable operator determines that communication with the logical operator node (if it is being checked) has changed status. If the status has changed, a line is stacked LIFO for the EXEC. The line is either "nodeid UP" or "nodeid DOWN", where "nodeid" is the RSCS nodeid of the logical operator and "UP" and "DOWN" have the same meaning as for PROPPCHK.

LOG Error Exit

If a virtual machine resource limit is reached, such as "disk-full", it may not be possible to write another record to the programmable operator facility log file. If this happens, a user-written EXEC is invoked to perform whatever recovery action the user thinks is desirable or necessary. The user EXEC must have the filename of PROPLGER. The programmable operator facility stacks (LIFO) the error code received from the CMS FSWRITE function. The programmable operator performs the following actions depending on the return code from the EXEC:

RC = 0 recovered from error. The programmable operator facility should retry logging. If it is still unable to log, an error message is sent.

RC = 4 unable to do recovery. The programmable operator facility should send an error message.

The error message is sent to the logical operator. If the PROPLGER EXEC cannot be found, the programmable operator facility acts as if RC = 4 has

been returned thus, an error message is sent to the logical operator. Whatever action is taken, the programmable operator facility continues operation. The IBM-supplied sample PROPLGER EXEC:

- Closes the current log file
- Sends the last two log files to the logical operator
- Erases the last two log files.

Note: PROPLGER does not try to send the log files to the logical operator if the logical operator is an NCCF or NetView operator. The files are instead sent to the programmable operator's virtual reader. You can change the EXEC to have the files sent elsewhere, if you wish.

If the same logging error occurs on two successive logging attempts, (for example, two consecutive incoming messages cause the same logging error) the programmable operator sets LOGGING to "OFF". This prevents unpredictable looping in some situations. Note, though, that the logical operator may receive only two error messages when logging errors occur.

Problem Determination - Debug Mode

Debug mode determines problems with the programmable operator facility itself. It allows responses to commands issued from the programmable operator virtual machine console to be returned back to the console without being intercepted by the programmable operator facility. This permits any CP command (for example, CP TRACE and ADSTOP commands), to be issued without having its response trapped by the programmable operator facility.

SET DEBUG ON may be used after the programmable operator facility responds with the message:

```
PROP running - enter STOP to terminate
```

indicating that the programmable operator facility is running and operational. The programmable operator facility then responds with the message:

```
PROP is running in DEBUG mode
```

which is also written to the log file. Once in debug mode, the programmable operator facility waits to receive messages from another virtual machine, or for the system programmer to enter input from the console. Because only two commands, STOP and SET are accepted from the programmable operator virtual machine console, the system programmer must enter the CP environment (using the PA1 key) to issue any CP commands. Otherwise, the commands are intercepted and rejected as invalid programmable operator commands.

Pressing the PA1 key or issuing the BEGIN command returns control to the programmable operator facility. From this environment, issuing SET DEBUG OFF returns the programmable operator facility to its normal function of trapping messages.

Chapter 16. Getting National Languages on Your System

VM is shipped with a *system national language* of American English. The system national language is automatically set for all virtual machines once VM is completely installed from the product tape -- when users log on, they receive messages, see panels, and can enter CMS commands in American English.

You can also order other languages for your VM system. When you order another language, you receive a feature tape with files for that language on it. These files contain translated information that let users interact with VM in this new language. Unique VM/SP HPO messages are translated. However, unique VM/SP HPO HELP files for commands and messages are not translated.

All VM languages are identified by a 1-to-5 character *langid*. The langid for American English, the default language, is "AMENG." Here are the langids for the default language and the other languages you can order:

Langid	Language
AMENG	American English
KANJI	Kanji
UCENG	Upper Case English
FRANC	French
PORTG	Brazilian Portuguese
GER	German

Contents of the Feature Tape

The feature tape you receive for a language contains three types of files:

- Source files
- Object code files
- Listing files.

Source Files and Listing Files

The language source files are in external form -- they cannot be read internally, but they can be edited. These source files then are converted into object files that can be read internally.

Fileid	Description
DMKMES[*y*] REPOS	CP message repository. This contains the translated versions of CP system messages.
DMKMES[*y*] LISTING	This is the listing file produced when compiling the CP message repository.
DMSMES[*y*] REPOS	CMS message repository. This contains the translated versions of CMS system messages.
DMSMES[*y*] LISTING	This is the listing file produced when compiling the CMS message repository.
DMSSPA[*y*] DLCS	CMS system command syntax definition file. This contains the translated syntax definitions for CMS commands.
DMSTRT[*y*] ASSEMBLE	CMS uppercase translate table. This maps lowercase alphabetic characters to uppercase for the language. (DMSTRT can be modified to display certain characters in special situations.)
CSIMES[*y*] REPOS	GCS message repository. This contains GCS messages (not translated for Release 5).
CSIMES[*y*] LISTING	This is the listing file produced when compiling the GCS message repository.
ATSCMR[*y*] REPOS	TSAF message repository. This contains TSAF messages (not translated for Release 5).
ATSCMR[*y*] LISTING	This is the listing file produced when compiling the TSAF message repository.
xxxxxxxx HELP*cccc*	HELP files, where *xxxxxxxx* is the command name and *cccc* is the VM component name (CP, CMS, etc.) The HELP files always have the same fileids regardless of the language. They reside on a different disk for each language.

Note: Not all Release 5 HELP files are translated. Also, VM/SP HPO-unique HELP files for commands and messages are not translated.

y is a unique character (or two-character string) that corresponds to a particular langid and language. The possible values for this "country code" are shown in the following table.

y	langid	language represented
A	KANJI	Kanji
B	UCENG	Uppercase English
C	PORTG	Brazilian Portuguese
D	FRANC	French
E	GER	German

These country code values are stored internally in a file called VMFNLS LANGLIST.

Note: The American English versions of translatable source files have a six-character filename; they do not use the country code.

Object Files

The language object files have been converted to internal form. These files get loaded into the CP, CMS, or GCS nucleus during the installation procedure.

(The text file produced for the TSAF message repository is loaded into the TSAF virtual machine storage using the SET LANGUAGE command.)

Fileid	Description
DMKMES TXT*langid*	CP message repository, compiled by GENMSG
DMSMES TXT*langid*	CMS message repository, compiled by GENMSG
DMSSPA TXT*langid*	CMS system command syntax definition file, created by CONVERT COMMANDS
DMSSSY TXT*langid*	CMS system national language translation and synonym table, created by CONVERT COMMANDS
DMSTRT TXT*langid*	CMS uppercase translate table, created by VMFASM
CSIMES TXT*langid*	GCS message repository, created by GENMSG
ATSUME TXT*langid*	TSAF message repository, created by GENMSG.

Installing National Language Files on Your VM System

You must decide which language is to be the system national language (instead of American English), or if you just want to make a certain language available as an option to users.

If you want a language other than American English to be the system national language, see "Installing a New System National Language" in the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide.* However, if you want a new language available for people to use but you don't want it to be the system national language, you must save all language information as described below.

Note: Some languages have character sets that require special hardware. Be sure that all display devices in your configuration can properly display the character set of any new language you add to your system.

Loading the National Language Files From Tape to Disk

The first thing you must do is load each language file from the feature tape to the proper minidisk. Use the VMFPLC2 command or the ITASK EXEC to do this. (See the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide* for more information on VMFPLC2 and ITASK.)

Saving National Language Files for CP and CMS

For each additional language you want to install on your VM system, you must perform some tasks to make the language files available to users. (See the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide* for the tasks you must perform to change your system national language.) Here is an overview of these steps:

1. Determine how much DASD space you need for the CP message repository.
2. Specify DASD space for the CP repository.
3. Create a DCSS to store the CMS language files.
4. Regenerate the CP nucleus.
5. Save CP language files on DASD and CMS language files in a DCSS.
6. Update users' directory entries.

Determine DASD space for CP Repository: If you plan to have more languages than American English available on your VM system, you must allocate DASD space for CP message repositories.

Each language has its own repository file for CP messages. The feature tape for a language has three forms of this repository: a source file, a compiled object file, and a listing file. You must reserve DASD space for the compiled object file. The last few lines of the listing file indicates the number of 4K pages required to save this repository.

Specify DASD space for the CP repository: Edit the DMKSNT ASSEMBLE file using the VM System Product Editor (XEDIT) and code a NAMELANG macro. The LANGID operand on this macro specifies the language of the repository you are saving. The NLSPGCT operatnd of this macro specifies the number of 4K pages to be reserved on DASD for the repository.

See the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference* for a detailed discussion of the NAMELANG macro.

Create a DCSS to store the CMS language files

1. Determine the size of the contents of the CMS repository

   a. Make a control file for the LANGMERG command that contains information for CMS language files. Name this file DMS*langid* LANGMCTL. Here is a sample:

```
*
* This is the LANGMERG control file for langid in CMS.
* The fileid of this control file is DMSlangid LANGMCTL.
*
DISK 19D
ETMODE OFF
MESSAGE  DMSMES TXTlangid *
PARSERS  DMSSPA TXTlangid *
SYNONYMS DMSSSY TXTlangid *
TRTABLES DMSTRT TXTlangid *
USER     DMSUSE TXTlangid *
```

   b. Issue LANGMERG to create a single text deck from all CMS language files.

```
LANGMERG langid DMS
```

      This text deck has the fileid DMSNLS TXT*langid*. A second file is also created with the fileid DMS*langid* LANGMAP. In this file, add the last length to the last starting location listed to find the size of the CMS segment (SYSPGCT). Now, given this size, you should determine the following:

      - DASD starting location (SYSSTRT)
      - Starting segment addresses in virtual storage (SYSHRSG)
      - Page addresses in virtual storage corresponding to the segment address (SYSPGNM).

      See the NAMESYS macro in the *VM/SP Planning Guide and Reference* or in the *VM/SP HPO Planning Guide and Reference* for more information about determining these values.

2. Edit the DMKSNT ASSEMBLE file using XEDIT and code a NAMESYS macro. The SYSNAME operand on this macro, which names the DCSS, must be in this format:

```
SYSNAME=NLxy.
```

where:

x
Is the *levelid*, which describes the particular version of a language DCSS. The levelid is a single alphameric character.

y
Describes the *langid*, which is the identifier for the language whose files you are saving.

Note: Be sure to use the same *langid* on NAMESYS that you specified on NAMELANG.

See the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference* for a detailed discussion of the NAMESYS macro and DCSSs.

Regenerate the CP nucleus: After assembling the modified DMKSNT module, you should regenerate the CP nucleus to get the new System Name Table (DMKSNT) into effect. Refer to the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide* for instructions on how to do this.

Save CP files on DASD and CMS files in a DCSS: First, you have to make sure **all users are logged off** the system.

1. Make a control file for the LANGGEN command that shows what object CP message repository and what CMS language files will be saved.

   ```
   DMKMES TXTlangid *
   DMSNLS TXTlangid A
   ```

   Name this file NL*xy* LANGGCTL where *x* is the *levelid* and *y* is the *langid*.

2. Make sure you have a current copy of the SYSTEM LANGUAGE file on your A-disk. If you don't, copy the SYSTEM LANGUAGE file from the system disk to your A-disk, if one exists. If no SYSTEM LANGUAGE file exists, LANGGEN creates one on the first read/write disk you have accessed.

3. Issue LANGGEN for this language, which puts the CP message repository into the DASD space named by NAMELANG, and puts the CMS text deck in the DCSS named in NAMESYS.

   ```
   LANGGEN langid
   ```

4. If you have other language files you want to save, repeat steps 1a, 1b, 1, and 3.

5. When you are finished, copy the SYSTEM LANGUAGE file from your A-disk back to the system disk. Note that you need read/write access to the system disk to perform this step. If this is not possible, have the system administrator copy the file to the system disk for you.

(See "The LANGMERG Command" on page 411 and "The LANGGEN Command" on page 413 for detailed explanations of the LANGMERG and LANGGEN commands.)

Update User's Directory Entries: You must update the directory for anyone who wants this new language as a system default when they log on.

All users' virtual machines are set to the system national language during logon, unless their directory contains an "OPTION LANG *langid*" statement that overrides the system national language.

If you want to change such a user's default language, you must do one of two things:

1. If the user was set to the system national language, add an "OPTION LANG *langid*" statement to that user's directory. Or,

2. If the user was set to a different default language, change the *langid* in the directory statement "OPTION LANG *langid*."

Users can now interact with VM in this new language. The language information is either automatically available to the users when they log on, or available by using the SET LANGUAGE command during a CMS session.

Saving National Language Files for GCS

If you want GCS to use a particular language to issue messages, you must build GCS with the appropriate text decks using VMFLOAD.

Adding National Language Information for an Application

VM is shipped with a single "application", CMS. However, you can add other applications to VM; these may be system applications, such as GCS or TSAF, or your own applications, such as an accounting package.

The application you add may include message files or command syntax files. If it does, and if you want the application to be available in another language already on your system, you must follow these steps:

1. Create an identifier for the application.
2. Ensure that the files for the new application can fit in the DCSS for a language.
3. Compile source message or command syntax files into object code.
4. Create a control file for the LANGMERG command, and then issue LANGMERG to create a single text deck from the application files.
5. Create a control file for the LANGGEN command, and then issue LANGGEN to save the application text deck in a language DCSS.

You must repeat these steps once for every language on the system that will support the new application.

Create an identifier for the application: You must give the application a 3-character identifier, called an *applid*. You will use this *applid* when issuing commands to save the application's files for a language. The *applid* for CMS is "DMS."

Check the size of the language DCSS: You must ensure that the application's files can fit in the language DCSS. (The size of a DCSS is defined on the NAMESYS macro in CP's System Name Table, DMKSNT.) If a DCSS is not large enough to accommodate the new application, you must follow these steps:

1. Change the NAMESYS macro that is coded for the language DCSS in CP's System Name Table;
2. Regenerate the CP nucleus; and
3. Re-IPL the system to activate the new System Name Table.

Compile the Message Repository: If the application has a source message repository file, issue GENMSG to convert it into a machine-readable object file:

```
GENMSG fn ft fm applid langid (options
```

(See the *VM/SP CMS Command Reference* for a detailed explanation of GENMSG.)

Compile the Command Syntax File: If the application has a source command syntax file, issue CONVERT COMMANDS to convert it into a machine-readable object file:

```
CONVERT COMMANDS fn ft fm (options
```

(See the *VM/SP CMS Command Reference* for a detailed explanation of CONVERT COMMANDS.)

Create a LANGMERG Control File: This file identifies the language files for this application. You should name this file "*ay* LANGMCTL" where *a* is the *applid* and *y* is the *langid* for the language files. Here is a sample LANGMERG control file for an application that has a message file and a command syntax file:

```
*
* This is the LANGMERG control file for
*  langid in applid.
* The fileid of this control file is
*  ay LANGMCTL.
*  where a is the applid and
*  y is the langid for the language files.
*
DISK address
ETMODE OFF
MESSAGE  applidMES TXTlangid *
PARSERS  applidSPA TXTlangid *
```

Issue the LANGMERG command: LANGMERG makes one text deck from all the application's language files:

```
LANGMERG langid applid
```

See "The LANGMERG Command" on page 411 for a complete discussion of LANGMERG.

Update the LANGGEN control file: A LANGGEN control file should exist for every language that is supported on your system. (The default identifier is "NL*xy* LANGGCTL," where *x* is the *levelid* and *y* is the *langid* for the file.) This file contains the language text decks for all applications. You must add an entry for the new application's text deck; it should be in the format "*applid* TXT*langid*." Here is an example of a text deck for a language that has CMS and one other application:

```
DMSNLS TXTlangid A
applidNLS TXTlangid A
```

Issue the LANGGEN command: LANGGEN saves the application's text deck in the language DCSS. This new version of the DCSS attaches to a user's virtual machine when that user issues a SET LANGUAGE command:

```
LANGGEN langid
```

See "The LANGGEN Command" on page 413 for a complete discussion of LANGGEN.

Updating Files for an Existing National Language

When CMS or a CMS application using NLS has changed some message or command syntax information, you will have to update the language DCSS. Or, if some message information has changed for CP, you will have to update the repository that is on DASD.

1. For message repositories and command syntax definition files, issue the VMFNLS EXEC.

   ```
   VMFNLS fn ft ctrlfile (options
   ```

   (See the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide* for a detailed explanation of VMFNLS.)

2. Issue a LANGMERG command, with the appropriate control file, to get the changed file(s) into the merged object file in machine-readable format.

3. Issue a LANGGEN command, with the appropriate control file, to get both the updated CP message repository (which is on DASD) and the updated language DCSS saved.

4. Re-IPL the system. CP must be re-IPLed after resaving the repository.

Deleting a National Language

If you have a national language available on your system, but decide that you no longer want the language available to users, you must follow these steps:

1. Edit (with XEDIT) DMKSNT ASSEMBLE and delete the NAMESYS and NAMELANG entries for the *langid* you want to remove.
2. Regenerate the CP nucleus to get the new DMKSNT ASSEMBLE file into effect.
3. Update the directory entry of any user who had this language as their logon default. If this is not done, a user receives this error message when logging on:

   ```
   DMKLOH365E   Requested language langid1 is unavailable.
                Language langid2 set.  RC=rc
   ```

   The system national language is set for the user's virtual machine.
4. Re-IPL the system.

You must also delete language information for any application (other than CMS) that was available in that language.

Deleting Language Information for an Application

If you decide that you no longer want to have language information for an application in a language DCSS, here are some steps you must perform: (you must do this once for each application)

1. Delete the entry for the application in the LANGGEN control file of the language.
2. Issue the LANGGEN command for the language that is getting this application deleted.
3. If any users no longer want this language as their default, update their directory entry.

You must perform this sequence of steps when you delete language information for one application. If you are deleting information for an entire language, you must go through these steps once for every application that uses that language.

The LANGMERG Command

Use LANGMERG to combine all the language-related files for an application into one text file. The LANGGEN command can then load this single text file into a DCSS as a language segment.

You should use LANGMERG for two purposes:

1. When you have changed something in a language file and you want to recombine all the updated language information for an application.
2. When you are building a language text deck for an application.

The format of LANGMERG is:

LANGMERG	langid applid [(CTL filename [)]]

where:

langid
: represents the language whose text file is being created. A langid may be 1-to-5 characters in length and must be made up of only CMS file system characters.

applid
: represents the application whose text file is being created. This must be three characters long.

CTL filename
: specifies the filename of a control file that is used instead of the default *ay* (where *a* is the *applid* and *y* is the *langid* for the language files). The filetype of this control file must be LANGMCTL.

  See below for more information about LANGMERG's control file.

Notes:

1. *The single text deck created by LANGMERG has the fileid applidNLS TXTlangid. You can then use the LANGGEN command to save this text deck in a DCSS.*

2. *When you want to use a different language file, you must either*

   a. *Edit the default LANGMERG control file, or*
   b. *Make your own control file and specify it as an option when you invoke LANGMERG.*

3. *LANGMERG creates a map on your A-disk which shows where language information is stored within a text deck. The fileid of this map is ay LANGMAP where a is the applid and y is the langid for the map.*

4. *The text files produced by CONVERT COMMANDS may be empty. LANGMERG detects this condition and notes it in the LANGMAP file. This normal condition indicates that no entries were produced for the file.*

LANGMERG's Control File

LANGMERG's control file supplies information about the language and identifies which files are to be loaded. You have to create a LANGMERG control file to issue the LANGMERG command.

The LANGMERG control file may contain the following types of records:

1. Comment - "*"

   Anything may follow the * on a comment record. The asterisk must be the first character on the record.

2. A DISK record, which identifies the address of a disk that is associated with the language.

   ```
   DISK vaddr
   ```

   For CMS, the DISK record identifies the HELP disk; other applications can use this record for different purposes.

3. An ETMODE record, which identifies whether the language named is a double-byte character set (DBCS) language.

   ```
   ETMODE ON|OFF
   ```

   ETMODE should be ON if the language is a DBCS language. If this record is omitted in the control file, OFF is assumed.

4. Records which identify the fileids of language files in the language control block:

   ```
   keyword [fileid]
   ```

   Valid keywords are:

MESSAGE	for a system message repository
PARSERS	for system command syntax definition files
SYNONYMS	for system national language synonym and translation file
TRTABLES	for translate tables
USER	for your installation's use.

   "fileid" is a filename, and optionally a filetype and filemode, that identifies from where the information is to be read. The filenames should be in this format:

*applid*MES	for a system message repository
*applid*SPA	for system command syntax definition files

	*applid*SSY	for a national language system synonym and translation file
	*applid*TRT	for translate tables
	*applid*USE	for your installation's use.

The default filetype is TXT*langid*, and the default filemode is *.

Example: Use the following LANGMERG control file for CMS in American English (applid for CMS is "DMS"; langid for American English is "AMENG"):

```
*
* This is the LANGMERG control file for American English in CMS.
* The fileid of this control file is DMSAMENG LANGMCTL.
*
DISK 19D
ETMODE OFF
MESSAGE  DMSMES TXTAMENG *
PARSERS  DMSSPA TXTAMENG *
SYNONYMS DMSSSY TXTAMENG *
TRTABLES DMSTRT TXTAMENG *
USER     DMSUSE TXTAMENG *
```

The order of the records in the LANGMERG control file determines where the application's files will go in the language DCSS. You may be able to improve your system's performance by changing the order of these files.

The LANGGEN Command

Use LANGGEN when adding a language to your system, adding an application to a language, or regenerating a language after a language-related file has been changed.

The LANGGEN command gets all the text files created by LANGMERG for a language and saves them in a DCSS named NL*xy*, where *x* is the *levelid* and *y* is the *langid* for the DCSS.

LANGGEN also saves CP's message repository on DASD.

The format of LANGGEN is as follows:

LANGGEN	langid [levelid] [(CTL filename [)]]

where:

langid
: Represents the language whose files you are saving in a DCSS. A langid may be one-to-five characters in length and must be made up of only CMS file system characters.

levelid
: is one character (A-Z,0-9) that identifies the version of the DCSS being built. It is used as the third character of the DCSS name after "NL." If omitted, it defaults to "S."

CTL filename
: specifies the filename of a control file to be used instead of the default of NL*xy*, where *x* is the *levelid* and *y* is the *langid* for the file. The filetype of this control file must be LANGGCTL.

  (See below for more information about LANGGEN's control file.)

Notes:

1. *You must have the system national language set on your virtual machine to use the LANGGEN command. The virtual machine must be large enough to include the segment being saved.*

2. *You can use the* **levelid** *operand of LANGGEN when you want to have more than one version of a language DCSS. For instance, you may have an American English DCSS called* NLSAMENG *that contains everything you want; however, you want to add some things to the DCSS while preserving your original copy. You could call this new American English DCSS* NL2AMENG. *If you want to save a language DCSS with a different levelid, a DCSS whose name contains the levelid must be defined in DMKSNT. This new language DCSS can be used by building a new CMS nucleus and specifying the new levelid in response to message 295R during the build process. Note that you must have a saved language DCSS with this levelid for every language that will be used with the new CMS nucleus.*

3. *When it successfully saves a language DCSS, LANGGEN tries to add an entry with that langid to the SYSTEM LANGUAGE file. For this to be successful, a copy of the SYSTEM LANGUAGE file must be on a read/write disk accessed before any read-only disk that contains a SYSTEM LANGUAGE file. To ensure this, copy the SYSTEM LANGUAGE file from the S-disk to your A-disk before running LANGGEN. When you are done, the SYSTEM LANGUAGE file on your A-disk must be copied back to the system disk. If you do not have read/write access to the system disk, ask someone who does to copy it back to the system disk for you.*

4. *LANGGEN creates a map on your A-disk which shows the location of each application text deck that was loaded into the DCSS. The fileid of this map is NLxy DCSSMAP where x is the levelid and y is the langid for the map.*

For more information about DCSSs, see the *VM/SP Planning Guide and Reference* or the *VM/SP HPO Planning Guide and Reference.*

LANGGEN's Control File

You must create a LANGGEN control file to issue the LANGGEN command.

For CP, the LANGGEN control file contains the identifier of the CP message repository. LANGGEN loads this message repository on the DASD that was previously defined with the NAMELANG macro. LANGGEN then saves the repository using DIAGNOSE code X'CC'.

For CMS or other applications, this control file identifies the language text decks (files) that LANGGEN loads into a DCSS for the language. These language text decks, which are produced by the LANGMERG command, have a fileid of the form *applid*NLS TXT*langid* *.

The order of the CMS and CMS application files in the LANGGEN control file determines their order in the DCSS. Also, you may not have a LANGGEN control file that identifies only an application; you must include an entry for CMS or else the language is not saved.

Example: Here is a sample LANGGEN control file for American English (langid = AMENG). The file identifies text decks for CP, CMS, and two applications named AP1 and AP2.

```
DMKMES TXTAMENG F
DMSNLS TXTAMENG F
AP1NLS TXTAMENG G
AP2NLS TXTAMENG B
```

When adding a new file for an application, you must be sure to include that application's text deck in the LANGGEN control file.

Appendixes

- Appendix A: CP Device Classes, Types, Models, and Features (for DIAGNOSE code X'24')
- Appendix B: Sample CMS IUCV Program
- Appendix C: Converting Programmable Operator Routing Tables

Appendix A. CP Device Classes, Types, Models, and Features

DIAGNOSE code X'24' requests CP to provide a virtual machine with identifying information and status information about a specified virtual device. The virtual machine must specify the virtual device for which information is requested. CP returns information about the virtual device and associated real device in the Rx, Ry, and Ry+1 registers. CP also provides a condition code identifying the specific device information returned to the virtual machine. These codes are as follows:

DEVICE CLASS CODES

Code	Device Class
X'80'	Terminal Device
X'40'	Graphics Device
X'20'	Unit Record Input Device
X'10'	Unit Record Output Device
X'08'	Magnetic Tape Device
X'04'	Direct Access Storage Device
X'02'	Special Device
X'01'	Fixed-Block Storage

DEVICE TYPE CODES

- For Terminal Device Class

Code	Device Type
X'80'	Binary Synchronous Line for Remote
X'40'	2700 Binary Synchronous Line
X'40'	2955 Communication Line
X'30'	Start/Stop Console
X'20'	Telegraph Terminal Control Type II
X'20'	Teletype Terminal
X'1C'	Undefined Terminal Device
X'18'	IBM 2741 Communication Terminal
X'14'	IBM 1050 Data Communication System
X'10'	IBM Terminal Control Type I
X'08'	Synchronous Data Link Control
X'00'	IBM 3210 Console
X'00'	IBM 3215 Console
X'00'	IBM 2150 Console
X'00'	IBM 1052 Console

Figure 37 (Part 1 of 6). CP Device Classes, Types, Models, and Features

- For Graphics Device Class

Code	Device Type
X'C0'	High Function Graphics Device
X'80'	IBM 2250 Display Unit
X'40'	IBM 2260 Display Station
X'20'	IBM 2265 Display Station
X'10'	IBM 3066 Console
X'08'	IBM 1053 Printer
X'04'	IBM 3138 System Console
X'04'	IBM 3148 System Console
X'04'	IBM 3158 System Console
X'04'	IBM 3277 Display Station
X'01'	IBM 3278 Display Station
X'01'	IBM 3279 Display Station
X'01'	IBM 3290 Information Panel
X'02'	IBM 3284 Printer
X'02'	IBM 3286 Printer
X'02'	IBM 3287 Printer
X'02'	IBM 3288 Printer
X'02'	IBM 3289-E Printer
X'02'	IBM 4250 Printer

- For Unit Record Input Device Class

Code	Device Type
X'90'	IBM 2520 Card Reader/Punch
X'88'	IBM 1442 Card Reader/Punch
X'84'	IBM 3505 Card Reader
X'82'	IBM 2540 Card Reader
X'81'	IBM 2501 Card Reader
X'80'	Card Reader
X'40'	Timer
X'24'	IBM 1017 Paper Tape Reader
X'22'	IBM 2671 Paper Tape Reader
X'21'	IBM 2495 Magnetic Tape Cartridge Reader
X'20'	Tape Reader

Figure 37 (Part 2 of 6). CP Device Classes, Types, Models, and Features

- For Unit Record Output Device Class

Code	Device Type
X'90'	IBM 2520 Card Punch
X'88'	IBM 1442 Card Punch
X'84'	IBM 3525 Card Punch
X'82'	IBM 2540 Card Punch
X'80'	Card Punch
X'4D'	IBM 3800 Model 8 Printing Subsystem
X'4B'	IBM 4248 Printer
X'4A'	IBM 4245 Printer
X'49'	IBM 3800 Model 3 Printing Subsystem
X'47'	IBM 3262 Printer
X'46'	IBM 3289 Printer
X'45'	IBM 3800 Model 1 Printing Subsystem
X'44'	IBM 1443 Printer
X'43'	IBM 3203 Printer
X'42'	IBM 3211 Printer
X'41'	IBM 1403 Printer
X'40'	Printer
X'24'	IBM 1018 Paper Tape Punch
X'20'	Tape Punch

- For Magnetic Tape Device Class

Code	Device Type
X'82'	IBM 3422 Tape Drive
X'80'	IBM 2401 Tape Drive
X'40'	IBM 2415 Tape Drive
X'20'	IBM 2420 Tape Drive
X'10'	IBM 3420 Tape Drive
X'08'	IBM 3410/3411 Tape Drive
X'04'	IBM 8809 Tape Drive
X'02'	IBM 3430 Tape Drive
X'01'	IBM 3480 Tape Drive

Figure 37 (Part 3 of 6). CP Device Classes, Types, Models, and Features

- For Direct Access Storage Device Class

Code	Device Type
X'80'	IBM 2301 Parallel Drum
X'80'	IBM 2303 Serial Drum
X'80'	IBM 2311 Disk Storage Drive
X'80'	IBM 2321 Data Cell Drive
X'40'	IBM 2314 Disk Storage Facility
X'40'	IBM 2319 Disk Storage Facility
X'20'	IBM 3380 Disk Storage Facility
X'10'	IBM 3330 Disk Storage Facility
X'10'	IBM 3333 Disk Storage and Control
X'08'	IBM 3350 Disk Storage Facility
X'04'	IBM 3375 Disk Storage Facility
X'02'	IBM 2305 Fixed Head Storage Device
X'01'	IBM 3340 Disk Storage Facility

- For Special Device Class

Code	Device Type
X'80'	Channel-to-Channel Device (CTCA or 3088)
X'40'	37XX Programmable Communications Controller
X'20'	3851 Mass Storage Controller
X'04'	SRF (7443) device
X'01'	Device not supported by VM/SP

- For Fixed-Block Storage Device Class

Code	Device Type
X'05'	3370-4
X'02'	3370, Model A1, A2, B1, and B2
X'01'	3310
X'00'	Generic Fixed-Block (see Note)

Note: Code X'00' applies to a device whose specific type CP has not yet determined. The proper bit value is assigned when a 'Read Device Characteristics' command is issued at IPL.

MODEL CODES (Column 35 in Accounting Card)

As specified in the RDEVICE macro at system generation.

Note: FB-512 device model codes are specified as:

Code	Device Type
X'00'	3310, 3370 Models A1 and B1
X'04'	3370 Models A2 and B2

Figure 37 (Part 4 of 6). CP Device Classes, Types, Models, and Features

FEATURE CODES (Column 36 in Accounting Card)

- For Printer Devices

Code	Feature
X'80'	3800 has four Writable Character Graphic Modifications (WCGM)
X'40'	Extended Sense Bytes
X'01'	UCS

- For Magnetic Tape Devices

Code	Feature
X'80'	7-Track
X'40'	Dual Density
X'20'	Translate
X'10'	Data Conversion

- For Direct Access Storage Devices

Code	Feature
X'80'	Rotational Position Sensing (RPS)
X'80'	Fixed Head Device
X'40'	Extended Sense Bytes (24 bytes)
X'20'	Top Half of 2314 Used as 2311
X'20'	Device is a 3330V system virtual machine
X'10'	Bottom Half of 2314 Used as 2311
X'08'	35MB Data Module (mounted)
X'04'	70MB Data Module (mounted)
X'02'	Reserve/Release are valid CCW operation codes
X'01'	Device is a 3330V virtual machine
X'01'	3330 Virtual MSS volume

- For special devices

Code	Feature
X'80'	Type Five Channel Adapter (3725)
X'40'	Channel-to-Channel is type 3088
X'20'	Type 2 channel adapter for 370X
X'20'	Type 3 channel adapter for 370X
X'10'	Type 1 channel adapter for 370X
X'10'	Type 4 channel adapter for 370X

- For terminal devices

Code	Feature
X'02'	3270 Mode, Virtual 3215 Device
X'01'	Dial Feature

Figure 37 (Part 5 of 6). CP Device Classes, Types, Models, and Features

- For Graphic Devices

Code	Feature
X'80'	Operator Identification Card Reader
X'01'	Device Supports WRITE STRUCTURED FIELD QUERY

Figure 37 (Part 6 of 6). CP Device Classes, Types, Models, and Features

Appendix B. Sample CMS IUCV Program

This program demonstrates how one might use the CMS IUCV support in VM to use the CP Message System Service. It establishes communications via IUCV and then waits for incoming messages. Each time a message comes in, it is logged in a file named 'MSG FILE'. When the user wishes to have the program terminate, he sends himself a message which contains 'STOP '.

Note: This program works, however, it could be enhanced as it virtually ignores any error conditions. It is presented as a sample and should be viewed as such.

```
*********************************************************************
*                                                                   *
*   CMS IUCV program to use the CP Message System Service.          *
*                                                                   *
*********************************************************************
         SPACE 2
MSG      CSECT
         BALR R12,0               Establish
         USING *,12               addressability
         USING NUCON,R0
         SSM   DISABLE            Disable interrupts til we want them
         LR    11,14              Save the return address in R11
*
*  Tell CMS that this program wishes to use IUCV, and will be
*  identified by the name of SAM.  An address is given for pending
*  connects, however, this should never happen.
*
         HNDIUCV SET,NAME=SAM,EXIT=CONPEND
         LA    R2,IUCVPLST        Get the address of our IUCV parameter
         USING IPARML,R2          list and establish addressability
*
*  Build the parameter list with the IUCV macro, then establish the
*  connection and set up the path's exit.
*
         IUCV CONNECT,PRMLIST =(R2),USERID=STARMSG,MF=L
         CMSIUCV CONNECT,NAME=SAM,PRMLIST=(R2),EXIT=MSGPATH
         MVC   PATHID(2),IPPATHID   Save the path id IUCV gives us
         DROP  R2
*
*  Wait for *MSG to tell us our connection is complete.  Our exit will
*  post that CONCOMP ECB when the interrupt comes in.  Note that
*  WAITECB internally will enable us for interrupts.
*
         WAITECB ECB=CONCOMP,FORMAT=OS
         LA    R11,SETMSG         Issue the SET MSG IUCV command to
         SVC   202                tell CP we want messages via IUCV
         DC    AL4(1)
*
*  Wait here for messages.  The MSGIN ECB will be posted each time a
*  message comes in.  If the message is 'STOP', quit and clean up,
*  otherwise, write the message to a log file name 'MSG FILE'.
*
WAITLOOP EQU   *
         WAITECB ECB=MSGIN,FORMAT=OS
         CLC   MSGAREA+8(4),=CL4'STOP'    If the message says to stop,
         BE    CLEANUP                    Stop and clean up.
         FSWRITE 'MSG FILE A',FORM=E,BUFFER=MSGAREA,RECFM=F,BSIZE=140
         XC    MSGIN(4),MSGIN             Zero out the ECB
         B     WAITLOOP                   Wait for next message
*
*  Issue the CP SET MSG ON command to tell CP we no longer want
*  messages via IUCV.  Then issue the HNDIUCV CLR macro to tell CMS
*  we no longer wish to use IUCV and return to CMS.
*
CLEANUP  EQU   *
         LA    R1,SETON           Issue the SET MSG ON command to
         SVC   202                tell CP we no longer want our
         DC    AL4(1)             messages via IUCV
         HNDIUCV CLR,NAME=SAM
         BR    R11                Return to CMS
         DROP  R12
         EJECT
```

```
*********************************************************************
*                                                                   *
*   This is our external interrupt exit for the path we've set up   *
*   between *MSG and ourselves.  It works as follows:               *
*                                                                   *
*   Connection Complete:  Post the CONCOMP ECB and return           *
*                                                                   *
*   Incoming message:  Receive the message data in our message      *
*                      buffer, post the MSGIN ECB and return.       *
*                                                                   *
*********************************************************************
          SPACE 2
MSGPATH   EQU   *
          BALR  R12,0              Establish
          USING *,R12              addressability
          USING IPARML,R2          R2 points to ext. int. buffer
          CLI   IPTYPE,X'02'       Is it a connection complete?
          BE    CONNCOMP           If so, it's a special case.
          XC    IUCVPLST(40),IUCVPLST  Zero out the IUCV parameter list
          XC    MSGAREA(140),MSGAREA   and the message buffer
          MVC   MSGCLASS,IPTRGCLS      Save the class
          MVC   MSGID,IPMSGID          and the id of the message
          LA    R2,IUCVPLST            Let R2 point to the PRMLIST
*
*  Get the message which has been sent to you by issuing an
*  IUCV RECEIVE.  Put the message in MSGAREA.
*
          IUCV  RECEIVE,PATHID=PATHID,PRMLIST=(R2),MSGID=MSGID,TRGCLS=MS*
                GCLASS,BUFFER=MSGAREA,BUFLEN=BUFERLEN

          OI    MSGIN,X'40'        Post the MSGIN ECB
          BR    R14                and return to CMS
*
*  Come here if it's a connection complete interrupt (IPTYPE = X'02')
*
CONNCOMP  EQU   *
          OI    MSGIN,X'40'        Post the CONCOMP ECB
          BR    R14                and return
*
*  This is our connect pending exit.  It should never be driven.
*  If it is, we'll just ignore it and return.
*
COMPEND   EQU   *
          BR    R14
          EJECT
```

```
*********************************************************************
*                                                                   *
*                E Q U A T E S and C O N S T A N T S                *
*                                                                   *
*********************************************************************
          SPACE 2
          DS    0D
SETMSG    DC    CL8'CP'             Parameter list to tell CP we
          DC    CL8'SET'               want our message via IUCV.
          DC    CL8'MSG'
          DC    CL8'IUCV'
          DC    8X'FF'
SETON     DC    CL8'CP'             Parameter list to tell CP we
          DC    CL8'SET'               want our message set back
          DC    CL8'MSG'               to 'ON'.
          DC    CL8'ON'
          DC    8X'FF'
IUCVPLST  DC    40X'00'             Used for the IUCV parameter list
MSGID     DS    F                   Holds the IUCV message id
MSGCLASS  DS    F                   Holds the IUCV message class
STARMSG   DC    CL8'*MSG'           The name of the IUCV Message Service
SAM       DC    CL8'SAM'            Name which CMS will know us by
CONCOMP   DC    F'0'                Connection Complete ECB
MSGIN     DC    F'0'                Message has arrived ECB
PATHID    DC    H'0'                Holds the IUCV path id
BUFERLEN  DC    H'140'              Says our buffer will be 140 chars
MSGAREA   DC    CL140' '            Message buffer area
DISABLE   DC    X'00'               Byte to disable us for interrupts
          REGEQU
          NUCON
          COPY  IPARML
          END
```

Appendix C. Converting Programmable Operator Routing Tables

The **routing table** is a CMS file that contains the information used to control the operation of the programmable operator facility. The routing table enables the programmable operator facility to recognize a message as a command, to determine the action to take when a message comes in, and to recognize the authorized users of programmable operator functions.

The format of the routing table is different from the format in the initial version of the programmable operator facility documented in Release 2 of VM. This format makes the specifications easier and the information clearer. Release 2 routing tables are not compatible with the current format and must either be regenerated by hand or converted using PROPRTCV. PROPRTCV is a utility provided to convert old routing tables to the current format. This utility is written using the System Product Interpreter. Using an old RTABLE as input, PROPRTCV creates a new RTABLE, leaving the old one unchanged. When you execute PROPRTCV, it converts routing tables in the following order:

1. Generates the appropriate configuration statements at the beginning of the new routing table file. See "Routing Table Entry Formats" in the chapter on the Programmable Operator Facility.

   - A LGLOPR statement is added using the existing logical operator userid and nodeid. PROPRTCV prompts you to change this information, if you wish.

   - A TEXTSYM statement is added. Select the TEXTSYM characters to be used. The text fields of the file are scanned for these characters. If any of these characters are found, PROPRTCV informs you and then prompts you for different characters. You can also exit and change the texts that caused the conflict.

   - PROPCHK statements are added, if desired. PROPRTCV prompts you for this information.

   - A HOSTCHK statement is added, if desired. PROPRTCV prompts you for this information.

   - A ROUTE statement is placed after all the above statements have been completed.

   - An entry for the SET command is added with text "/SET /", message type 1, action routine DMSPOR, and parameter SET. PROPRTCV prompts you for any authorization desired for this entry. See "DMSPOR - Miscellaneous supplied action routines" for more information on the SET command.

Note: The SET entry is simply placed after the ROUTE statement. This is probably not where you want it. Move the entry when the routing table conversion is completed.

For each routing table entry, PROPRTCV

2. Encloses the specified text with the blank-separator, (/), for the specified length of the text.

3. Generates a starting column value and an ending column value from the existing displacement and length values.

4. Converts an entry with action routine DMSPOR and parameter TOLGLOPR to the action routine name DMSPOS. PROPRTCV prompts you for the routing target information to be used as the parameter.

5. Converts an entry with action routine DMSPOR and parameter LOADTBL to the action routine name DMSPOL and no parameter.

Summary of Changes

Structural Changes

This book contains material formerly found in the *VM/SP System Programmer's Guide*, SC19-6203. and the *VM/SP HPO System Programmer's Guide*, SC19-6224. Figure 38 shows the Release 4 books made obsolete by this reorganization, the new system programming books, and the topics these new books contain.

To obtain editions of the *VM/SP System Programmer's Guide*, you must order using the pseudo-number assigned to the respective edition. For:

VM/SP Release 4, order ST00-1578
VM/SP Release 3, order ST00-1352
VM/SP Release 2, order SQ19-6203
VM/SP Release 1, order ST19-6203.

To obtain editions of the *VM/SP HPO System Programmer's Guide*, you must order using the pseudo-number assigned to the respective edition. For:

VM/SP HPO Release 4.2, order ST00-1897.

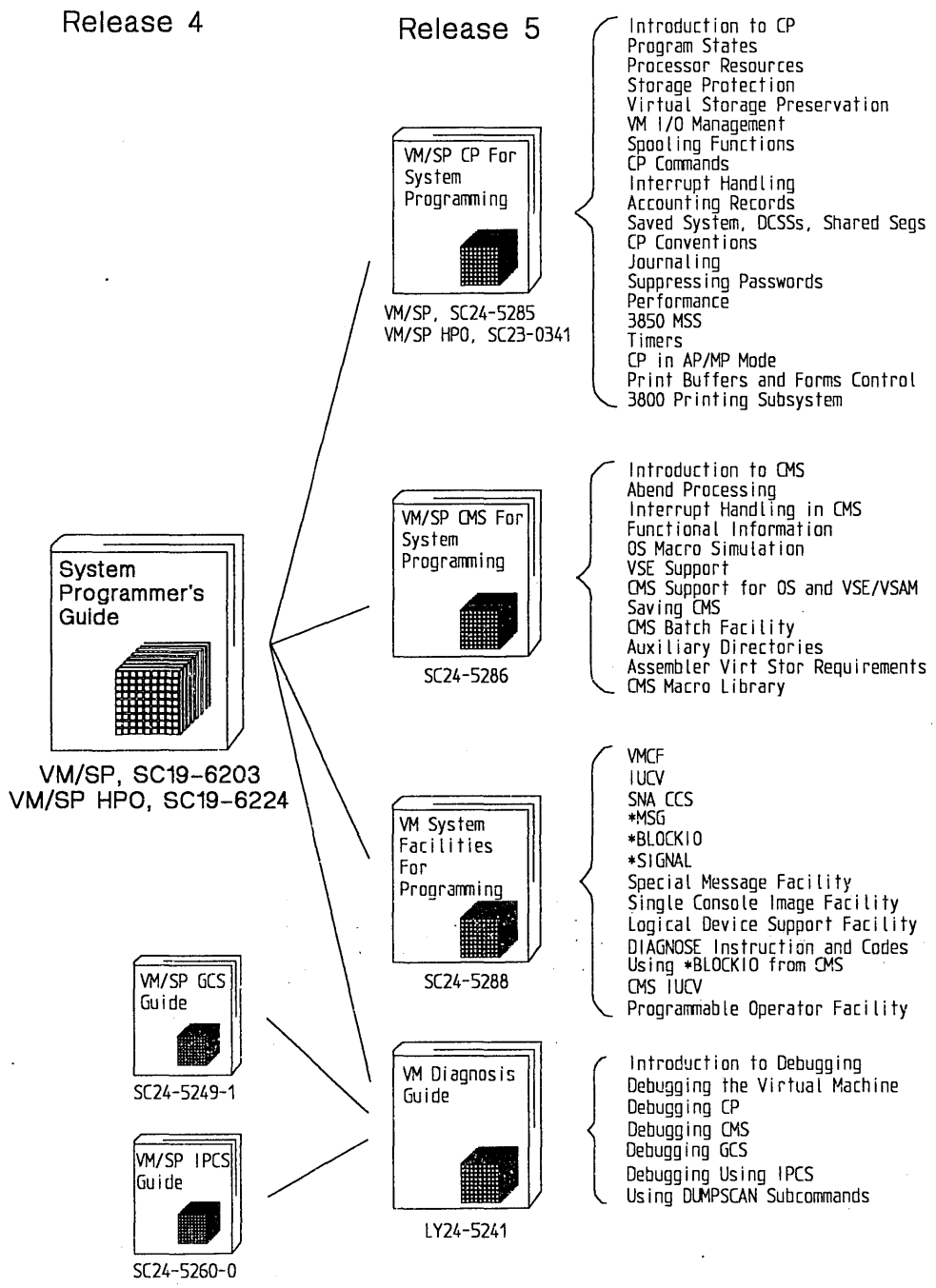


Figure 38. New VM System Programming Manuals for Release 5

Technical Changes for VM System Facilities for Programming

Summary of Changes
for SC24-5288-0
for VM Release 5

Transparent Services Access Facility (TSAF)

This new facility lets users connect to and communicate with local or remote virtual machines within a group of systems. With TSAF, a user can connect to a program by specifying a name that the program has made known, instead of specifying a userid and node id. The Transparent Services Access Facility (TSAF) consists of the TSAF virtual machine component, APPC/VM, and two CP system services.

Advanced Program-to-Program Communication/VM (APPC/VM)

Is based on IUCV and provides services similar to IUCV. APPC/VM, with the TSAF virtual machine, provides services within a single system and throughout a group of systems, unlike IUCV, which provides services only within a single system.

TSAF virtual machine component

This new new VM/SP component handles communication between systems by letting APPC/VM paths span more than one system.

National Languages Supported by VM/SP

VM/SP now supports the following national languages: American English, German, French, Kanji, and Uppercase English. This book shows how to store language-related files and make them available to users. This information includes,

- The LANGMERG and LANGGEN commands
- DIAGNOSE Codes X'C8' and X'CC'.

The SPOOL System Service

This new CP system service provides a way for VM/SP to support a printer subsystem. It allows authorized users an interface for communication between CP and a printer subsystem. (The subsystem printers are called logical printers.) The SPOOL System Service allows a virtual machine to manipulate spool files in several ways.

Enhanced IUCV

IUCV adds a new parameter on the IUCV macro and a new CP system service.

- The new CONTROL= parameter on the DECLARE BUFFER and CONNECT functions enable the control program to manage paths to the virtual machine.
- The new Message All System Service controls output to the terminal.
- The entire IUCV chapter has been rewritten and reorganized for easier use.

CMS and GCS Support for IUCV

CMS and GCS IUCV changes are due to requirements needed for windowing in CMS, CMS and GCS Supervisor IUCV use, and the Transparent Services Access Facility (TSAF).

Access Verification Routines

While VM/SP provides many security functions, added support for access verification routines provides a standard interface to RACF/VM Support PRPQ or user-written routines that can provide a higher lever of security. Although the access verification routines support does not by itself provide security functions, it allows you to install software that does.

For example, to increase security of minidisk accesses, logon passwords, and movement of spool files, you can install access verification routines with the Resource Access Control Facility (RACF) (Program Number 5740-XXH) and RACF/VM Support PRPQ (Program Number 5767-002).

This support added DIAGNOSE code X'A0'.

Enhanced DIAGNOSE Code X'08'

This enhanced DIAGNOSE code provides a virtual machine with the capability of managing a full-screen environment at all times. (The 8K response buffer limit is eliminated.)

Enhanced DIAGNOSE Code X'5C'

This enhanced DIAGNOSE code supports the use of a message number greater than three characters. A user may now change the length of the message identifier.

DIAGNOSE Code X'B0'

This new DIAGNOSE code allows a virtual madchine to access diagnostic information saved for a user running in a protected application, for whom a re-IPL has been tried.

DIAGNOSE Code X'B4'

This new DIAGNOSE code allows a user to associate an External Attribute Buffer (XAB) that an application provides with a virtual printer device. The XAB is a control block that contains data the user creates to specify additional information about a print file. Each print file has its own XAB and CP has the facilities to maintain the XABs.

DIAGNOSE Code X'B8'

This new DIAGNOSE code lets an application virtual machine read, write, or erase an External Attribute Buffer (XAB) associated with a spool file. The XAB is a control block that contains data the user creates to specify additional information about a print file. Each print file has its own XAB and CP has the facilities to maintain the XABs.

DIAGNOSE Code X'BC'

This new DIAGNOSE code opens a spool file for a spooled reader device and returns spool file identification. If the spool file is already open on the reader

device, this DIAGNOSE still returns the spool file identification for the open file.

DIAGNOSE Code X'D0'

This new DIAGNOSE code lets any virtual machine provide CP with the virtual device address and the volume serial of a 3480 volume.

DIAGNOSE Code X'D4'

This new DIAGNOSE code provides support for an alternate userid. It lets a class B virtual machine tell CP the userid of a worker machine that is performing the work and the userid of the end-user for which it is authorized to work. The end-user's userid is considered to be the "alternate userid."

Miscellaneous changes

TERMINAL CONMODE 3270 Enhancements

Support for TERMINAL CONMODE 3270 is expanded so that virtual machine START I/Os are handled as 3270 START I/Os. The console is placed in full-screen mode with an application program controlling information sent to the terminal.

Programmable Operator Facility Documentation

The Programmable Operator chapter of the *VM/SP System Programmer's Guide* has been combined with the same chapter in the *VM/SP Operator's Guide*. This chapter now contains a complete description of the programmable operator and all commands formats.

DIAGNOSE codes

The DIAGNOSE codes have been reorganized for ease-of-use. The entry values for the registers are now easier to find as are the program exceptions.

Minor technical and editorial changes have been made throughout this publication.

Summary of Changes for VM/SP and VM/SP HPO System Programmer's Guides

Summary of Changes
for SC19-6203-3
for VM/SP Release 4

Group Control System (VM/SP GCS)

This new component of VM/SP is a virtual machine supervisor that provides simulated MVS services and supports a multitasking environment. For more information on the Group Control System (GCS), refer to the *VM/SP Group Control System Guide*, SC24-5249.

Signal System Service

This new CP system service allows virtual machines in a Virtual Machine Group to signal each other. The Signal System Service can only be used by virtual machines in a Virtual Machine Group.

Saved System 8M Byte Limit Removal

With the addition of this support, the SAVESYS, VMSAVE, and IPL functions have been enhanced to allow a page image copy of up to a 16M byte virtual machine to be saved and restored.

CP FRET Trap

The CP FRET Trap can be used as an aid in solving problems caused by improper use of CP storage and to solve many storage overlay problems.

VMDUMP Enhancements

DIAGNOSE Code X'94' is available to allow a virtual machine to request dumping of its virtual storage. Also, the three address range restriction has been removed from the VMDUMP command.

DIAGNOSE Code X'98'

Using DIAGNOSE Code X'98', a virtual machine can lock and unlock virtual pages, and execute its own real channel programs.

The Programmable Operator Facility

The Programmable Operator Facility has been enhanced to support distributed operations in an SNA network through an interface, the Programmable Operator/NCCF Message Exchange (PMX), with the Network Communications Control Facility (NCCF). The VM/SP Release 4 programmable operator:

- Allows an NCCF operator to be identified to the programmable operator so that any messages intended for the logical operator may be routed to that NCCF operator.
- Allows an NCCF operator to issue programmable operator commands and receive responses.
- Provides the LGLOPR command for assigning, releasing and replacing the logical operator during operation.

CPTRAP Enhancements

CPTRAP is a major service aid used in problem determination. Enhancements to the CPTRAP command provide two additional functions, GROUPID and WRAP, and one additional entry type, X'3D'.

Enhancements to TRAPRED makes reviewing the trap data easier by providing more selectivity for X'3D', X'3E', and X'3F' entries and by providing a way to display formatted output of the trapped data.

Information on CPTRAP has been rewritten and reorganized for ease-of-use. It has also been moved to the Part 3, the debugging section, since it is a debugging tool.

Interactive Problem Control System (VM/SP IPCS)

VM/SP Release 4 has been enhanced to include IPCS as a component of VM/SP. VM/SP IPCS is equivalent to the VM/Interactive Problem Control System Extension (VM/IPCS/E) Licensed Program (5748-SA1).

Inter-User Communications Vehicle (IUCV) Enhancements

IUCV now supports the movement of data on the SEND, RECEIVE, and REPLY functions from discontiguous buffers. The modified IUCV macro handles the new BUFLIST= parameter on SEND and RECEIVE functions and the new ANSLIST= parameter on the SEND and REPLY functions.

Expansion of User Classes

The DIRECT command has been enhanced and the OVERRIDE command has been added to provide the user with more than the seven IBM defined user classes. You can now choose from 32 user classes, A - Z, and 1 - 6.

Remote Spooling Communications Subsystem Networking Version 2

With the release of the Remote Spooling Communications Subsystem Networking Version 2 licensed program (5664-188), any reference to RSCS in this manual applies to RSCS Version 2. Information pertaining to RSCS can be found in the *VM/SP Remote Spooling Communications Subsystem Version 2 General Information*, GH24-5055.

Miscellaneous changes

IOCP Support Enhancements

This support adds new MSSF command words to DIAGNOSE code X'80'.

Integration of Functional Enhancements to VM/SP Release 3

Information has been added to support:

- The 3290 Information Panel
- The 3370 Direct Access Storage Model
- The 4248 Printer
- The 4361 Model Groups 3, 4, and 5 Processor
- The 4381 Model Groups 1 and 2 Processor
- VM/SP 3800 Model 3 Compatibility Support

  Compatibility support allows VM/SP users to access the 3800 Model 3 Printing Subsystem. Existing programs designed to produce 3800 Model 1 printer output may produce output for the 3800 Model 3 printer with little or no program change. Use of this support provides improved print quality (240 x 240 pel resolution) and the addition of a 10 lines-per-inch (LPI) vertical space option.

DIAGNOSE Code X'8C'

DIAGNOSE code X'8C' has been enhanced to allow a user to access all of the data returned by CP's WRITE STRUCTURED FIELD QUERY.

DMKFRE/DMKFRT Split

The module DMKFRE has been split into two modules, DMKFRE and DMKFRT. DMKFRE handles all requests for free storage as well as calls to DMKFRET to release free storage. DMKFRT handles all requests to return

free storage that cannot be handled by the microcoded CP assist FRET function.

Minor technical and editorial changes have been made throughout this publication.

Summary of Changes
for SC19-6223-6
As Updated January 1986
for VM/SP HPO Release 4.2

AUTODEACTIVATION OF RESTRICTED PASSWORDS AND DIRECTORY ENHANCEMENTS

New: Programming Support

Adds support to enhance system integrity by minimizing the exposure of unauthorized system access through the use of restricted passwords.

Directory enhancements include two new control statements, PROFILE and INCLUDE, which permit a group of control statements common to more than one user directory entry to be coded only once, and the removal of the directory size limit.

ACCESS VERIFICATION ROUTINES

New: Programming Support

Adds support, including a new directory control statement, ACIGROUP, for access verification routines. When used with the RACF/VM Support PRPQ, access verification routines help increase security for VM/HPO installations. The access verification routines support does not itself provide security functions; but it allows the installation to install software that does. To increase security of minidisk accesses, logon passwords, and movement of spool files, an installation should install access verification routines with the Resource Access Control Facility (RACF) (Program Number 5740-XXH) and RACF/VM Support PRPQ (Program Number 5767-002).

SECURITY ENHANCEMENTS

New: Programming Support

This support allows an installation to specify the number of password attempts that a user is allowed before being locked out of the system and the amount of time that the user will be locked out after the unsuccessful attempts. In addition, if an installation has installed RACF, this support now calls RACF to authorize the use of STCP and LINK commands.

VECTOR FACILITY

New: Hardware Support

Support is provided for the Vector Facility in System/370 mode configured to a 3090 Processor. The Vector Facility is a synchronous vector/scalar instruction processor that can manipulate values (usually floating-point) at high speed. Compiled engineering and scientific FORTRAN applications can use the array processing capability of the Vector Facility. VM/SP HPO supports the use of this facility by multiple virtual machines.

PAGE MIGRATION

Changed: Programming Support

Page migration is changed to select pages (rather than segments) for migration on a reference basis instead of by time-stamp (age basis). Also, pages are migrated down the demand page hierarchy, instead of being migrated directly to the pre-allocated migration area. This reduces the time required to retrieve those pages that become active in the near future.

Because migration of swap tables is sometimes necessary even when page migration is not actively moving pages, swap table migration is now invoked independently of page migration (rather than after page migration). Swap table migration is further improved by migrating swap tables regardless of whether all the pages in the segment have been migrated.

SYSPAG macro, commands, and monitor are enhanced to support the changed migration algorithms. Installations should retune free storage.

3380 DIRECT ACCESS STORAGE DEVICE MODELS AE4/BE4

New: Hardware Support

VM/SP HPO now supports the 3380 DASD Models AE4/BE4. The 3380 Models AE4/BE4 are count-key-data (CKD) devices that attach to high-speed channels only, via the 3880 Control Unit. The 3880 can attach up to 16 physical spindles (32 logical devices) of 3380 Models AE4 and BE4 directly to data streaming channels. The AE4 models attach to the system and may be the first device on a string. Strings of different 3380 device models may be intermixed at the control unit level.

DOCUMENTATION CHANGES

Minor technical and editorial changes have been made throughout this publication.

Summary of Changes
for SC19-6203-2
for VM/SP Release 3

Programmable Operator Facility

Several enhancements to the programmable operator facility added are:

- Message routing with nicknames
- Remote node availability
- Enhanced text comparison
- EXEC action routines
- LOG recording and error handling

PER

Problem determination capability is greatly extended and enhanced by the new CP command, PER.

DASD Block I/O System Service

The DASD Block I/O System Service allows a virtual machine fast, device-independent asynchronous access to fixed size blocks on CMS formatted virtual DASD I/O devices.

IUCV

Inter-User Communication Vehicle (IUCV) extensions provide:

- SEND and REPLY extensions
- An extended mask capability for control interrupts
- An expanded trace capability to record all IUCV operations
- A macro option to initialize the parameter list
- Support for the DASD block I/O system service.

The IBM 3088 Multisystem Communications Unit

The IBM 3088 Multisystem Communications Unit interconnects multiple systems using block multiplexer channels. The 3088 uses an unshared subchannel for each unique address and is fully compatible with existing channel-to-channel adapter protocol.

CMS IUCV support

Support for IUCV communication has been introduced into CMS. This support allows multiple programs within a virtual machine to use IUCV functions. Included is the ability to initialize a CMS machine for IUCV communication and to invoke IUCV functions via new CMS macros. These macros also allow the user to specify path-specific exits for IUCV external interrupts.

CMS abend exits

A general CMS abnormal exit capability is provided so that user programs may specify the address of a routine to get control before CMS abend recovery begins. An exit is established and cleared through a new CMS macro.

Enhanced immediate command support

The immediate command capability of CMS is extended by allowing users to define their own immediate commands.

Enhanced VSAM support

CMS supports VSE/VSAM Release 3 which includes significant enhancements designed to improve catalog reliability and integrity while providing additional serviceability and usability. VSE/VSAM Release 2 is not supported.

Miscellaneous

Changes to the DIAGNOSE code X'00' interface provide the time zone differential from Greenwich Mean Time.

DIAGNOSE code X'8C' allows a virtual machine to access device dependent information without having to issue a WRITE STRUCTURE FIELD QUERY REPLY.

CMSSEG has been eliminated and the code was merged into the CMS Nucleus.

The Remote Spooling Communications Subsystem (RSCS) section of this manual has been removed as it pertained to RSCS as a component of VM/370. Now, any reference to RSCS in this manual applies to the RSCS Networking Programming Product, and information can be found in the *VM/SP Remote Spooling Communications Subsystem Networking Program Reference and Operations Manual*, SH24-5005.

A newly added appendix lists and describes the CMS macros applicable to VM/SP.

Minor technical and editorial changes have been made throughout this publication.

Glossary of Terms and Abbreviations

This glossary defines new terms and all-capitals abbreviations related to the VM/SP and VM/SP HPO. This glossary is especially oriented for system programmers. Therefore, some terms already defined in the *VM/SP Library Guide, Glossary, and Master Index*, SC19-6207, do not appear here or may be defined slightly differently. Another glossary you may want to reference is the *IBM Data Processing Glossary*.

A

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method

Advanced Communications Function for the Virtual Telecommunications Access Method (ACF/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. VTAM runs under MVS (OS/VS1 and OS/VS2), VSE, VM/SP, and VM/SP HPO, and supports direct control application programs and subsystems such as VSE/POWER.

Advanced Program-to-Program Communication/VM (APPC/VM). An application program interface (API) for communicating between two virtual machines that is mappable to the SNA LU 6.2 APPC interface and is based on IUCV functions. Along with the TSAF virtual machine, APPC/VM provides this communication within a single system and throughout a collection of systems.

AP/MP mode. A mode of VM used when running in an attached processor or multiprocessor system.

attached processor. A processor with no I/O capability. An attached processor is always linked to the processor initialized for I/O handling.

auxiliary storage. Data storage other than main storage; in VM, auxiliary storage is usually a direct access device.

B

basic control (BC) mode. A mode in which a virtual machine resumes execution after an I/O interrupt, a page fault, or a DIAGNOSE code X'18'.

C

CAW. channel address word

CCW. channel command word

channel address word (CAW). An area in storage that specifies the location in main storage at which a channel program begins.

channel command word (CCW). A doubleword at the location in main storage specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

channel status word (CSW). An area in storage that provides information about the termination of input/output operations.

Channel-to-Channel Adapter. A hardware device that can be used to connect two channels on the same computing system or on different systems.

CKD. Count-Key-Data

CMS. refers to the VM/370 Conversational Monitor System component enhanced by the functions included in the VM/SP package.

CMS/DOS. refers to the DOS-like simulation environment provided under the CMS component of the VM/SP.

concurrently. Concerning a mode of operation that includes the performance of two or more operations within a given interval of time.

CMS system disk. The virtual disk (S-disk) that contains the CMS nucleus and the disk-resident

CMS commands. The CMS system disk can have extensions, usually the Y-disk.

Count-Key-Data. Those DASD devices whose architecture defines variable size records consisting of count, key, and data fields.

CP. refers to the VM/370 Control Program component enhanced by the functions included in the VM/SP package.

CSW. channel status word

D

DCSS. Discontiguous shared segments.

directory. For VM, a CP disk file that defines each virtual machine's normal configuration: the userid, password, normal and maximum allowable virtual storage, CP command privilege class or classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired.

discontiguous shared segments (DCSS). Synonymous with *discontiguous segment.*

discontiguous segment. A 64K segment of storage that was previously loaded and saved and assigned a unique name. The segment(s) can be shared among virtual machines if the segment(s) contain reentrant code.

dispatch list. A list of those virtual machines that are executable and currently competing for a time slice of processor resources.

dispatcher. The program in CP that places virtual machines or CP tasks into execution. The dispatcher selects the next virtual machine to run and prepares the virtual machine for problem state execution.

dispatch request queue. A queue of executable CP tasks, I/O tasks, and timer requests that are ready to be dispatched.

DPA. dynamic paging area

dynamic address translation. In System/370 virtual storage systems, the change of a virtual address to a real storage address during execution of an instruction.

E

EXEC. refers to EXECs using the System Product Interpreter (REXX), EXEC 2, or CMS EXEC languages.

extended control (EC) mode. Extended control mode, a System/370 mode for formatting and use of control and status information. Contrast with "basic control (BC) mode."

F

FBA. Fixed-block architecture.

file status table (FST). A table that describes the attributes of a file on a CMS disk, including filename, filetype, filemode, date last written, and other status information.

Fixed-Block Architecture (FBA). Those DASD devices whose architecture uses fixed blocks or records of 512 bytes.

FST. file status table

G

GCS. Group Control System facility

Group Control System. An operating environment that provides a problem state OS subtasking environment with common storage access for members of a virtual machine group.

guest virtual machine. A virtual machine in which an operating system is running.

I

interactive. (1) An application in which each user entry calls forth a response from a system or program. (2) The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine uses less than its allocated time slice because of terminal I/O, the virtual machine is classified as being interactive. See also non-interactive.

Interactive Problem Control System (IPCS or VM/SP IPCS). A component of VM that permits on-line problem management, interactive problem diagnosis, on-line debugging for disk-related CP or virtual machine abend dumps, problem tracking, and problem reporting.

Inter-User Communication Vehicle (IUCV). A VM generalized CP interface that aids the transfer of messages either among virtual machines or between CP and a virtual machine.

IPCS. Interactive Problem Control System.

IUCV. Inter-User Communication Vehicle.

L

logical operator. The name given to the virtual machine from which OPERATOR functions requested by the programmable operator facility virtual machine are performed. This name also may describe the person who normally operates the logical operator virtual machine. In a mixed environment, an NCCF operator can be assigned as the logical operator to control a VM distributed system.

logon. The procedure by which a user begins a terminal session.

logoff. The procedure by which a user ends a terminal session.

M

message repository. A source file that contains message texts for a VM component or user application. It is compiled into internal form by the GENMSG command. The message texts in a repository file can be translated and used to support national languages.

minidisk. Synonym for virtual disk.

MSSF. Monitoring and service support facility

N

named system. A collection of saved pages a user can IPL or load by name.

native mode. A mode in which an operating system is run stand-alone on the real machine instead of under VM.

noninteractive. The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine usually uses all its allocated time slice, it is classified as being noninteractive or compute bound. See also interactive.

non-resident pages. Pages whose contents are on DASD but not in real storage. A page is considered non-resident when an attempt to load its real address returns a nonzero condition code.

P

page frame. A block of 4096 bytes of real storage.

page table. A table in CP that indicates whether a page is in real storage and matches virtual addresses with real storage addresses.

prefix storage area (PSA). a page zero of real storage that contains machine-used data areas and CP global data.

preferred machine assist. The hardware feature of certain processors that improves MVS/SP V=R virtual machine performance. The MVS/SP guest virtual machine operates in supervisor state with direct control of its own I/O operations under VM/SP High Performance Option. Note that preferred machine assist is an extension of virtual machine assist, which eliminates CP simulation of certain instruction and interrupts.

programmable operator facility. A facility that allows automatic filtering and routing of messages from a specified virtual machine (for example, the system operator's virtual machine) to a logical operator virtual machine. The logical operator virtual machine is in a local, distributed, or mixed environment. The programmable operator facility also permits installation defined actions to be carried out automatically.

program status word (PSW). An area in storage used to indicate the order in which instructions are

executed, and to hold and indicate the status of the computer system. Synonymous with processor status word.

PSA. Prefix storage area.

PSW. Program status word, or processor status word.

R

real machine. The actual processor, channels, storage, and I/O devices required for operation of VM.

routing table. A CMS file that contains the information used to control the operation of the programmable operator facility. It lets the programmable operator facility recognize a message as a command, determine the action to take when a message comes in, and recognize the authorized users of programmable operator functions.

RSCS. unless otherwise noted, refers to the RSCS Networking Version 2 Program Product (5664-188). When you install and use VM/SP in conjunction with the VM/370 Release 6 System Control Program (SCP), it becomes a functional operating system that provides extended features to the Control Program (CP) and Conversational Monitor System (CMS) components of VM/370 Release 6. VM/SP adds *no* additional functions to the Remote Spooling Communications Subsystem (RSCS) component of VM/370. However, you can appreciably expand the capabilities of this component in a VM/SP system by installing RSCS Networking Version 2 (5664-188).

S

S-disk. See CMS system disk.

S-STAT. A block of storage that contains the file status tables (FSTs) associated with the S-disk. The FSTs are sorted so that a binary search can be used to search for files. The S-STAT usually resides in the CMS nucleus so it can be shared. Only files with filemode of 2 will have their associated FSTs in the S-STAT.

segment. A contiguous 64K area of virtual storage (not necessarily contiguous in real storage) that is allocated to virtual machine or CP.

shadow page table. A table that maps real storage allocations (first level storage) to a virtual machine's virtual storage (third level storage) for use by the real machine in its paging operations.

spool, spooled, spooling. Relates to the reading of input data streams and the writing of output data streams on auxiliary storage devices.

System/370. applies to the 4300 and 303X series of processors.

T

time sharing. Sharing of computer time and resources.

Transparent Services Access Facility (TSAF). A facility that lets users connect to and communicate with local or remote virtual machines within a collection of systems. With TSAF, a user can connect to a program by specifying a name that the program has made known, instead of specifying a userid and nodeid. The Transparent Services Access Facility consists of the TSAF virtual machine component, APPC/VM, and two CP system services.

TSAF. Transparent Services Access Facility

TSAF virtual machine component. A component in VM that handles communication between systems by letting APPC/VM paths span more than one system.

V

virtual address. An address that refers to virtual storage or a virtual I/O device address. It must, therefore, be translated into a real storage or I/O device address when it is used.

virtual disk. A logical subdivision (or all) of a physical disk storage device that has its own address, consecutive storage space for data, and an index or description of the stored data so that the data can be accessed. A virtual disk is also called a minidisk.

virtual machine. A functional simulation of a computer and its associated devices.

Virtual Machine Communication Facility (VMCF). A CP function that provides a method of communication and data transfer between virtual machines operating under the same VM systems.

virtual machine group. The concept in the Group Control System of two or more virtual machines associated with each other through the same named system (e.g. IPL GCS1). Virtual machines in a group share common read/write storage and can communicate with one another through facilities provided by the Group Control System.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

VM. refers to VM/370, VM/SP, or VM/SP HPO.

VMCF. Virtual Machine Communication Facility.

VM/SP. refers to the VM/SP program package when you use it in conjunction with VM/370 Release 6.

VM/VCNA. VM/VTAM Communication Network Application

VM/VTAM Communication Network Application (VM/VCNA). A VTAM application which allows a SNA terminal user to logon to VM though OS/VS1 or VSE.

VSCS. VTAM SNA Console Support component

VSE. refers to the combination of the DOS/VSE system control program and the VSE/Advanced Functions licensed program. "DOS", in certain cases, is still used as a generic term. For example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VSE system may be referred to as DOS disks.

VTAM SNA Console Support component (VSCS). A VTAM application which allows a SNA terminal user to logon to VM though the Group Control System (GCS) facility.

Y

Y-disk. An extension of the CMS system disk.

Y-STAT. A block of storage that contains the file status tables (FSTs) associated with the Y-disk. The FSTs are sorted so that a binary search can be used to search for files. The Y-STAT usually resides in the CMS nucleus so it can be shared. Only files with filemode of 2 will have their associated FSTs in the Y-STAT.

Bibliography

Here is a list of IBM books that can help you use your system. If you don't see the book you want in this list, you might want to check the *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001.

- **Prerequisite Publications**

  IBM System/360 Principles of Operation, GA22-6821

  IBM System/370 Principles of Operation, GA22-7000.

- **Books About VM/SP**

  Virtual Machine/System Product:

  General Information, GC20-1838
  Introduction, GC19-6200
  Release 5 Guide, SC24-5290
  CP for System Programming, SC24-5285
  CMS for System Programming, SC24-5286
  Transparent Services Access Facility Reference, SC24-5287
  Group Control System Command and Macro Reference, SC24-5250
  CMS Command Reference, SC19-6209
  CMS Macros and Functions Reference, SC24-5284
  Application Development Guide, SC24-5247
  Planning Guide and Reference, SC19-6201
  CP Command Reference, SC19-6211
  CMS User's Guide, SC19-6210
  Installation Guide, SC24-5237
  System Messages and Codes, SC19-6204
  System Messages Cross-Reference, SC24-5264
  OLTSEP and Error Recording Guide, SC19-6205
  Terminal Reference, SC19-6206
  Library Guide, Glossary, and Master Index, SC19-6207
  Operator's Guide, SC19-6202
  EXEC 2 Reference, SC24-5219
  System Product Editor User's Guide, SC24-5220
  System Product Editor Command and Macro Reference, SC24-5221
  System Product Interpreter User's Guide, SC24-5238
  System Product Interpreter Reference, SC24-5239
  Problem Reporting Guide, SC24-5282

  Virtual Machine/System Product High Performance Option:

  Operator's Guide, SC19-6225
  Planning Guide and Reference, SC19-6223
  CP for System Programming, SC19-6224
  Installation Guide, SC38-0107
  CP Command Reference, SC19-6227
  System Messages and Codes, SC19-6226
  System Messages Cross-Reference, SC20-0190

Virtual Machine:

Diagnosis Guide, LY24-5241
Running Guest Operating Systems, SC19-6212

Notes:

1. *References in text to titles of publications are given in abbreviated form.*
2. *The VM/SP Library Guide, Glossary, and Master Index, GC19-6207, describes all the VM/SP books and contains an expanded glossary and master index to all the books in the VM/SP library.*
3. *The VM/SP HPO Library Guide, Glossary, and Master Index, GC23-0187 describes all the VM/SP HPO books and contains an expanded glossary and master index to all the books in the VM/SP HPO library.*

- **Other Publications**

IBM 2821 Control Unit Component Description, GA24-3312

IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide, GA24-3543

IBM 3262 Printers 1 and 11 Component Description, GA24-3733

IBM 3270 Information Display System Library User's Guide, GA23-0058

3704 and 3705 Communications Controllers

Introduction to the IBM 3704 and 3705 Communications Controllers, GA27-3051

IBM 3704 and 3705 Communications Controllers Operator's Guide, GA27-3055

IBM 3704 Control Panel Guide, GA27-3086

IBM 3705 Control Panel Guide, GA27-3087

IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual (OS/VS TCAM Levels 5 and 6 SVS - 5742-017) SCP 5742, 5744-AN1/BA2, 5747-AG1/AJ2, GC30-3008 in VS1; VS2 Rel 1.6, 1.7, 2, SCP 5744-BA1, GC30-3007

IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual (TCAM 10 SVS - 5742-017) SCP 5742, 5744-AN1/BA2, 5747-AG1/AJ2, GC30-3008

IBM 3725 Communication Controller Operator's Guide, GA33-0014

IBM 3725 Operator Console Reference and Problem Analysis Guide, GA33-0015

IBM Virtual Machine Facility/370: Performance/Monitor Analysis Program, SB21-2101

ACF/VTAM

Network Program Product General Information Manual, GC23-0108
ACF/VTAM System Programmer's Guide, SC38-0258
Network Program Products Planning, SC23-0110
VTAM Installation and Resource Definition, SC23-0111
VTAM Customization, SC23-0112.
VTAM Operation, SC23-0113.

EREP

Environmental Recording Editing and Printing (EREP) Program, GC28-1178
Environmental Recording Editing and Printing (EREP) Program User's Guide and Reference, GC28-1378

VM/SP Remote Spooling Communications Subsystem Networking (RSCS Networking) Version 2

Planning and Installation, SH24-5057
Operation and Use, SH24-5058
Diagnosis Reference, LY24-5228

VM/SP Data Areas and Control Block Logic,

Volume 1 Control Program (CP), LY24-5220
Volume 2 Conversational Monitor System (CMS), LY24-5221

VM/SP HPO Data Areas and Control Block Logic - CP, LY20-0896

VM/SP System Logic and Problem Determination,

Volume 1 Control Program (CP), LY20-0892
Volume 2 Conversational Monitor System (CMS), LY20-0893

VM/SP HPO System Logic and Problem Determination - CP, LY20-0897

IBM 3850 Mass Storage System (MSS)

Introduction and Preinstallation Planning, GA32-0038
Principles of Operation: Theory, GA32-0035
Principles of Operation: Reference, GA32-0036

OS/VS Mass Storage System (MSS) Services:

General Information, GC35-0016
Reference Information, GC35-0017

OS/VS Message Library: Mass Storage System (MSS) Messages, GC38-1000

Operator's Library: IBM 3850 Mass Storage System (MSS) Under OS/VS, GC35-0014.

OS/VS Data Management Macro Instructions, GC26-3793

OS/VS Supervisor Service and Macro Instructions, GC27-6979

If you use the IBM 3767 Communication Terminal as a virtual machine console, the *IBM 3767 Operator's Guide*, GA18-2000 may also be helpful.

Special Characters

+ and - subcommands of DUMPSCAN command
  See DIAG
(VTOC) Volume Table of Contents
  See CMSPROG
&name subcommand of DUMPSCAN command
  See DIAG
bclose
  See CMSPROG
bdump
  See CMSPROG
bopen
  See CMSPROG
bopenr
  See CMSPROG
bopnlb
  See CMSPROG
bopnr2
  See CMSPROG
bopnr3
  See CMSPROG
bosvlt
  See CMSPROG
*BLOCKIO (DASD Block I/O System Service) 244
*CCS (SNA Console Communication Services) 219
*LOGREC (Error Logging System Service) 257
*MSG (Message System Service) 239
*MSGALL (Message All System Service) 241
*NCCF 343, 355, 356
*SIGNAL (Signal System Service) 252
*SPL (Spool System Service 259
/JOB control cards
  See CMSPROG
? subcommand of DUMPSCAN command
  See DIAG

A

abend
  See abnormal termination (abend)
ABEND macro
  See DIAG
ABEND macro (SVC 13)
  See CMSPROG
abend messages
  See DIAG
  identification
    See DIAG
abend, reason for
  See DIAG
ABENDs
  See also CMSPROG
  See also DIAG
  programmable operator facility 320, 386
ABNEXIT macro
  See CMSPROG
abnormal termination (abend)
  See also CMSPROG
  programmable operator facility 320, 386
abnormal termination procedures
  See DIAG
ACCEPT
  IUCV function 111, 137
    parameter list format 139
    trace table entry format 194
    using 137
  logical device support facility function 63, 65, 282
ACCESS command
  See CMSPROG
access diagnostic information saved for protected application facility users 92
access method services
  See CMSPROG
access method, OS, support of
  See CMSPROG
account number, replacing directory entry 75
accounting
  See also CPPROG
  activating the TOD-clock interface 57
  records, generating 36
  VM SNA support 231
accounting records 36
ACF/VTAM, VM/SP SNA support 219
action routines 318, 386
  See also programmable operator facility
  call interface 387
  DMSPOL 396
  DMSPOR 393
    GET 393
    LGLOPR 393
    QUERY 393
    SET 393
    STOP 393
    TOFB 393

TOVM 393
DMSPOS 394
error message and response handling 391
EXEC 390
handling console I/O 392
interface 387
parameter interface 387
supplied 392
writing 390
activating the TOD-clock accounting interface 57
active disk table (ADT)
See CMSPROG
adding national language information for an application 407
ADSTOP command
See DIAG
ADT (active disk table)
See CMSPROG
ALL subcommand of TRAPRED command
See DIAG
alter contents of storage
See DIAG
altering storage contents
See DIAG
alternate userid, DIAGNOSE code X'D4' 106
APAR command
See DIAG
APARs (Authorized Program Analysis Reports)
See DIAG
APPC/VM synchronous event (type X'0C') entry
See DIAG
applications, VMCF 285
AREGS subcommand of DUMPSCAN command
See DIAG
ARIOBLOK subcommand of DUMPSCAN command
See DIAG
assembler language macros
See CMSPROG
assembler language programs
See CMSPROG
assembler virtual storage
See CMSPROG
ASSGN command
See CMSPROG
assigning logical operator 334
LGLOPR action routine 393
LGLOPR command 334
ATTACH macro (SVC 42)
See CMSPROG
attached processor mode (AP)
See also CPPROG
examine real storage 8
AUTHORIZE function of VMCF 293
automatic invocation of the programmable operator 323
auxiliary directories
See CMSPROG
auxiliary files
See CMSPROG
AUXPROC option of FILEDEF command
See CMSPROG
avoiding IUCV external interrupts 116

B

batch facility
See CMSPROG
BATEXIT1 routine
See CMSPROG
BATEXIT2 routine
See CMSPROG
BATLIMIT macro
See CMSPROG
BDAM
See CMSPROG
BEGIN command
See DIAG
bits for licensed program identification 6
BLDL macro (SVC 18)
See CMSPROG
BLIP character
See CMSPROG
*BLOCKIO 244
BOTTOM subcommand of TRAPRED command
See DIAG
BPAM
See CMSPROG
branch entry Freemain (type X'0B') entry
See DIAG
branch entry Getmain (type X'0A') entry
See DIAG
breakpoint setting
See DIAG
BSAM/QSAM
See CMSPROG
BSP macro (SVC 69)
See CMSPROG

C

C subcommand of DUMPSCAN command
See DIAG
calling IBM for assistance, data needed
See DIAG
inquiry data sheet
See DIAG
CANCEL function of VMCF 295
*CCS 219
CHAIN subcommand of DUMPSCAN command
See DIAG
changes, summary of 431
changing a logical operator 334
LGLOPR action routine 393
LGLOPR command 334

channel program modification 30
channel program support, real 89
CHAP macro (SVC 44)
  See CMSPROG
CHECK macro
  See CMSPROG
CHKPT macro (SVC 63)
  See CMSPROG
class override file
  See CPPROG
classes, types, models, and features of devices (DIAGNOSE code X'24') 27, 419
classes, user privilege
  See CPPROG
clear error recording cylinders 24
CLOSE function of SPOOL system service 265
CLOSE/TCLOSE macro (SVC 20/23)
  See CMSPROG
CMD command, programmable operator 369
CMD option of the PER command
  See DIAG
CMS (Conversational Monitor System)
  abnormal termination
    See CMSPROG
  commands
    See CMS (Conversational Monitor System) commands
  IUCV 199
    between two virtual machines 212
    CMSIUCV macro 204
    example 212
    exits 211
    guidelines and limitations 215
    HNDIUCV macro 199
    sample program 425
  macros
    See CMSPROG
  message repository
    updating 409
  modules
    See CMSPROG
  sequence of functions 212
  storage
    See CMSPROG
CMS (Conversational Monitor System) commands
  immediate commands 394
  LANGGEN 413
  LANGMERG 411
  RESERVE 246
CMS abend dump reading
  See DIAG
CMS abend recovery function
  See DIAG
CMS control block relationship
  See DIAG
CMS debugging
  See DIAG
CMS dump file printing
  See DIAG
CMS IUCV 199-217
  exits 211
  sample program 425
CMS subcommand of DUMPSCAN command
  See DIAG
CMS/DOS
  See CMSPROG
CMSGEND EXEC and function 321
CMSIUCV macro 204
  complex list format (MF=(L,addr[,label])) 208
  execute format (MF=(E,addr)) 208
  list format (MF=L) 207
  standard format 205
CMSPOINT subcommand of DUMPSCAN command
  See DIAG
CMSPROG
  See the VM/SP CMS for System Programming manual
coding conventions
  See CPPROG
collecting CP data
  See DIAG
collecting virtual machine data
  See DIAG
command access, CP
  See CPPROG
command handling, SNA CCS 228
commands
  DIAL used with VSCS 227
  immediate commands 394
  LANGGEN 413
  LANGMERG 411
  programmable operator 369
    CMD 369
    FEEDBACK 371
    GET 372
    LGLOPR 373
    LOADTBL 375
    LOG 377
    QUERY 379
    SET 382
    STOP 385
  RESERVE 246
  SEND used with single console image facility 279
  SET EMSG 51
  SMSG 277
  TERMINAL
    BREAKIN GUESTCTL 49
    BRKKEY 49
    CONMODE 3270 48
    SCRNSAVE OFF 48
    SCRNSAVE ON 48
communication
  between the programmable operator and the network 320
  between virtual machines 111, 283

checking for the programmable operator 359
establishing SNA links 225
example 117
interfaces of SNA 223
IUCV
DASD block I/O system service 243
error logging system service 257
message system service 239
signal system service 251
spool system service 259
PMX protocol 331
programmable operator 325
programmable operator and NCCF or NetView 330
complex list format (MF=(L,addr[,label]))
CMSIUCV macro 208
HNDIUCV macro 202
condition codes
DIAGNOSE code
X'A0' 92
X'BC' 101
X'B4' 95
X'B8' 99
X'D0' 105
X'0C' 13
X'00' 7
X'08' 10
X'10' 13
X'14' 14
X'18' 24
X'28' 31
X'3C' 35
X'30' 33
X'34' 33
X'38' 34
X'4C' 39
X'58' 42
X'6C' 57
X'64' 54, 55
X'7C' 64
X'78' 61
X'80' 70
X'84' 78
X'94' 86
FINDSYS function 55
IUCV
ACCEPT 138
CONNECT 133
DECLARE BUFFER 129
DESCRIBE 181
PURGE 165
QUERY 127
QUIESCE 174
RECEIVE 150
REJECT 162
REPLY 156
RESUME 177
RETRIEVE BUFFER 172
SEND 145
SET CONTROL MASK 192
SET MASK 189
SEVER 169
TEST COMPLETION 185
TEST MESSAGE 179
LOADSYS function 54
PURGESYS function 55
configuration file for GCS
See DIAG
CONNECT function of IUCV 131
parameter list format 134
to the DASD block I/O system service 244
to the signal system service 252
to the SPOOL system service 260
trace table entry format 194
using 131
connection complete external interrupt in IUCV 140
connection pending external interrupt in IUCV 135
connection quiesced external interrupt in IUCV 175
connection resumed external interrupt in IUCV 178
connection severed external interrupt in IUCV 170
console function, virtual 9
console, single 279
contents of the feature tape for a national language 401
control blocks
See CMSPROG
control file
LANGGEN 415
example 415
updating 409
LANGMERG 405, 412
example 413
control functions of VMCF 293
control paths in IUCV 121
Control Program (CP)
commands
See CP (Control Program) commands
device classes, types, models, and features (DIAGNOSE code X'24') 27, 419
message repository 404
DASD space for 404, 405
saving with DIAGNOSE code X'CC' 103
system service
DASD block I/O 243
error logging 257
IUCV communication 197
message 239
message all 241
signal 251
SNA virtual console communication 219
spool 259
control program, 370X 39
control register allocation
See DIAG
control registers

See DIAG
controlling authorization 352
Conversational Monitor System (CMS)
abnormal termination
See CMSPROG
commands
See CMS (Conversational Monitor System) commands
IUCV 199
between two virtual machines 212
CMSIUCV macro 204
example 212
exits 211
guidelines and limitations 215
HNDIUCV macro 199
sample program 425
macros
See CMSPROG
message repository
updating 409
modules
See CMSPROG
sequence of functions 212
storage
See CMSPROG
CONVERT command
See DIAG
converting routing tables 429
CONVIPCS EXEC
See DIAG
CORTABLE subcommand of DUMPSCAN command
See DIAG
COUNT subcommand of the PER command
See DIAG
CP (Control Program)
See also CPPROG
commands
See CP (Control Program) commands
device classes, types, models, and features (DIAGNOSE code X'24') 27, 419
message repository 404
DASD space for 404, 405
saving with DIAGNOSE code X'CC' 103
system service
DASD block I/O 243
error logging 257
IUCV communication 197
message 239
message all 241
signal 251
SNA virtual console communication 219
spool 259
CP (Control Program) commands
DIAL used with VSCS 227
SEND used with single console image facility 279
SET EMSG 51
SMSG 277
TERMINAL
BREAKIN GUESTCTL 49
BRKKEY 49
CONMODE 3270 48
SCRNSAVE OFF 48
SCRNSAVE ON 48
CP abend dumps, reading
See DIAG
CP data, recording
See DIAG
CP debugging
See DIAG
CP FRET Trap
See DIAG
CP internal trace table
See DIAG
CP SET DUMP command
See DIAG
CP trace table entries, recording
See DIAG
CPEREP program
See DIAG
CPPROG
See the CP for System Programming manual.
CPTRAP command
See DIAG
CPTRAP facility
See DIAG
CSIYTD control program
See DIAG
CUSTOMER PROFILE file
See DIAG
CVTSECT (CMS Communications Vector Table)
See DIAG

D

DASD Block I/O System Service 243-249
establishing communications 243
from CMS 246
IUCV communication 248
IUCV CONNECT 244
IUCV SEND 245
DASD Dump/Restore (DDR) program
See DIAG
DASD I/O function 22
DASD space for CP message repository 404, 405
data extraction routine
See DIAG
data management
See DIAG
data needed before calling IBM for assistance
See DIAG
data security in batch facility
See CMSPROG

Data Set Control Block (DSCB)
See CMSPROG
data sets, DOS
See CMSPROG
data sheet, problem inquiry
See DIAG
data transfer error codes, VMCF 312
data transfer functions of VMCF 296
DCB (data control block)
See CMSPROG
DCB macro
See CMSPROG
DCP command
See DIAG
DCSS
See discontiguous saved segment (DCSS)
DCSS (discontiguous shared segment)
See CMSPROG
DDR command
See CMSPROG
DDR program
See DIAG
debug mode, programmable operator 398
debugging a dump
See DIAG
debugging an AP/MP system
See DIAG
debugging CMS
See DIAG
debugging CP
See DIAG
debugging GCS
See DIAG
debugging the virtual machine
See DIAG
debugging tools summary
See DIAG
debugging TSAF
See DIAG
debugging, introduction
See DIAG
declarative macros, DOS
See CMSPROG
DECLARE BUFFER function of IUCV 128
parameter list format 129
trace table entry format 194
using 128
default logical operator 317
DELETE macro (SVC 9)
See CMSPROG
deleting a national language 410
DEQ macro (SVC 48)
See CMSPROG
DESCRIBE function of IUCV 181
parameter list format 182
trace table entry format 194
using 181
descriptions of IUCV macro parameters 124
DETACH macro (SVC 62)
See CMSPROG
determining DASD space for CP message repository 404
determining virtual machine storage size 52
device classes, types, models, and features (DIAGNOSE code X'24') 27, 419
device table (DEVTAB)
See CMSPROG
device type class and values
See DIAG
devices
feature codes 423
model codes 422
type codes 419
DEVTAB (device table)
See CMSPROG
DEVTYPE macro (SVC 24)
See CMSPROG
DIAG
See the VM Diagnosis Guide
DIAGNOSE code
instruction use 3
X'A0', retrieve a group name 92
X'BC', open a spool file 100
X'B0', access diagnostic information saved for protected application facility users 92
X'B4', virtual printer external attribute buffer manipulation 94
X'B8', spool file external attribute buffer manipulation 98
X'CC', saving the CP message repository 103
X'C8', set language 101
X'D0', provide 3480 tape volume serial number 105
X'D4', specify an alternate userid 106
X'D8', system spool information 108
X'0C', pseudo timer 12
X'00', store extended-identification code 4
licensed program identification bits 6
X'04', examine real storage 7
X'08', virtual console function 9
X'1C', clear error recording cylinders 24
X'10', release pages 13
X'14', input spool file manipulation 14
X'18', standard DASD I/O 22
X'2C', start of LOGREC area 31
X'20', general I/O 25
X'24', device type and features 27
device classes, types, models, and features 419
X'28', channel program modification 30
X'3C', VM directory 34
X'30', read LOGREC data 32
X'34', read system dump spool file 33
X'38', read system symbol table 34
X'4C', generate accounting records for the virtual user 36
X'40', clean-up after virtual IPL by device 35
X'48', issue SVC 76 from a second level machine 36

X'5C', error message editing 50
X'50', save the 370X control program image 39
X'54', control function of the PA2 function key 40
X'58' 40
display data on 3270 console screen 41
3270 virtual console interface, full screen interactions 45
3270 virtual console interface, full screen interactions (3270 SIO) 48
3270 virtual console interface, full screen mode 43
X'6C', shadow table maintenance 57
X'60', determine virtual machine storage size 52
X'64', finding, loading, purging named segments 52
FINDSYS function 55
LOADSYS function 53
PURGESYS function 54
X'68', VMCF function 55, 283, 300
X'7C', logical device support facility 61
X'70', activating TOD-clock accounting interface 57
X'74', saving or loading a 3800 named system 59
X'78', MSS communication 60
X'8C', access device dependent information 80
X'80', MSSFCALL 69
X'84', directory update in-place 72
X'94', VMDUMP Function 81
X'98', real channel program support 89
DIAGNOSE code interface with a DCSS
See CPPROG
DIAGNOSE code interface with named segments
See CPPROG
DIAGNOSE instruction 3
access device dependent information 80
access diagnostic information saved for protected application facility users 92
activating the TOD-clock accounting interface 57
channel program modification 30
clean-up after virtual IPL by device 35
clear error recording 24
control function of the PA2 function key 40
determine virtual machine storage size 52
device type and features 27
directory update in-place 72
display data on 3270 console screen 41
error message editing 50
examine real storage 7
finding, loading, purging named segments 52
FINDSYS function 55
format 3, 300
general I/O 25
generate accounting records for the virtual user 36
input spool file manipulation 14
issue SVC 76 from a second level virtual machine 36
logical device support facility 61
MSS communication 60
MSSFCALL 69
open a spool file 100
page release function 13
provide 3480 tape volume serial number 105
pseudo timer 12
PURGESYS function 54
read LOGREC data 32
read system dump spool file 33
read system symbol table 34
real channel program support 89
retrieve a group name 92
save the 370X control program image 39
saving or loading a 3800 named system 59
saving the CP messages repository 103
set language 101
shadow table maintenance 57
specify an alternate userid 106
spool file external attribute buffer manipulation 98
standard DASD I/O 22
start of LOGREC area 31
store extended-identification code 4
system spool information 108
update VM directory 34
virtual console function 9
virtual printer external attribute buffer manipulation 94
VMCF function 55, 283, 300
data transfer error codes 312
return codes 309
VMCPARM parameter list 301
VMDUMP Function 81
3270 virtual console interface
full screen interactions 45, 61
full screen interactions (3270 SIO) 48
full screen mode 43
diagnosing problems
See DIAG
diagnostic information saved for protected application facility users 92
DIAL command used with VSCS 227
directory
authorization for IUCV 116
control statement for IUCV 111, 114, 118, 132, 137
entries for national languages, updating 407
entries in IUCV 116, 132, 138
reading 34
replacing entries 72
update in-place 72
updating with DIAGNOSE code X'3C' 34
discontiguous saved segment (DCSS)
loading 53
purging 54

These symbols are used in the index to refer to other VM and VM/SP books:
CPPROG—CP for System Programming (for VM/SP or VM/SP HPO)
CMSPROG—VM/SP CMS for System Programming
DIAG—VM Diagnosis Guide

discontiguous saved segments (DCSS)
See CPPROG
discontiguous shared segment (DCSS)
See CMSPROG
Disk Operating System (DOS)
See CMSPROG
dispatcher (type X'01') entry
See DIAG
dispatching priority, replacing directory entry 74
dispatching virtual machines
See CPPROG
DISPLAY command
See DIAG
display real CP data
See DIAG
DISPLAY subcommand of DUMPSCAN command
See DIAG
display terminals, CMS interface
See CMSPROG
display virtual data
See DIAG
displaying
data on a 3270 console screen 41
DISPW macro
See CMSPROG
distributed system use of the programmable operator 316
distribution word, replacing directory entry 75
DMMTAB communication table
See DIAG
DMSABN macro
See CMSPROG
See DIAG
DMSEXS macro
See CMSPROG
DMSFRE service routines
See CMSPROG
DMSFREE macro
See CMSPROG
DMSFRES macro
See CMSPROG
DMSFRET macro
See CMSPROG
DMSFST macro
See CMSPROG
DMSINA module
See CMSPROG
DMSINT module
See CMSPROG
DMSIOW module
See CMSPROG
DMSITE module
See CMSPROG
DMSITI module
See CMSPROG
DMSITP module
See CMSPROG
DMSITP routine
See DIAG
DMSITS module
See CMSPROG
DMSKEY macro
See CMSPROG
DMSNUC
See CMSPROG
DMSPOL programmable operator action routine 396
DMSPOR programmable operator action routine 393
DMSPOS programmable operator action routine 394
DMSTVS module
See CMSPROG
DMSXFLPT XEDIT routine
See CMSPROG
DMSXFLRD XEDIT routine
See CMSPROG
DMSXFLST XEDIT routine
See CMSPROG
DMSXFLWR XEDIT routine
See CMSPROG
DOS (Disk Operating System)
See CMSPROG
DOSPOINT subcommand of DUMPSCAN command
See DIAG
DOWN subcommand of TRAPRED command
See DIAG
DSCB (Data Set Control Block)
See CMSPROG
DTFCD macro
See CMSPROG
DTFCN macro
See CMSPROG
DTFDI macro
See CMSPROG
DTFMT macro
See CMSPROG
DTFPR macro
See CMSPROG
DTFSD macro
See CMSPROG
dump address parameter list 85
DUMP command
See DIAG
dump debugging
See DIAG
dump spool file, read 33
dump, used in problem determination
See DIAG
DUMPID subcommand of DUMPSCAN command
See DIAG
dumping to DASD
See DIAG
dumping to printer
See DIAG
dumping to tape
See DIAG
DUMPSCAN command and subcommands
See DIAG

DUMPSCAN scroll interface
See DIAG
dynamic linkage
See CMSPROG
dynamic load overlay
See CMSPROG

E

editing error messages with DIAGNOSE code X'5C' 50
EMSG setting 50, 51
enabling SNA terminals 225
END subcommand of DUMPSCAN command
See DIAG
end, abnormal
See abnormal termination (abend)
ENQ macro (SVC 56)
See CMSPROG
ensuring a complete log 356
in a distributed system 357
in a mixed environment 358
in a single system 357
entry points
See CMSPROG
environments
programmable operator 315
distributed VM 316
mixed 316
single system 315
SNA 221
EPLIST (extended PLIST)
See CMSPROG
EPLIST macro
See CMSPROG
error codes for DMSFREE, DMSFRES, DMSFRET
See CMSPROG
Error Logging System Service 257-258
establishing communications with 257
error message editing with DIAGNOSE code X'5C' 50
error recording cylinders, clear 24
establishing communication
DASD block I/O system service 243
error logging system service 257
message system service 239
programmable operator and NCCF or NetView 330
programmable operator to IUCV 325
signal system service 251
spool system service 259
establishing SNA communications links 225
ETRACE command
See DIAG
ETRACE GROUP
See DIAG
examine real storage with DIAGNOSE code X'04' 7
example of IUCV virtual machine communication 117
examples
DIAGNOSE code X'08' 11
programmable operator 363
in a distributed environment 364
in a local environment 363
in a mixed environment 365
EXCP, CMS/DOS support for
See CMSPROG
EXEC action routines 390
EXEC procedures
See CMSPROG
execute format (MF=(E,addr))
CMSIUCV macro 208
HNDIUCV macro 203
exit EXECs, programmable operator 396
communication error 397
PROPHCHK 397
PROPPCHK 397
interface 396
log error 397
PROPLGER 397
EXIT/RETURN macro (SVC 3)
See CMSPROG
extended control PSW description
See DIAG
extended-identification code 4
External Attribute Buffer (XAB) 94
manipulation of spool file 98
manipulation of virtual printer 94
READ-XAB function of SPOOL system service 272
external interrupt
See also CPPROG
functions controlling 125
in VMCF 283, 305
message header 306
IUCV 114
avoiding 116
connection complete 140
connection pending 135
connection quiesced 175
connection resumed 178
connection severed 170
control 115
enabling or disabling 115
message 115
message complete 158
message pending 147
external interrupt (type X'02') entry
See DIAG
external interrupt, CMS
See CMSPROG
external tracing facilities, GCS
See DIAG
EXTRACT macro (SVC 40)

See CMSPROG

F

FCB (file control block)
See CMSPROG
FDISPLAY subcommand of DUMPSCAN command
See DIAG
feature tape, national language 401
listing files 402
object files 403
source files 402
features, device 423
features, types, models, and classes of devices (DIAGNOSE code X'24') 27, 419
FEEDBACK command, programmable operator 371
feedback file 358
See also programmable operator facility
FEOV macro (SVC 31)
See CMSPROG
fetch-protected storage, not dumped 81
file control block (FCB)
See CMSPROG
FILEDEF command
See also CMSPROG
options
See CMSPROG
to invoke the programmable operator 322
files
for a language
listing 402
object 403
source 402
updating
for a national language 409
filtering messages 350
FIND macro (SVC 18)
See CMSPROG
finding
saved systems 55
FINDSYS function 55
FOB (font offset buffer)
See CPPROG
foreign languages
See languages, national
format of DIAGNOSE instruction 3
FORMAT subcommand of TRAPRED command
See DIAG
free storage, CMS
See CMSPROG
FREEDBUF macro (SVC 56)
See CMSPROG
FREELOWE
See CMSPROG
FREEMAIN macro (SVC 5)
See CMSPROG
Freemain via SVC (type X'09') entry
See DIAG
FREEWORK (DMKFRE and DMKFRT save area)
See DIAG
FSCB (file system control block)
See CMSPROG
FSCB macro
See CMSPROG
FST (file status table)
See CMSPROG
functional descriptions of IUCV macro parameters 124
functions, SNA 224

G

G subcommand of DUMPSCAN command
See DIAG
GCS (Group Control System)
and the programmable operator facility 316, 328
national language files, saving 407
GCS configuration file
See DIAG
GCS debugging
See DIAG
GCS dumping facilities
See DIAG
GCS dumps, analyzing
See DIAG
GCS dumps, initiating
See DIAG
GCS external tracing facilities
See DIAG
GCS internal trace table
See DIAG
GCS internal trace table formats
See DIAG
GDUMP command
See DIAG
GENDIRT command
See CMSPROG
general I/O function 25
generate accounting records with DIAGNOSE code X'4C' 36
GENIMAGE command
See CPPROG
GENIMAGE service program
See CPPROG
GENMOD command
See DIAG
GET
action routine 393
command, programmable operator 372
GET macro
See CMSPROG
GETMAIN macro
See CMSPROG

GETMAIN macro (SVC 4)
See CMSPROG
Getmain via SVC (type X'08') entry
See DIAG
GETPOOL/FREEPOOL macro
See CMSPROG
getting national languages on your system 401-415
deleting a language 410
feature tape 401
installing 404
LANGGEN command 413
LANGMERG command 411
loading the files from disk to tape 404
saving files for CP and CMS 404
updating files for an existing language 409
Group Control System (GCS)
and the programmable operator facility 316, 328
national language files, saving 407
group name, retrieve with DIAGNOSE code X'A0' 92
group, virtual machine 251
GTF header
See DIAG
GTRACE (type X'0E') entry
See DIAG
GTRACE macro
See DIAG
GUESTR option of PER command
See DIAG
GUESTV option of PER command
See DIAG
guidelines and limitations of CMS IUCV 215

H

HELP subcommand of DUMPSCAN command
See DIAG
helpful hints, programmable operator 367
HEX subcommand of TRAPRED command
See DIAG
HNDIUCV macro 199
complex list format (MF=(L,addr[,label])) 202
execute format (MF=(E,addr)) 203
list format (MF=L) 202
standard format 200
HOSTCHK statement 339
See also programmable operator facility
HX subcommand of DUMPSCAN command
See DIAG

I

I/O (input/output)
function
DASD 22
general 25
reduction, VTAM 231
VM SNA support
processing 229, 230
I/O interrupt (type X'03') entry
See DIAG
identification bits for licensed programs 6
IDENTIFY
VMCF function 296
VMCF protocol 292
IDENTIFY macro (SVC 41)
See CMSPROG
IIP (ISAM Interface Program)
See CMSPROG
immediate commands in CMS 394
imperative macros
See CMSPROG
INDICATE command
See DIAG
Initial Program Load (IPL)
device, replacing directory entry 75
initializing the programmable operator 324
INITIATE, logical device support facility function 61, 62, 65, 282
input spool file manipulation with DIAGNOSE code X'14' 14
input/output
See I/O (input/output)
installing
national languages on VM 404
PMX 328
programmable operator 321
instruction
See DIAGNOSE instruction
Inter-User Communications Vehicle (IUCV) 111
basic communication functions 125
CMS 199
CMS, between two virtual machines 212
communication
DASD block I/O system service 243
error logging system service 257
example 117
message system service 239
signal system service 251
spool system service 259
using control paths 121
using data in a buffer 117
using data in a parameter list 120
with CP system services 197
with DASD Block I/O 248
connection complete 140

connection pending 135
connection quiesced 175
connection resumed 178
connection severed 170
control paths 121
DASD block I/O system service, use 243
Error Logging System Service, use 257
external interrupt 114
avoiding 116
control 115
enabling or disabling 115
message 115
functional terminology 124
functions
See IUCV functions
functions controlling external interrupts 125
introduction 111
invoking 123
macro description 123
Message All System Service 241
message complete 158
message pending 147
Message System Service use 239
messages 111, 112
data transfer 112
identification 113
parameters, specifying 123
paths 111
control 121
programmable operator 325
receiving messages from the special message facility 278
security considerations 116
sequence of functions 117, 120
signal system service, use 251
SNA environment use 223
special message facility 278
Spool System Service, use 259
system services 197
DASD block I/O 243
error logging 257
message 239
message all 241
signal 251
SNA console communication services 221
spool 259
trace table entries 193
field definitions 195
formats 194
suppressing 193
Interactive Problem Control System (IPCS)
See DIAG
internal trace table formats, GCS
See DIAG
internal trace table, CP
See DIAG
internal trace table, GCS
See DIAG
internal trace table, TSAF
See DIAG
internal tracing facilities, GCS
See DIAG
interrupt handler, DMSITI module
See CMSPROG
interrupt handling
See CPPROG
interrupt handling, CMS
See CMSPROG
interrupt, external
functions controlling 125
IUCV 114
avoiding 116
control 115
enabling or disabling 115
message 115
INTSVC for SVC handling routine
See CMSPROG
invoking
IUCV functions 123
programmable operator facility 322
automatically 323
commands 361
manually 322
VMCF functions 300
IPCS (Interactive Problem Control System)
See DIAG
IPCS interface files
See DIAG
IPCS variables
See DIAG
IPCSDUMP command
See DIAG
IPCSMAP subcommand of DUMPSCAN command
See DIAG
IPL (Initial Program Load)
See also CPPROG
device, replacing directory entry 75
IPL performance using saved system
See CMSPROG
ISAM Interface Program (IIP))
See CMSPROG
ITRACE command
See DIAG
IUCV (Inter-User Communications Vehicle) 111
basic communication functions 125
CMS 199
CMS, between two virtual machines 212
communication
DASD block I/O system service 243
error logging system service 257
example 117
message system service 239
signal system service 251
spool system service 259
using control paths 121
using data in a buffer 117
using data in a parameter list 120
with CP system services 197
with DASD Block I/O 248

connection complete 140
connection pending 135
connection quiesced 175
connection resumed 178
connection severed 170
control paths 121
DASD block I/O system service, use 243
Error Logging System Service, use 257
external interrupt 114
avoiding 116
control 115
enabling or disabling 115
message 115
functional terminology 124
functions
See IUCV functions
functions controlling external interrupts 125
introduction 111
invoking 123
macro description 123
Message All System Service 241
message complete 158
message pending 147
Message System Service use 239
messages 111, 112
data transfer 112
identification 113
parameters, specifying 123
paths 111
control 121
programmable operator 325
receiving messages from the special message facility 278
security considerations 116
sequence of functions 117, 120
signal system service, use 251
SNA environment use 223
special message facility 278
Spool System Service, use 259
system services 197
DASD block I/O 243
error logging 257
message 239
message all 241
signal 251
SNA console communication services 221
spool 259
trace table entries 193
field definitions 195
formats 194
suppressing 193
IUCV control paths 121
IUCV functions 111
ACCEPT 111
parameter list format 139
using 137
CONNECT 111
parameter list format 134
to the SPOOL system service 260
using 131
DECLARE BUFFER, using 128
parameter list format 129
DESCRIBE 112
parameter list format 182
using 181
PURGE
parameter list format 165
using 164
QUERY, using 127
QUIESCE 112
parameter list format 174
using 173
RECEIVE 112
parameter list format 151
using 149
REJECT 112
parameter list format 162
using 161
REPLY 112
parameter list format 156
using 154
RESUME 112
parameter list format 177
using 176
RETRIEVE BUFFER, using 172
SEND 112
parameter list format 145
to the DASD block I/O system service 245
to the signal system service 254
to the SPOOL system service 261
using 142
SET CONTROL MASK 115, 191
parameter list format 192
using 191
SET MASK 115, 188
parameter list format 189
using 188
SEVER 111
parameter list format 169
using 168
terminology 124
TEST COMPLETION 112
parameter list format 185
using 184
TEST MESSAGE, using 179
IUCV subcommand of DUMPSCAN command
See DIAG

J

job control cards (/JOB)
  See CMSPROG
journaling
  See CPPROG

K

keys
  See CMSPROG

L

LANGGEN command 409, 413
  control file 415
    example 415
    updating 409
  format 413
LANGMERG command 405, 408, 411
  control file 405, 412
    example 413
  format 411
languages, national 401, 415
  adding information for an application 407
  deleting 410
  deleting application information 410
  feature tape 401
  installing on VM 404
  LANGGEN command 413
  LANGMERG command 411
  loading the files from tape to disk 404
  saving files for CP and CMS 404
  setting, DIAGNOSE code X'C8' 101
  update user directory entries for 407
  updating
    files for an existing language 409
    message repositories 409
leaving the signal system service 255
LGLOPR
  See also programmable operator facility
  action routine 393
  command 393
  statement 337
libraries
  See CMSPROG
licensed program identification bits 6
limitations of CMS IUCV 215
LINK command
  See CPPROG
LINK macro (SVC 6)
  See CMSPROG
LIOCS routines supported by CMS/DOS
  See CMSPROG
list format (MF=L)
  CMSIUCV macro 207
  HNDIUCV macro 202
listing files, national language 402
LOAD command
  See DIAG
LOAD macro (SVC 8)
  See CMSPROG
load map generation
  See DIAG
load maps
  See DIAG
load maps, CMS
  See CMSPROG
loader
  See CMSPROG
loader tables in CMS
  See CMSPROG
loading
  a 3800 named system with DIAGNOSE code X'74' 59
  national language files for CP and CMS 404
loading discontiguous saved segments 53
LOADSYS function 53
LOADTBL command, programmable operator 375
LOCATE (UP) subcommand of DUMPSCAN command
  See DIAG
LOG command, programmable operator 377
log file 354
  See also programmable operator facility
LOGGING statement 340
  See also programmable operator facility
logical device support facility 281, 282
  description 281
  implementing via DIAGNOSE code X'7C' 61
logical editing symbols, replacing directory entry 74
logical operator 317, 318, 332
  action routine 393
  assigning, releasing, replacing 334
  command 334, 373, 393
  compared to the CP system operator 332
  default 317, 333
  NCCF or NetView 334
  statement 337
logical units
  See CMSPROG
LOGON command
  See CPPROG
*LOGREC 257
LOGREC area
  getting starting address 31
  reading 32
loop procedures
  See DIAG
looping programs
  See DIAG

M

machine check
  See CPPROG
machine check interrupts in CMS
  See CMSPROG
macros
  See also CPPROG
  CMS
    CMSIUCV 204
    HNDIUCV 199
  CP
    IUCV, description 123
  IUCV 111
    ACCEPT 137
    advantages of using 124
    CONNECT 131
    DECLARE BUFFER 128
    DESCRIBE 181
    PURGE 164
    QUERY 127
    QUIESCE 173
    RECEIVE 149
    REJECT 161
    REPLY 154
    RESUME 176
    RETRIEVE BUFFER 172
    SEND 142
    SET CONTROL MASK 191
    SET MASK 188
    SEVER 168
    terminology 124
    TEST COMPLETION 184
    TEST MESSAGE 179
MAINHIGH
  See CMSPROG
management of data
  See DIAG
management of problems
  See DIAG
manual invocation of the programmable operator 322
MAP command
  See DIAG
map compressing routine
  See DIAG
MAP option of GENMOD command
  See DIAG
MAP option of LOAD command
  See DIAG
MAPA subcommand of DUMPSCAN command
  See DIAG
MAPN subcommand of DUMPSCAN command
  See DIAG
Mass Storage System (MSS)
  communication 60
  mount and demount processing 60
Message All System Service 241
message complete external interrupt, IUCV 158
MESSAGE function of SPOOL system service 268
message output format, programmable operator 327
message pending external interrupt, IUCV 147
message repository
  for CMS
    updating 409
  for CP 103, 404
    DASD space for 404, 405
    saving 103
    updating 409
  updating 409
Message System Service 239-240
  establishing communications 239
messages
  data transfer, IUCV 112
  identification, IUCV 113
  IUCV 111, 112
minidisk link mode, replacing directory entry 77
minidisk multiple password, replacing directory entry 77
minidisk read password, replacing directory entry 77
minidisk write password, replacing directory entry 77
minidisks
  See CMSPROG
mixed environment use of the programmable operator 316
model, device 422
models, classes, types, and features of devices (DIAGNOSE code X'24') 27, 419
modifying channel program 30
MODMAP command
  See DIAG
module load map
  See DIAG
MONITOR command
  See DIAG
MONITOR START command
  See DIAG
MORE/HOLDING condition, SNA handling 231
MOVEFILE command
  See CMSPROG
MREGS subcommand of DUMPSCAN command
  See DIAG
MRIOBLOK subcommand of DUMPSCAN command
  See DIAG
*MSG 239
*MSGALL 241
MSS (Mass Storage System)
  See also CPPROG
  communication 60
  mount and demount processing 60
MSSF SCPINFO command 69
MSSFCALL 69

SCPINFO command 69
multiple address stops
See DIAG
multiprocessor
See also CPPROG
examine real storage 8
multiprocessor mode (MP)
See CPPROG

N

named segments, finding, loading, purging 52
named system
saving or loading a 3800 59
national languages 401
native languages
See languages, national
*NCCF 343, 355, 356
NCCF (Network Communications Control Facility)
and the programmable operator facility 316, 328
logging NCCF messages 356
logical operator 334
operator 334
command authorization 330
QUERY command authorization 334
routing messages to 350
operator station 362
PMX 316, 328
when stopping programmable operator 328
NCP and PEP sharing 232
NCPDUMP command
See DIAG
NCPDUMP service program
See DIAG
NetView
and the programmable operator facility 316, 328
logging NetView messages 356
logical operator 334
operator 334
command authorization 330
QUERY command authorization 334
operator station 362
PMX 316, 328
when stopping programmable operator 328
NETWORK command
See DIAG
Network Communications Control Facility (NCCF)
and the programmable operator facility 316, 328
logging NCCF messages 356
logical operator 334
operator 334
command authorization 330
QUERY command authorization 334
routing messages to 350
operator station 362
PMX 316, 328
when stopping programmable operator 328
network dump operations
See DIAG
non-recoverable machine check
See DIAG
NOTE macro
See CMSPROG
NOTIFY function of SPOOL system service 275
nucleus free storage
See CMSPROG
nucleus load map
See DIAG
nucleus, CMS
See CMSPROG
NUCON macro
See CMSPROG

O

object files for a national language 403
open a spool file with DIAGNOSE code X'BC' 100
OPEN macro
See CMSPROG
OPEN/OPENJ macro (SVC 19/22)
See CMSPROG
Operating System (OS)
See CMSPROG
operator
NCCF station 362
NetView station 362
operator considerations, SNA 232
operator, logical 317
order of functions in CMS IUCV 212
order of functions in IUCV 117, 120
OS (Operating System)
See CMSPROG
OS/VSAM
See CMSPROG
OSPOINT subcommand of DUMPSCAN command
See DIAG
overlay structures in CMS
See CMSPROG
overview, programmable operator 315

P

page
See also CPPROG
contiguous storage
discontiguous storage 54
page management
See CMSPROG

pageable module, identify and locate
See DIAG
parameter list
IUCV
ACCEPT 139
CONNECT 134
DECLARE BUFFER 129
DESCRIBE 182
parameter list data 120
PURGE 165
QUIESCE 174
RECEIVE 151
REJECT 162
REPLY 156
RESUME 177
SEND 145
SET CONTROL MASK 192
SET MASK 189
SEVER 169
TEST COMPLETION 185
VMCPARM 301
parameters, IUCV, specifying 123
password
See also CPPROG
replacing directory entry 72, 73, 74
paths, IUCV
control, IUCV 121
PA1 program function key 49
with DIAGNOSE code X'58' 44, 47, 49
with the programmable operator facility 324, 398
with VSCS or VCNA 231
PA2 program function key 40
PEP and NCP sharing 232
PER command
See DIAG
performance of virtual machines
See CPPROG
PGRLSE macro (SVC 112)
See CMSPROG
PLIST (parameter list)
See CMSPROG
PMX (Programmable Operator/NCCF Message Exchange) 328
See also programmable operator facility
POINT macro
See CMSPROG
POST macro (SVC 2)
See CMSPROG
poster, CP internal trace table
See DIAG
PRB command
See DIAG
PRBnnnnn aaaaaaaa file
See DIAG
PRBnnnnn dump file
See DIAG
PRBnnnnn report
See DIAG
number assignment
See DIAG
PRBnnnnn report file
See DIAG
report file (PRBnnnnn REPORT)
See DIAG
PRB00003 report file with status updates added
See DIAG
PRESENT, logical device support facility function 63, 66, 282
prestructured overlays
See CMSPROG
PRINT subcommand of DUMPSCAN command
See DIAG
printer
See CPPROG
printer external attribute buffer manipulation, virtual 94
printer interrupts
See CMSPROG
PRINTER subcommand of TRAPRED command
See DIAG
printing tape dump
See DIAG
priority
messages 294, 298
privilege classes
replacing directory entry 74
privilege classes, user
See CPPROG
PROB command
See DIAG
problem exist ?
See DIAG
processing descriptions, SNA 222
processor
See CPPROG
program check debugging
See DIAG
program exceptions
See DIAG
program interrupt (type X'04') entry
See DIAG
program interrupts
See CMSPROG
Program Support Representative (PSR)
See DIAG
program temporary fix (PTF)
See DIAG
programmable operator facility 315-399
abend 320, 331, 386
action routine interface
call interface 387
parameter interface 387
action routines 318, 386
DMSPOL 396
DMSPOR 393

DMSPOS 394
error message and response handling 391
EXEC 386, 390
handling console I/O 392
supplied 392
writing 390
assigning a logical operator 334, 393
authorization 352
commands 368
CMD 369
FEEDBACK 371
GET 372
LGLOPR 373
LOADTBL 375
LOG 377
QUERY 379
SET 382
STOP 385
communication
checking 359
with the network 320
debug mode 398
default 333
ensuring a complete log 356
environments
distributed VM 316, 364
mixed 316, 365
single system 315, 363
examples of communicating
distributed environment 364
local environment 363
mixed environment 365
exit EXECs
communication error 397
interface 396
log error 397
PROPHCHK EXEC 360, 397
PROPLGER EXEC 356, 397
PROPPCHK EXEC 360, 397
feedback file 358
helpful hints 367
initializing 324
installing 321
CMSGEND EXEC 321
PMX 328
invoking 322
automatically 323
commands 361
manually 322
PROPPROF EXEC 325
PROPST EXEC 322
issuing commands 361, 363-367
LGLOPR 332
action routine 393
command 334, 373, 393
default 317
sample command entries in RTABLE 336
statement 317, 337
log file 354
logging NCCF or NetView messages 356
logical operator 318, 332
action routine 393
assigning, releasing, replacing 334
command 334, 393
default 333
NCCF or NetView 334
statement 337
message format
distributed 362
local 361
message output format 327
NCCF or NetView logical operator 334
overview 315
flow of operation 319
how it works 318
in a distributed VM system 316
in a mixed environment 316
in a single system 315
logical operator 317
partial routing table 344
PMX 316, 328
abend 331
installing 328
starting 331
stopping 331, 332
PROPEPIF EXEC 327
PROPLIB LOADLIB 321, 324
PROPPROF EXEC 325
PROPRTCV 429
QUERY command authorization for an NCCF operator 334
register conventions for invoking an action routine 389
relationship with RSCS Networking 320
releasing a logical operator 334, 393
replacing a logical operator 334, 393
restricting 353
authorization by nodeid 353
authorization by userid and nodeid 353
command use 354
routing entries to filter responses to commands 351
routing messages 357
routing table 336, 341
converting 429
entry formats and statements 337
tailoring 346
use 337
routing table (RTABLE) 317
routing table entries 341
specifying routing texts 346
routing table statements
HOSTCHK 339
LGLOPR 337
LOGGING 340
order of 341
PROPCHK 339
ROUTE 340
TEXTSYM 338

shortcuts 367
starting 324
stopping 327, 331
PROPEPIF EXEC 327
tasks 315
uncontrolled authorization 353
use in a distributed VM system 316
use in a mixed environment 316
use in a single system 315
virtual machine 321
with IUCV 325
with NCCF 316, 328, 350
command authorization 330
routing messages to 350
with NetView 316, 328, 350
routing messages to 350
Programmable Operator/NCCF Message Exchange (PMX) 328
See also programmable operator facility
PROP
See programmable operator facility
PROPCHK statement 339
See also programmable operator facility
PROPEPIF EXEC 327
PROPHCHK EXEC 397
PROPLGER EXEC 397
PROPLIB LOADLIB 321, 324
PROPPCHK EXEC 397
PROPPROF EXEC 325
PROPRTCV, for converting routing tables 429
protected application facility, DIAGNOSE code X'B0' 92
protocol
PMX communication 331
VMCF 288
IDENTIFY 292
SEND 289
SEND/RECV 290
SENDX 291
provide 3480 tape volume serial number, DIAGNOSE code X'D0' 105
PRTDUMP command
See DIAG
PSA (Prefix Storage Area)
See CPPROG
pseudo timer, DIAGNOSE code X'0C' 12
PSR (Program Support Representative)
See DIAG
PSW (program status word)
See CMSPROG
PTF (program temporary fix)
See DIAG
punch interrupts
See CMSPROG
PURGE
IUCV function 164
parameter list format 165
trace table entry format 194
using 164
SPOOL system service function 275
PURGESYS function 54
purging discontiguous saved segments 54
PUT macro
See CMSPROG
PUTX macro
See CMSPROG

Q

QUERY
action routine 393
command, programmable operator 379
IUCV function 127
trace table entry format 194
using 127
QUERY command, CMS
See CMSPROG
QUERY SRM command
See DIAG
QUIESCE
IUCV function 112, 173
parameter list format 174
trace table entry format 194
using 173
VMCF function 295
QUIT subcommand of DUMPSCAN command
See DIAG
QUIT subcommand of TRAPRED command
See DIAG

R

RDJFCB macro (SVC 64)
See CMSPROG
READ functions of SPOOL system service 270
READ-SFBLOK 271
READ-SPLINK 270
READ-XAB 272
READ macro
See CMSPROG
read system dump spool file with DIAGNOSE code X'34' 33
read system symbol table with DIAGNOSE code X'38' 34
reader interrupts
See CMSPROG
real channel program support 89
real device simulation, VM SNA support 228
real storage
See also CPPROG
examine with DIAGNOSE code X'04' 7

in attached processor environment 8
in multiprocessor environment 8
RECEIVE
IUCV function 112, 149
parameter list format 151
trace table entry format 194
using 149
VMCF function 299
receiving signals from the signal system service 254
records, accounting 36
See also CPPROG
recovery, CMS abend
See CMSPROG
redisplay of input line for SNA terminals 229
REGS subcommand of DUMPSCAN command
See DIAG
REJECT
IUCV function 112, 161
parameter list format 162
trace table entry format 194
using 161
VMCF function 296
release pages with DIAGNOSE code X'10' 13
releasing a logical operator 334
LGLOPR action routine 393
LGLOPR command 334
Remote Spooling Communications Subsystem (RSCS) Networking
programmable operator facility relationship 320, 359
repetitive output, analysis and types
See DIAG
replacing a logical operator 334
LGLOPR action routine 393
LGLOPR command 334
REPLY
IUCV function 112, 154
parameter list format 156
trace table entry format 194
using 154
VMCF function 299
reporting problems
See DIAG
diagnosis
See DIAG
RESERVE
command 246
Resource Access Control Facility (RACF)
See CPPROG
RESTORE macro (SVC 17)
See CMSPROG
restricting command use 354
RESUME
IUCV function 112, 176
parameter list format 177
trace table entry format 194
using 176
VMCF function 295
retrieve a group name with DIAGNOSE code X'A0' 92
RETRIEVE BUFFER function of IUCV 172
trace table entry format 194
using 172
return codes
CMSIUCV 210
DASD Block I/O System Service 246
DIAGNOSE code
X'BC' 101
X'CC' 104
X'C8' 102
X'D0' 105
X'D4' 107
X'0C' 13
X'00' 7
X'08' 10
X'10' 13
X'18' 24
X'28' 31
X'64' 54
X'68' 56, 309
X'7C' 64
X'74' 60
X'78' 61
X'8C' 80
X'80' 70
X'84' 78
X'94' 86, 87
X'98', subcode X'0000' 90
X'98', subcode X'0004' 91
X'98', subcode X'0008' 91
FINDSYS function 55
HNDIUCV 203
IUCV
ACCEPT 140
CONNECT 135
DECLARE BUFFER 130
DESCRIBE 183
PURGE 167
QUIESCE 175
RECEIVE 153
REJECT 163
REPLY 157
RESUME 178
SEND 146
SET CONTROL MASK 193
SET MASK 190
SEVER 170
TEST COMPLETION 187
LOADSYS function 54
PURGESYS function 55
read, write, multi password prompt 10
Signal System Service 253
VMCF 304, 309
return DASD start of LOGREC area 31
REUSE subcommand of DUMPSCAN command
See DIAG
RIOBLOK subcommand of DUMPSCAN command

See DIAG
ROUTE statement 340
See also programmable operator facility
routines, action 318
routing table (RTABLE) 317, 336
See also programmable operator facility
RSCS (Remote Spooling Communications Subsystem) Networking
programmable operator facility relationship 320, 359
RTABLE (routing table) 317, 336
See also programmable operator facility
running the programmable operator from NCCF or NetView 328

S

SAD MACRO
See DIAG
SADGEN ASSEMBLE file
See DIAG
SADGEN TEXT file
See DIAG
SADUMP EXEC
See DIAG
sample CMS IUCV program 425
saved system
See CMSPROG
See CPPROG
saving
a 3800 named system with DIAGNOSE code X'74' 59
CP message repository, DIAGNOSE code X'CC' 103
national language files for CP and CMS 404
national language files for GCS 407
the 370X control program 39
SCBLOCK control block
See CMSPROG
SCBLOCK macro
See CMSPROG
scheduling virtual machines
See CPPROG
SCIF (Single Console Image Facility) 279-280
controlling multiple virtual machines 279
using 279
SCPINFO command 69
screen management, VM SNA support 222
scroll interface, DUMPSCAN
See DIAG
SCROLL subcommand of DUMPSCAN command
See DIAG
SDUMP command
See DIAG
search order
See CMSPROG
second level storage, dumping of 82
second level virtual machine issues SVC 76 36
secondary user 279
security considerations, IUCV 116
SELECT function of SPOOL system service 262
SEND
command with single console image facility 279
IUCV function 112, 142
parameter list format 145
to the DASD block I/O system service 245
to the signal system service 254
to the SPOOL system service 261
trace table entry format 194
using 142
SPOOL system service function 274
VMCF function 297
VMCF protocol 289
SEND/RECV
VMCF function 297
VMCF protocol 290
sending signals to the signal system service 254
SENDX
VMCF function 298
VMCF protocol 291
sequence of functions in CMS IUCV 212
sequence of functions in IUCV 117, 120
SET
action routine 393
command, programmable operator 382
SET command
See CMSPROG
SET CONTROL MASK function of IUCV 115, 191
parameter list format 192
trace table entry format 194
using 191
set language, DIAGNOSE code X'C8' 101
SET MASK function of IUCV 115, 188
parameter list format 189
trace table entry format 194
using 188
SEVER function of IUCV 111, 168
parameter list format 169
trace table entry format 194
using 168
shadow table maintenance with DIAGNOSE code X'6C' 57
shared segments
See CPPROG
shutdown, SNA 232
*SIGNAL 252
Signal System Service 251-256
connecting with 252
establishing communications with 251
leaving 255
receiving signals 254
sending signals 254
simulated OS supervisor calls
See CMSPROG

simulation
See CMSPROG
Single Console Image Facility 279
Single Console Image Facility (SCIF) 279
controlling multiple virtual machines 279
using 279
single processor mode
See CPPROG
single system use of the programmable operator 315
SIO (Start I/O instruction)
initiating full screen mode 48
SIO (type X'06') entry
See DIAG
SMSG command 277
SNA (System Network Architecture)
console communication services 219
VM/SP support 219
accounting 231
CMS mode 221
command handling 228
communication interfaces 223
console mode 221
enabling SNA terminals 225
environments supported 221
establishing connections 225
full screen support mode 222
functions, SNA 224
I/O processing 229, 230
MORE/HOLDING condition 231
NCP and PEP sharing 232
operator considerations 232
real device simulation 228
redisplay of input line for terminals 229
screen management 222
shutdown 232
system structure 220
trace table entries 233
TRQBLOK 230
user termination 232
WEBLOK 223, 229
WEIBLOK 229
VM/SP virtual console support 219
VTAM service machine 220
SNAP macro (SVC 51)
See CMSPROG
source files, national language 402
spanned records, usage
See CMSPROG
Special Message Facility 277-278
buffer length 277
description 277
introduction 277
receiving messages via IUCV 278
receiving messages via VMCF 277
sending messages 277
SMSG command 277
special message flag (VMCPSMSG) 277
turning on or off 278
specify an alternate userid with DIAGNOSE code X'D4' 106
specifying DASD space for CP message repository 405
specifying routing texts 346
to an NCCF or NetView operator 349
SPIE macro (SVC 14)
See CMSPROG
*SPL 259
spool file
See also CPPROG
manipulation 14
spool file external attribute buffer manipulation 98
spool file opening with DIAGNOSE code X'BC' 100
spool file, system dump 33
SPOOL System Service 259-275
definition 259
establishing communications 259
functions 259
CLOSE 265
IUCV CONNECT 260
IUCV SEND 261
MESSAGE 268
NOTIFY 275
PURGE 275
READ-SFBLOK 271
READ-SPLINK 270
READ-XAB 272
SELECT 262
SEND 274
to a logical printer 273
spooling
See CPPROG
STAE macro (SVC 60)
See CMSPROG
stand-alone dump facility
See DIAG
standard format
CMSIUCV macro 205
HNDIUCV macro 200
Start I/O (SIO) instruction
initiating full screen mode 48
starting the programmable operator 324
STAT command
See DIAG
statall local file
See DIAG
status file
See DIAG
STATUS, logical device support facility function 63, 67, 282
STAX macro (SVC 96)
See CMSPROG
STCP command
See DIAG
STIMER macro (SVC 47)
See CMSPROG
STOP
action routine 393

command 385
STOP command, programmable operator 385
stop execution
See DIAG
stopping the programmable operator facility 327
storage contents, altering
See DIAG
storage protection
See CPPROG
storage size
maximum, replacing directory entry 74
virtual machine, replacing directory entry 74
STORE command
See DIAG
store extended-identification code with DIAGNOSE code X'00' 4
store real CP data
See DIAG
store virtual data
See DIAG
STOW macro (SVC 21)
See CMSPROG
STRINIT macro
See CMSPROG
structure of CMS storage
See CMSPROG
structure, SNA system 220
structured data base (SDB)
See DIAG
reporting
See DIAG
SUBCOM function
See CMSPROG
summary of changes 431
summary record file
See DIAG
supervisor calls
See CMSPROG
SVC interrupt (type X'05') entry
See DIAG
SVC interrupts
See CMSPROG
SVC save area (SVCSAVE)
See DIAG
SVC types
See CMSPROG
SVC 199 services
See DIAG
SVC 202
See CMSPROG
SVC 203
See CMSPROG
SVC 76 from a second level virtual machine 36
SVCTRACE command
See DIAG
switching, CMS tape volume
See CMSPROG
symbol table, read 34
SYMP subcommand of DUMPSCAN command
See DIAG
symptom records
See DIAG
symptom summary file
See DIAG
symptom summary file conversion
See DIAG
SYNADAF macro (SVC 68)
See CMSPROG
SYNADRLS macro (SVC 68)
See CMSPROG
SYSCOR macro
See DIAG
SYSOPR macro
See DIAG
SYSRLB system logical units assigned in CMS/DOS
See CMSPROG
SYSSLB system logical units assigned in CMS/DOS
See CMSPROG
system
dump spool file, reading 33
structure, SNA 220
symbol table, reading 34
system abend
See DIAG
system information, collect and analyze
See DIAG
System Network Architecture (SNA)
See also SNA (System Network Architecture)
console communication services 219
VM/SP support 219
accounting 231
CMS mode 221
command handling 228
communication interfaces 223
console mode 221
enabling SNA terminals 225
environments supported 221
establishing connections 225
full screen support mode 222
functions, SNA 224
I/O processing 229, 230
MORE/HOLDING condition 231
NCP and PEP sharing 232
operator considerations 232
real device simulation 228
redisplay of input line for terminals 229
screen management 222
shutdown 232
system structure 220
trace table entries 233
TRQBLOK 230
user termination 232
WEBLOK 223, 229
WEIBLOK 229
VM/SP virtual console support 219
VTAM service machine 220

system security
See CPPROG
system service, CP
DASD block I/O 243
error logging 257
IUCV communication 197
message 239
message all 241
signal 251
SNA virtual console communication 219
spool 259
system spool information, DIAGNOSE code X'D8' 108

T

table, routing 317
TACTIVE subcommand of DUMPSCAN command
See DIAG
tailoring a programmable operator routing table 346
tape volume serial number, 3480 105
tape volume switching in CMS
See CMSPROG
TCLEARQ macro (SVC 94)
See CMSPROG
TERMINAL command
BREAKIN GUESTCTL 49
BRKKEY 49
CONMODE 3270 48
SCRNSAVE OFF 48
SCRNSAVE ON 48
terminal interrupts
See CMSPROG
TERMINATE ALL, logical device support facility function 63, 67, 282
TERMINATE, logical device support facility function 63, 66, 282
terminating the programmable operator facility 327
termination, abnormal
See abnormal termination (abend)
terminology of IUCV macro parameters 124
TEST COMPLETION
IUCV function 112
TEST COMPLETION function of IUCV 184
parameter list format 185
trace table entry format 194
using 184
TEST MESSAGE function of IUCV, using 179
trace table entry format 194
TEVC (trace entry verification code)
See DIAG
TEXTSYM statement 338
See also programmable operator facility
TGET/TPUT macro (SVC 93)
See CMSPROG
third level storage, not dumped 82
TIME macro (SVC 11)
See CMSPROG
timer, pseudo 12
timers
See CPPROG
TLOADL subcommand of DUMPSCAN command
See DIAG
TOD-clock accounting interface 57
TOFB action routine 393
tokenized PLIST
See CMSPROG
TOP subcommand of TRAPRED command
See DIAG
TOVM action routine 393
TRACCURR (current CP internal trace table entry)
See DIAG
TRACE command
See CPPROG
See DIAG
trace entry verification code (TEVC)
See DIAG
trace execution
See DIAG
trace real machine events
See DIAG
TRACE subcommand of DUMPSCAN command
See DIAG
trace table
CP
See CPPROG
entry
formats, IUCV 194
formats, SNA CCS 233
IUCV field definitions 195
TRACEND (end of CP internal trace table)
See DIAG
tracing
See also DIAG
IUCV functions 193
SNA Console Communication Services 233
tracing capabilities in EXECs
See DIAG
tracing storage alteration using PER
See DIAG
TRACSTRT (start of CP internal trace table)
See DIAG
transient program area
See CMSPROG
transient routines
See CMSPROG
TRAPRED command format
See DIAG
TRAPRED facility
See DIAG
TRAPRED subcommands
See DIAG
TRKBAL macro (SVC 25)
See CMSPROG

TRQBLOK, VM SNA support 230
TSAB subcommand of DUMPSCAN command
  See DIAG
TSAF console log sample
  See DIAG
TSAF debugging
  See DIAG
TSAF dumps
  See DIAG
TSAF internal trace table
  See DIAG
TSAF QUERY command
  See DIAG
TSAF SET ETRACE command
  See DIAG
TSAFIPCS MAP
  See DIAG
TTIMER macro (SVC 46)
  See CMSPROG
TVSPARMS macro
  See CMSPROG
type (device) 419
TYPE subcommand of TRAPRED command
  See DIAG
TYPEBACK subcommand of TRAPRED command
  See DIAG
typenum subcommand of TRAPRED command
  See DIAG
types, classes, models, and features of devices (DIAGNOSE code X'24') 27, 419

U

UCS (Universal Character Set)
  See CPPROG
UCSB (Universal Character Set Buffer)
  See CPPROG
UNAUTHORIZE function of VMCF 294
unexpected results procedures
  See DIAG
UP subcommand of TRAPRED command
  See DIAG
updating
  directory
    entries for national languages 407
    in-place with DIAGNOSE code X'84' 72
    with DIAGNOSE code X'3C' 34
  files for an existing national language 409
updating problem report
  See DIAG
user area (USERSECT)
  See CMSPROG
user directory
  entries for national languages, updating 407
  reading 34
  updating with DIAGNOSE code X'3C' 34
user doubleword function of VMCF 309
user free storage
  See CMSPROG
user options, replacing directory entry 76
user privilege classes
  See CPPROG
user program area
  See CMSPROG
user save area
  See CMSPROG
user termination, SNA 232
user-controlled device interrupts
  See CMSPROG
userid, alternate 106
USERMAP subcommand of DUMPSCAN command
  See DIAG
USERSECT (user area)
  See CMSPROG
using
  CMS IUCV to communicate between two virtual machines 212
  DASD block I/O system service from CMS 246
  IUCV data in a buffer 117
  IUCV data in a parameter list 120
  programmable operator facility
    in a distributed VM system 316
    in a mixed environment 316
    in a single system 315
  single console image facility (SCIF) 279
  VMCF 284

V

verifying existence of saved systems 55
VIOBLOK subcommand of DUMPSCAN command
  See DIAG
virtual
  console function, DIAGNOSE instruction 9
  IPL by device, clean-up after 35
virtual machine
  See CPPROG
virtual machine assist feature
  See CPPROG
Virtual Machine Communication Facility (VMCF) 283
  control functions 293
  data transfer functions 296
  DIAGNOSE code X'68' 55, 283, 300
    data transfer error codes 312
    return codes 309
  external interrupt 305
  functions 293
    AUTHORIZE 293
    CANCEL 295
    IDENTIFY 296

PRIORITY option 294, 298
QUIESCE 295
RECEIVE 299
REJECT 296
REPLY 299
RESUME 295
SEND 297
SEND/RECV 297
SENDX 298
special message facility 294
SPECIFIC option 294
UNAUTHORIZE 294
introduction 283
invoking functions 300
protocol 288
IDENTIFY 292
SEND 289
SEND/RECV 290
SENDX 291
receiving messages from the special message facility 277
return codes 309
special message facility 284
table of functions 284
user doubleword 309
using 284
applications 285
general considerations 288
performance considerations 287
security 286
virtual machine data, recording
See DIAG
virtual machine debugging
See DIAG
virtual machine storage size
maximum, replacing directory entry 74
replacing directory entry 74
Virtual Machine/System Product (VM/SP)
See VM/SP (Virtual Machine/System Product)
virtual machines
determine storage size with DIAGNOSE code X'60' 52
DIAGNOSE instruction use 3
group 251
multiple, controlling from a single console 279
programmable operator 321
to virtual machine communication, CMS IUCV 212
to virtual machine communication, IUCV example 117
virtual printer external attribute buffer manipulation 94
Virtual Storage Access Method (VSAM)
See CMSPROG
virtual storage preservation
See CPPROG
virtual storage, altering
See DIAG
VM/SP (Virtual Machine/System Product)
device types in 27
DIAGNOSE instruction 3
directory
authorization for IUCV 116
entries for national languages, updating 407
entries in IUCV 116
replacing entries 72
update in-place 72
updating with DIAGNOSE code X'3C' 34
VM/VCNA, VM/SP SNA support 219
VM/VTAM, VM/SP SNA support 219
VMBLOK subcommand of DUMPSCAN command
See DIAG
VMCF (Virtual Machine Communication Facility) 283, 312
control functions 293
data transfer functions 296
DIAGNOSE code X'68' 55, 283, 300
data transfer error codes 312
return codes 309
external interrupt 305
functions 293
AUTHORIZE 293
CANCEL 295
IDENTIFY 296
PRIORITY option 294, 298
QUIESCE 295
RECEIVE 299
REJECT 296
REPLY 299
RESUME 295
SEND 297
SEND/RECV 297
SENDX 298
special message facility 294
SPECIFIC option 294
UNAUTHORIZE 294
introduction 283
invoking functions 300
protocol 288
IDENTIFY 292
SEND 289
SEND/RECV 290
SENDX 291
receiving messages from the special message facility 277
return codes 309
special message facility 284
table of functions 284
user doubleword 309
using 284
applications 285
general considerations 288
performance considerations 287
security 286
VMCPARM parameter list 301
VMDUMP command
See DIAG
VMDUMP function 81
VMLOADL subcommand of DUMPSCAN command

See DIAG
Volume Table of Contents (VTOC)
See CMSPROG
VSAM (Virtual Storage Access Method)
See CMSPROG
VSAM data sets
See CMSPROG
VSAM dumping information
See DIAG
VSCS printing formatted control blocks
See DIAG
VSCS, VM/SP SNA support 219
VSE
See CMSPROG
VSE/VSAM
See CMSPROG
VTAM I/O reduction 231
VTAM printing formatted control blocks
See DIAG
VTAM service machine, VM SNA support 220

W

wait bit, modifying
See DIAG
WAIT macro (SVC 1)
See CMSPROG
wait state procedures
See DIAG
WEBLOK, VM SNA support 223, 229
WEIBLOK, VM SNA support 229
WRITE macro
See CMSPROG
WTO/WTOR macro (SVC 35)
See CMSPROG

X

XAB (External Attribute Buffer) 94
definition 96
format 96
manipulation of spool file 98
manipulation of virtual printer 94
READ-XAB function of SPOOL system service 272
XCTL macro (SVC 7)
See CMSPROG
XDAP macro (SVC 0)
See CMSPROG
XEDIT (System Product Editor)
See CMSPROG

Z

ZAP command
See DIAG
ZAPTEXT command
See DIAG

Numerics

3081 processor, MSSFCALL - DIAGNOSE code X'80' 69
3203
See CPPROG
3270
logical, creating via logical device support facility 281
virtual console interface
attribute bytes, how to supply 42
full screen interactions 45
full screen interactions (3270 SIO) 48
full screen mode 43
selector-pen limitations 42
37XX Control Program
See also CPPROG
DIAGNOSE code X'50' 39
370X control program, saving 39
DIAGNOSE code X'50' 39
370X dump processing
See DIAG
3800 printer
See also CPPROG
load CCWs in spool file 19

International Business
Machines Corporation
P.O. Box 6
Endicott, New York 13760

File No. S370/4300-36
Printed in U.S.A.

SC24-5288-0

VM/SP System Facilities
for Programming
Order No. SC24-5288-0

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

_____ _____ _____ **Help Information line ___ of ___**

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

_____

_____

_____

_____

IBM Branch Office serving you: _____

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

SC24-5288-0

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

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, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business
Machines Corporation
P.O. Box 6
Endicott, New York 13760

File No. S370/4300-36
Printed in U.S.A.

SC24-5288-0

SC24-5288-00

IBM®