INTERCOMM

OPERATING REFERENCE MANUAL

ISOGON CORPORATION
330 Seventh Avenue, New York, New York 10001

LICENSE: INTERCOMM TELEPROCESSING MONITOR

Publishing History

Publication	Date	Remarks
First Edition	February 1974	This manual corresponds to Intercomm Release 6.0. It incorporates and supercedes documentation formerly in the Intercomm Users Guide, now obsolete.
Second Edition	March 1983	General updates and additions corresponding to Intercomm Release 9.0.

PREFACE

Intercomm is a state-of-the-art teleprocessing monitor system executing on the IBM System 360/370 family of computers and operating under the control of IBM Operating Systems (MFT, MVT, VS1, MVS). Intercomm monitors the transmission of messages to and from terminals, concurrent message processing, centralized access to I/O files, and the routine utility operations of editing input messages and formatting output messages, as required.

Installing and maintaining an on-line system is a complex task with many variables ranging from coordination of equipment delivery and associated environmental planning to scheduling the implementation of application programs which service users at remote locations. One phase of this installation is implementing Intercomm, the on-line system monitor which schedules and controls the operation of the communications network, as well as the application programs that process the traffic input from, and produce the output to, the network.

This document provides guidelines for the installation, maintenance and tuning of Intercomm, including an orderly breakdown of responsibility for system definition, testing and production operation. It serves as a reference manual for systems personnel responsible for the operation of the on-line system.

The following Intercomm publications are prerequisite and/or relevant to this document:

- Concepts and Facilities
- Installation Guide
- Basic System Macros
- Messages and Codes
- System Control Commands

A Users Review Form is included at the back of this manual. We welcome recommendations, suggestions and reactions to this or any Intercomm publication.

INTERCOMM PUBLICATIONS

GENERAL INFORMATION MANUALS

- Concepts and Facilities
- Planning Guide

APPLICATION PROGRAMMERS MANUALS

- Assembler Language Programmers Guide
- COBOL Programmers Guide
- PL/1 Programmers Guide

SYSTEM PROGRAMMERS MANUALS

- Basic System Macros
- BTAM Terminal Support Guide
- Installation Guide
- Messages and Codes
- Operating Reference Manual
- System Control Commands

CUSTOMER INFORMATION MANUALS

- Customer Education Course Catalog
- Technical Information Bulletins
- User Contributed Program Description

FEATURE IMPLEMENTATION MANUALS

- Amigos Users Guide
- Autogen Facility
- ASMF Users Guide
- DBMS Users Guide
- Data Entry Installation Guide
- Data Entry Terminal Operators Guide
- Dynamic Data Queuing Facility
- Dynamic File Allocation
- Extended Security System
- File Recovery Users Guide
- Generalized Front End Facility
- Message Mapping Utilities
- Model System Generator
- Multiregion Support Facility
- Page Facility
- Remote Job Entry (OS)
- Store/Fetch Facility
- SNA Terminal Support Guide
- TCAM Support Users Guide
- Utilities Users Guide

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1

THE INTERCOMM ENVIRONMENT

1.1 INTRODUCTION

The Intercomm on-line teleprocessing monitor may be utilized on the System 360/370 (and compatible) family of computers (including 303x, 43xx, 308x, etc.) and executes under the control of the Operating System (MFT or MVT) or System/370 Virtual Storage system (VS1 or MVS). With any one of the operating systems, any number of concurrent independent jobs may be submitted and executed while the Intercomm system is operating.

Intercomm operates as a job in a multiprogramming, multitasking, time-dependent environment. Any number of applications may be concurrently executed under the control of the Intercomm monitor; any number of terminals, types of input, application programs, and file access methods may be used.

Application programs executing under Intercomm may be written in any of the System 360/370 compiler languages: Assembler Language, COBOL, PL/1, or FORTRAN. The user can also convert from a batch processing to an on-line environment without having to totally rewrite application programs.

Intercomm is a table-driven system; that is, operating specifications are described to the system in the form of tables. Thus, Intercomm components are individual routines coded in generalized form where applicable, utilizing table entries for execution requirements. The application programmer is generally not concerned with these table entries, but is responsible only for the problem solving logic. All message routing, time-sharing, message mix, and communication functions within Intercomm are, in general, transparent to the application programmer.

The prerequisite publication to this document, Intercomm Concepts and Facilities, describes the general system logic of an Intercomm environment. In this section a brief review is provided of the major system components, region organization, modes of execution, and user-specified tables.

An Intercomm system consists of user-coded application subsystems (message processing programs) and the following Intercomm components:

- Front End Teleprocessing Interface
  System programs responsible for all operation of the telecommunications network.

- Subsystem Controller
  System programs responsible for all scheduling, loading and activating of message processing subsystems.

- Queue Management
  System programs controlling queuing and retrieval of messages waiting for processing or transmission.

- File Handler
  System programs exercising centralized control over all Operating System data management functions.

- Dispatcher
  The multithreading control routine that schedules use of the CPU among concurrently executing tasks.

- Resource Management
  Optional system programs provided to ensure efficient main storage management and control over system resources in the event of program failure.

- Utility Programs
  System programs provided to simplify design and implementation of application programs and message processing logic.

- System Control Routines
  Optional system programs providing logging (journaling), restart/recovery, system control transactions, a comprehensive dynamically controlled security environment, debugging and tuning aids, program error interception, system reliability, etc.

1.2 FRONT END

This component of Intercomm controls all teleprocessing functions of the system. An on-line installation may optionally utilize one or more of the following Teleprocessing Interface components:

- The Intercomm BTAM Front End, a conditionally assembled, table-driven series of programs providing efficient interface to a wide variety of terminals through IBM's Basic Telecommunicatons Access Method and Graphics Access Method.

- The Intercomm TCAM Interface to a Message Control Program operating in a separate region where all line control functions are performed according to macro-generated specifications for IBM's Telecommunications Access Method. The Extended TCAM support provides interface to TCAM process and destination queues via the BTAM Front End.

- The Intercomm VTAM Front End, communicating with a VTAM control region and interfacing with both SNA and non-SNA devices.

- A user-supplied interface to nonsupported devices implemented by the Generalized Front End Interface of the BTAM Front End.

1.3 SUBSYSTEM CONTROLLER

The Subsystem Controller interacts with the Teleprocessing Interface via the queue management routines to control all message processing within the on-line system. It directs incoming messages to the proper application programs, schedules and loads nonresident subsystems as required.

The Subsystem Controller optimizes dynamic loading of subsystems and/or program swapping (overlay management) to increase throughput, and diagnoses application program errors to provide an uninterrupted on-line operation.

Subsystem Controller processing is governed by user-varied tables specifying the message routing structure and variable processing factors which can be adjusted to maximize throughput.

1.4 QUEUE MANAGEMENT ROUTINES

Message queues are the prime interface between the Front End (TP Device Control) and Back End (Message Processing Control) components of Intercomm. Input messages are queued for processing by subsystem; output messages are queued for transmission by logical unit, terminal, line, or user-specified discipline. Messages may be queued in main storage and/or on disk at the user's option. Disk queues are wraparound, reuseable BDAM data sets. A queue is a logical entity; one physical data set may be shared for several queues. The queue management routines are service routines utilized by both system programs and application subsystems.

1.5 FILE HANDLER

By processing all on-line files through a single module, Intercomm eliminates duplication of I/O routines, control blocks and buffers in application programs. It also eliminates the highly wasteful opening and closing of data sets for each message processed--files are opened only once per day (or shift). In concert with the Dispatcher, tasks that access files are maximally overlapped with other tasks (processing threads) requiring CPU time.

All data set organizations (sequential, direct, indexed) and processing techniques (by logical record, by physical block, keyed access, random access) are available to programs written in any language. Comprehensive diagnostics for on-line security and I/O error analysis are provided, as well as write-protection of master files.

Exclusive control of individual records or blocks within files, recommended where simultaneous updating could occur, is also provided as one of the File Handler's functions, and, via an exclusive control time-out, those records held beyond a specified time limit may be released from exclusive control.

1.6 DISPATCHER

The Intercomm Front End Teleprocessing Interface, the Subsystem Controller and the File Handler create multiple independent threads (parallel program paths for parallel message processing) using the Dispatcher, which allocates and overlaps CPU time among any number of concurrent work requests, and establishes any number of concurrent real-time clocks. This is achieved within a single Operating System task, thus obviating the need for a multitasking operating system and formal dynamic program linkage through the Supervisor. The Dispatcher also assists in overlay management and dynamic program management under direction of the Subsystem Controller.

1.7 RESOURCE MANAGEMENT

The Resource Management facilities of Intercomm provide efficient storage management techniques, unless specifically bypassed by the user. Additionally, a storage cushion feature is available to serve as a protection against a temporary shortage of main storage. The cushion (of user-selected size) is an area gotten from subpool zero at startup and held, but not used, until a request for dynamic storage cannot be satisfied. At that point, the cushion is returned to subpool zero and used to satisfy storage requests for messages currently in progress. No new message processing is started until reduced storage demands, as messages are completed and transmitted, allow the cushion to be reacquired by the monitor. The impact of a noncritical shortage of dynamic storage is therefore avoided. Resource Management options are described below and may be used singly or in combination with each other.

The resource auditing and purging option provides a chain of control blocks built for every active program thread. These blocks correspond on a one-to-one basis with resources acquired by the program. Resources may be areas of storage, files, or any facility subject to ownership. Purging is accomplished by freeing unreleased resources, represented by the control block chain, for a program thread when the thread normally or abnormally completes. A thread resource dump (TDUMP) is provided as an audit utility to print out control block chains, showing which thread is in control of what unreleased resources, through which module the resources were obtained and in what order acquisition occurred.

As an adjunct to audit/purge or as an independent option, the creation of main storage pools, which section a contiguous area of storage into specified block sizes, is offered with Resource Management. Storage pools are generated by a macro which defines the size and number of pools, and the number of blocks within each pool to be generated to fit user requirements. The pool option not only manages storage allocation to eliminate fragmentation problems but furthermore, through indexed access to the pools, provides a significant increase in the speed with which storage may be obtained and freed, owing to the elimination of GETMAIN and FREEMAIN SVCs.

The third option consists of two distinct sets of core-use statistics: global and detail. Inclusion of either set may be made without reference to the other. The global statistics present such information as the number of requests for storage and requests to free storage, the average storage request length, and the number of requests filled from the pools. Detail statistics consist of the breakdown of storage requests into size ranges. The primary purpose of the detail statistics report is to provide sufficient statistics from actual system usage so that an effective selection of the number and sizes of pool blocks may be made at an installation.

1.8 UTILITY PROGRAMS

In addition to the File Handler, a number of on-line utility functions are provided to ease programming of application subsystems and to centralize control of such functions. The interface is via standard call logic in the subsystem. These facilities include:

- Message Mapping Utilities--device-independent message editing, formatting, and output routing
- Store/Fetch--temporary data string storage and retrieval
- Dynamic Data Queuing--transient queues of data strings, file records, or messages
- Page Browsing--collections of output messages for paging access from a CRT device

- Dynamic File Allocation--allocate and/or access data sets not defined explicitly via JCL.

Additionally, the EDIT, OUTPUT, DISPLAY and CHANGE Utilities provide alternate means of message and file record processing. EDIT strips the incoming message of TP control characters and provides for complete field-by-field editing of the input message. It also performs keyword parameter analysis. OUTPUT supplies device-independent output capabilities to application programmers. DISPLAY allows a remote operator to display an individual file record (for BDAM or ISAM files) in a fixed character format on his terminal. CHANGE allows the operator to modify selected fields in a file record obtained by DISPLAY.

1.9 REGION ORGANIZATION

At execution time, the Intercomm region (or partition) consists of system programs, tables, and message-processing subsystems.

- Resident Intercomm routines
  These routines are required constantly for Intercomm functions and must be resident. Residing in this required area is the Intercomm nucleus, that is, such routines as the Subsystem Controller and Dispatcher.

- Resident tables
  Certain tables are necessarily resident in that they specify actual control functions of Intercomm. For example, the System Parameter Area (SPA) describes systemwide characteristics. Resident tables share the Intercomm nucleus with resident routines.

- Resident subsystems
  Frequently used subsystems and subroutines should remain in main storage. Whether a program is resident is a factor in good planning and can provide for both maximizing system throughput and minimizing individual transaction response time.

- Nonresident subsystems; dynamically loadable
  Nonresident subsystems and subroutines can be defined as dynamically loadable into main storage. These programs are loaded on an as-required basis. Reuseable subsystems remain resident until prescribed message processing limits are reached or message traffic ceases, and nonreuseable subsystems are reloaded for every message processed.

- Nonresident subsystems; planned overlay structure
  The Intercomm region may contain one or more overlay regions: Overlay A,B,C,D. The first region therein, Overlay A, has special characteristics in that groups of subsystems are loaded to process messages concurrently. Overlays B,C,D are utilized for single-thread, noncritical message-processing subsystems. The sequence of overlay load is based on message traffic and scheduling criteria.

- Nonresident service routines
  Service routines that may be nonresident are those not called frequently. When required, they are loaded into the transient overlay area of the Intercomm region. If an overlay structure is not defined, all Intercomm service routines must be resident in the Intercomm region, or in the Intercomm portion of the Link Pack Area.

- Nonresident table entries
  Infrequently used table specifications, for example, message formats for the Message Mapping and Output Utilities, can be contained on disk and loaded when needed.

- Dynamic Subpool Area
  This is the areas of main storage that are obtained dynamically (as needed) for loading Intercomm or user routines or tables. The subpool area is dynamic in that the composition varies and areas are assigned, or released and made available for reuse, as soon as the monitor determines that the area is no longer needed.

1.9.1 Dynamic Program Loading

Nonresident subsystems and subroutines are loaded into the dynamic subpool area during ongoing execution of the Intercomm partition/region via the dynamic load facility which interfaces with an asynchronous loader task. Programs are expeditiously loaded on demand, according to arrival sequence of incoming message traffic. A loaded subsystem remains resident until a maximum of messages is processed (limit specified by the Subsystem Control Table), or until message traffic ceases.

Once loaded, any subsystem defined as reuseable or reentrant is left resident in the dynamic area and rescheduled as needed, as long as the storage it occupies is not required for a subsequent subsystem load during an unscheduled interval. A nonreuseable subsystem will be reloaded for every message. Within this framework any reuseable/reentrant subsystem processes more than one message, if queued.

A BLDL, or load list, area may optionally be requested for each dynamically loaded program. Although load list specification increases the size of the resident Intercomm tables, it provides for faster loading and is recommended for frequently used programs.

The predefined maximum amount of storage useable for concurrently loadable subsystems can be varied while Intercomm is operational via a system control command. The load module used for a dynamically loaded program may be reloaded via a system control command to allow replacement of that program during Intercomm execution. Dynamic Linkedit, an optional feature, resolves external references between loaded and resident programs at startup and when a replacement program copy is loaded by command.

1.9.2 Overlay Program Loading

Loading of subsystems may be controlled by the Intercomm Overlay Management scheduling facility, in which case subsystems are linkedited as overlay region segments and loaded according to a preplanned structure and sequence. As with dynamically loadable subsystems, the sequence of subsystem load is dictated by message traffic.

1.9.3 Asynchronous Overlay Loader

The Intercomm Overlay Loader is an asynchronous multiprogramming interface between Intercomm and the OS Overlay Supervisor that allows Intercomm to coordinate the loading of programs asynchronously with the execution of other Intercomm threads. This prevents Intercomm from being placed in a wait state by the Overlay Supervisor, while still allowing full use of overlay facilities.

When multiple messages for subsystems in more than one overlay area require concurrent loading of multiple regions, they are automatically queued by being dispatched on one of the communications Event Control Blocks (ECB) between the two tasks. This technique permits resident subsystems and those active (already loaded) overlay areas to continue processing.

The Intercomm Overlay Loader allows greater versatility than an independent loader--due to the power of the OS Overlay Supervisor, and at the same time provides full processing overlap.

The Operating System (MVT,MFT) must have the ATTACH and IDENTIFY options (standard features in VS1 and MVS) to utilize this Intercomm facility.

1.10 MODES OF EXECUTION

Mode of execution in the Intercomm environment pertains to operation with or without on-line terminals and to operation with or without consideration for previous execution ("cold" vs. "warm" start). Further, reference may be made in this document to operation in the production environment or testing environment. The Intercomm mode of execution is determined by parameters specified via JCL to indicate whether or not terminals are operational or whether or not restart functions are to be performed. The actual application subsystems executed to process messages are unaffected by the production or testing status of the system.

Intercomm operates in Test Mode in three ways: via message processing in a batch mode; or via time-oriented simulation of terminals whereby disk data sets of input messages exist for each terminal simulated; or with a combination of live and simulated terminals. These three types of test facilities are provided without any changes to the user application program(s) being tested.

Batch Test Mode allows for input of transaction data at system startup time through SYSIN. Those transactions are then queued and passed into the system at the rate of an extremely high volume environment, with multithreading taking place in the application programs almost immediately, just as if the messages had come from on-line terminals. The Batch Mode testing facility allows for pseudo high volume testing, but in no way represents a projected processing capability based on random message arrival rates from a simulated network.

A second type of testing facility is provided with the BTAM "terminal simulator". Separate message queues are established on direct access sequential data sets for each simulated terminal. Intercomm retrieves messages from "terminal queues" based on a unique time value for each pseudo terminal. The terminal simulator allows the user to simulate a "live" Intercomm environment by defining a network of these pseudo terminals. This network could represent the eventual network a user expects to install, or already has in use. Note that although definition of a BTAM terminal network is required for the simulator, input and output processing of messages is essentially the same no matter which type of Front End (BTAM, TCAM or VTAM) is used for the live Intercomm system. In addition, the user may request a printed display of how 3270 terminal messages (formatted and unformatted) will appear in live mode.

The third type of testing facility allows the user to operate with all the terminals of his present on-line system and to simulate those terminals which are not presently operating or which represent the eventual projected network. This facility allows the testing of application programs with a combination of both live terminals and pseudo terminals. This combined network can then be operated under control of Intercomm. This feature merely expands the capabilities of the Intercomm Front End.

Additionally, Intercomm provides a Multiregion mode of execution, wherein there is one "control" region (partition) containing the Front End teleprocessing interface and system control routines, and one or more "satellite" regions containing only Back End facilities and user application processing programs. Optionally, high-volume application subsystems may execute in the control region. One of the satellite regions may be used only for live testing of application programs. Thus, the separation of application subsystems into several regions provides file or data base access centralization, additional security control, and system integrity and storage protection, without impacting the terminal user or response time.

1.11 INTERCOMM TABLES

Intercomm is a generalized on-line system and, as such, requires operating specifications for each particular installation. This information is provided to the system in the form of tables which are coded using Intercomm macros. An application programmer is usually not involved in defining the Intercomm tables, except for the application program requirements. Tables are coded for each of the following Intercomm functions, by which the user specifies his unique requirements:

- Line Control
  - -- network configuration
  - -- transaction validation
  - -- terminal queues
- Message Processing Control
  - -- application subsystem specifications
  - -- subsystem queues
- System Control
  - -- storage pool specifications
  - -- logging requirements
  - -- checkpoint/restart/recovery specifications
  - -- debugging options
  - -- statistics and tuning facilities
- Application Program Services and Utilities

Thus, Intercomm is a table-driven system. Line control information, that is, the number of logical units or terminals and their exact hardware characteristics, is provided to the system, facilitating such operations as LOGON control, polling and addressing, process and destination queuing, and rerouting of messages.

Specifications for message processing control functions are tabular: the type of applications the user has, their scheduling, whether an application program is capable of processing several messages concurrently and, if so, the maximum number of messages to be handled concurrently.

System control functions are table-driven; tables provide specifications for which logging entries are required, the frequency of checkpoint and information to be checkpointed, the particular files to be updated, and specifications relating to restart requirements and file integrity. In addition, the application program services, such as Message Mapping, operate according to user-specified table entries and definitions.

Major functions in Intercomm are controlled by the following tables:

- System Global Tables (SETENV, SETGLOBE)
  Global tables used to control conditional assembly of many Intercomm system routines, thus tailoring code requirements to the individual installation.

- Front End Verb Table (BTVRBTB)
  A table listing all valid four-character transaction identifiers (verbs) and relating them to the subsystem used for message processing. There is one entry per transaction or message type.

- Front End Network Configuration Tables
  Tables describing the terminal network hardware operating characteristics, queuing specifications, logging/restart requirements, and relating individual devices to five-character station identifications.

- Station Table and Device Table
  Tables describing terminal device-dependent characteristics to the Back End utilities.

- System Parameter Area (SPA)
  A table describing systemwide operating characteristics. This table may be extended to include a user area with installation-defined parameters or tables, accessible to all subsystems.

- Subsystem Control Table (SCT)
  A table listing the characteristics (reentrancy, language, entry point, etc.), queue specifications (main storage and/or disk queues), scheduling (resident or loadable, concurrent message processing limits, etc.) and logging/restart specifications for application subsystems. There is one entry per subsystem.

- Data Set Control Table (DSCT)
  A table automatically generated by the File Handler describing on-line data sets. Information in the table is derived from JCL and File Attribute Record (FAR) statements at execution time.

- Intercomm Storage Pools
  A table of Intercomm-managed storage resource pool blocks, in ascending order by block size. The pools may be resident in the Intercomm linkedit, or dynamically loaded at system startup.

- Message Mapping Definitions
  Sets of external and symbolic (Dsect) maps, along with tables of logical terminal definitions, referenced by application subsystems when invoking the Message Mapping Utilities to edit and format messages and data strings. The definitions are made via MMU macros and stored in prescribed files.

- Edit Control Table (ECT)
  A table describing input message editing specifications for transactions edited by the Edit Utility. There is one entry per transaction. Entries are optionally disk-resident.

- Output Format Table (OFT)
  A table describing output message formatting specifications for messages formatted by the Output Utility. There is one entry per output format. Entries are optionally disk-resident.

Thus, the Intercomm system components are individual routines, coded in a generalized form, where applicable. Each system component receives detailed specification for its program functions via table entries defined via global SET symbols, coding of Intercomm system macros, or DC or parameter statements. Table entries may describe a hardware configuration (for example, the communications network) or software specification (for example, EDIT control functions). By adjusting variable table entries, the user effectively tailors Intercomm routines to his installation without modifying any program logic. Appendix A summarizes all table entries.

This document provides processing features and table entries for many of the system components. Others are described in manuals defining installation for the Front End, System Control Commands, and various Intercomm system and application program facilities.

Chapter 2

THE INTERCOMM OPERATIONAL SYSTEM

2.1 INSTALLATION OVERVIEW

This chapter describes the major requirements for successful installation, standardization and maintenance of the Intercomm teleprocessing system, as follows:

- Intercomm Libraries and Naming Conventions
- Intercomm JCL Procedures
- System Installation and Maintenance Responsibilities
- System Standards
- System Control Functions and Tables

The installation of a basic Intercomm system consists of allocation and cataloging of standard Intercomm libraries, loading the Intercomm release tape to disk via standard OS/VS utilities, copying selected Intercomm JCL procedures to an installation's procedure library, customizing system global tables, and then executing various preparatory steps prior to performing a linkedit and execution of the system. This first installation phase ensures the proper functioning of the system with respect to message processing control functions. Thus, once installation is complete, testing of new application subsystems may begin immediately, independent of the hardware delivery schedule or utilization schedule for existing terminals.

Front End installation consists of table specifications and assembly of the appropriate line and terminal control programs to satisfy the specific requirements of a particular hardware configuration and the teleprocessing access method(s) used.

Instructions for installing the system accompany the release tape, as the system generation procedures may vary from time to time with changes in the system programs, quantity of data to be distributed, and customer equipment to be used (see Installation Guide).

2.2 LIBRARIES

At installation time, the Intercomm system is copied from tape to disk into libraries allocated and cataloged for this specific use.

A library is an Operating System partitioned data set (PDS) consisting of a directory and individual members. Each library is identified by a 4- to 8-character name. A source library is named SYMxxxxx where xxxxx is 1 to 5 characters to complete a unique name. An object library is named OBJxxxxx. A load library is named MODxxxxx.

A systemwide high-level qualifier for the library data sets may be defined at installation time. Intercomm JCL procedures provide for override of the system default (INT) via a P parameter.

The Intercomm system is released on three libraries:

- SYMREL--system macros, COPY members and Dsects, source programs, tables and Job Control Procedures.
- MODREL--system load modules
- SYMUCL--Intercomm User Group contributed programs (see User Contributed Program Description).

These libraries are not to be used for user programs or user modifications to Intercomm modules, as new Intercomm releases are effected by complete replacement of these libraries.

The following libraries must be created at installation time by the user:

LIB-- to hold user-modified versions of Intercomm global tables defined via SET statements:

- SYMLIB--updated system source members
- MODLIB--load modules

NOTE: these libraries are used by the ASMF Facility to hold Intercomm members updated by SMs (periodic system modifications); therefore, they should not contain user-modified Intercomm modules.

MDF-- to hold map group definitions for the Intercomm Message Mapping Utilities:

- SYMMDF--source map definitions
- MODMDF--load module versions of maps

USR-- to contain linkedit control decks, installation JCL, user programs, user-modified versions of, or additions to, Intercomm system tables, or user modifications to Intercomm modules:

- SYMUSR--modified source modules
- MODUSR--load modules

NOTE: SYMUSR is intended as the common link across Intercomm system releases in that it should contain user versions of system tables (or COPY members to be inserted in system tables; see Section 2.7), change decks for user modification of Intercomm system modules (in addition to the changed modules), Intercomm linkedit modifications (to order Csects under VS1/MVS, and to add user modules), etc. All changes to Intercomm system modules and tables must be reexamined for applicability and sequence numbering whenever SMs are applied or a new release is installed.

REF-- a dummy data set (one track) to set the largest block size for a SYSLIB concatenation stream (see Section 2.2.1):

- SYMREF--for block size determination

This is the minimal configuration of the Intercomm libraries.

If desired, all user programs may be placed into the common USR libraries, or "private" libraries may be created for individual programmers or groups:

- SYMxxx--private source programs
- MODxxx--private load modules

For testing purposes, a set of "scratch" libraries may be created, to be scratched and recreated periodically to eliminate unneeded modules and recover space used during updating:

- SYMSCR--Test source programs
- MODSCR--Test load modules

NOTE: Several Job Control Procedures producing executable load modules specify data set MODSCR (see LKEDE, LKEDT).

The Intercomm JCL procedures are so arranged that, whenever a search must be made in a library for a member (such as a macro name, source code to be copied or updated, or modules to be included in a linkedit), a concatenation is used to cause a progressive search to be made for the member in

- The specified private library
- The system modification USR library
- The system update LIB library
- The system release REL library
- Operating System libraries, such as MACLIB, COBLIB, TELCMLIB, etc. (where appropriate)

The search for a member ends with the first library (in the above sequence) containing the member name in its directory, even if another library also contains the named member. Thus, the user of a private library can modify any system component for his own use without affecting the user of any different private library. An installation may choose to modify or add a component to the system USR library, and it will automatically become available to all users. Components modified by SMs will be taken from the system update library, while those not modified/updated by the user will be taken from the library supplied by Intercomm, and components of the Operating System will be taken from the appropriate operating system libraries.

NOTE: If executing under MVS, it may be necessary to modify Intercomm procedures which execute an assembly so that SYS1.AMODGEN is concatenated after SYS1.MACLIB in the ASM step, in order to make system macros and Dsects available for assembly of Intercomm modules.

2.2.1 Source Library Concatenation Sequence

Due to the existence of macros on SYS1.MACLIB that have the same name as Intercomm macros, the Intercomm SYMxxx libraries must be placed before SYS1.MACLIB. When the block size of SYS1.MACLIB is larger than the Intercomm SYMxxx libraries, placing it after the SYMxxxs can cause I/O errors in reading macros, COPY code, etc. There are three ways around the problem:

1. Reblock SYS1.MACLIB to Intercomm Source Libraries block size.

2. Reblock Intercomm source libraries to SYS1.MACLIB block size.

3. Tell the Assembler what the largest block size on SYSLIB is.

Method 1 can propagate the problem to other assemblies. Method 2 is workable but still requires a reblock, and all libraries must have the same block size. Method 3 is the one that is provided by Intercomm installation for all Intercomm JCL procedures using the Assembler (ASMPC, ASMPCL, LIBEASM, LIBELINK, etc.):

```
//SYSLIB    DD  DSN=SYMREF,DISP=SHR
//          DD  DSN=SYM&Q,DISP=SHR
//          DD  DSN=SYM&U,DISP=SHR
//          DD  DSN=SYMLIB,DISP=SHR
//          DD  DSN=SYMREL,DISP=SHR
//          DD  DSN=SYS1.MACLIB,DISP=SHR
```

where:

SYMREF is a dummy PDS with the correct largest block size,
SYM&Q is the private library (specified via Q parameter),
SYM&U defaults to SYMUSR.

NOTE: SYS1.MACLIB must be concatenated after all SYMxxx libraries.

2.3 JCL PROCEDURES

To simplify the execution of assemblies, linkage editing, and utilities in an Intercomm environment, a number of Job Control Procedures are supplied with the Intercomm system as members on SYMREL. These procedures provide a straightforward, uniform means to:

- Add and update source programs on source program libraries.
- Assemble or compile programs from source program libraries, producing either object modules (assembler or compiler output) or load modules (linkage editor output) on appropriate libraries.
- Print and punch source programs and object decks.
- Patch load modules on load module libraries.
- Linkedit any combination of object and load modules to produce executable programs.

  Note: for MVS systems, programs must not be linkedited as RENT (reentrant) unless they really are reentrant. The Intercomm load modules on MODREL are not linked as either reentrant or reusable.

- Execute general utility programs.

Figure 2-1 is a list of procedure names and the general function performed by each procedure. The Intercomm System Manager should evaluate this list carefully to determine which Intercomm procedures should be utilized as a standard for the installation. Many of these procedures are used in the Intercomm installation JCL and for specific feature installation as described in this and other Intercomm manuals.

Name	Function
ASMOC	Assembler source--produce object module
ASMPC	Assembler source--(produce object module or no output)
ASMPCL	Assembler source--produce load module
ASMPCM	Assemble a macro--(produce object module or no output)
COBPC	COBOL-F source--(produce object module or no output)
COBPCL	COBOL-F source--produce load module
COBUPC	ANS COBOL source--(produce object module or no output)
COBUPCL	ANS COBOL source--produce load module (with NCAL option)
COBUPCLD	ANS COBOL source--produce dynamic load module (link INTLOAD)
COMPRESS	compress a PDS
COPY	copy PDS or member
DEFSYM	see Message Mapping Utilities
FORTLINK	compile and link FORTRAN module
INTASMF	see ASMF Users Guide
LIBCOBDL	update ANS COBOL--produce dynamic load module (link INTLOAD)
LIBE	update a source member
LIBEASM	update Assembler source--produce object module
LIBECOB	update ANS COBOL source--produce object module
LIBECOBL	update ANS COBOL source--produce load module (NCAL option)
LIBELINK	update Assembler source--produce load module
LKEDE	object & load module(s)--produce executable load module
LKEDO	object & load module(s)--produce executable load module
LKEDP	load module(s)--produce executable load module
LKEDPL1	PL/1 object or load modules--produce executable load module
LKEDT	load module(s)--produce executable Test Mode module

Figure 2-1. Intercomm JCL Procedures (Page 1 of 2)

Name	Function
MODLIB	create load library
OBJLIB	create object library
OPSCN	Assembler source program scan (OPSCAN utility)
PATCH	patch load module(s)
PLIXPC	PL/1-optimizer--produce object module
PLIXPCL	PL/1-optimizer--produce load module
PL1LOC	PL/1-F source--produce object module
PL1LPC	PL/1-F source--(produce object or no output)
PL1LPCL	PL/1-F source--produce load module
PMIPCH	punch source or object deck
PMIPRT	print source member listing
SYMGEN	see Message Mapping Utilities
SYMLIB	create source library

Figure 2-1. Intercomm JCL Procedures (Page 2 of 2)

Unit name SYSDA is used wherever direct access space allocation is required.

Listings of individual members may be obtained by using the following JCL:

```
//PROCLIB      DD     DSN=INT.SYMREL,DISP=SHR
//             EXEC   PMIPRT,Q=REL,NAME=procname
```

A Job Control Procedure is invoked by coding the procedure name in an EXEC statement, along with appropriate keyword symbolic parameters to supply the library and member names.

Figure 2-2 summarizes the parameters specified for each Intercomm procedure.

Except for some of the utilities, all procedures below also have symbolic parameters Q, U and P, with Intercomm-supplied default values Q=XYZ, U=USR, P=INT. Bracketed parameters below are optional.

Procedure	Parameters			Comments/Other Parms
ASMOC	NAME=	OMOD=	[D=]	
ASMPC	NAME=		[SYSGO=]	
ASMPCL	NAME=	LMOD=		
ASMPCM	NAME=		[SYSGO=]	
COBPC	NAME=			
COBPCL	NAME=	LMOD=		
COBUPC	NAME=			
COBUPCL	NAME=	LMOD=		
COBUPCLD	NAME=	LMOD=		Dynamic Linkedit not used
COMPRESS		DSN=	[S=]	
COPY		INDSN=	OUTDSN=	
FORTLINK	NAME=	LMOD=	[S=]	[S1=]
LIBCOBDL	NAME=	LMOD=		Dynamic Linkedit not used
LIBE				
LIBEASM	NAME=	OMOD=	[D=]	
LIBECOB	NAME=	OMOD=	[D=]	
LIBECOBL	NAME=	LMOD=		
LIBELINK	NAME=	LMOD=		
LKEDE	[OMOD=]	LMOD=	[D=]	OMOD optional if INCLUDE
LKEDO	[OMOD=]	LMOD=		statement in input stream
LKEDP		LMOD=		
LKEDPL1		LMOD=	[INPUT=]	[OVLY=] [PL1=]
LKEDT		LMOD=	[D=]	
MODLIB		VOLSER=	[D=]	
OBJLIB		VOLSER=	[D=]	[BLKSIZE=]
OPSCN	NAME=			
PATCH				[T=]
PLIXPC	NAME=		[PARM2=]	
PLIXPCL	NAME=	LMOD=	[PARM2=]	
PL1LOC	NAME=	OMOD=	[PARM2=]	
PL1LPC	NAME=		[PARM2=]	
PL1LPCL	NAME=	LMOD=	[PARM2=]	
PMIPCH	NAME=		[S=]	[T=]
PMIPRT	NAME=		[S=]	[T=]
SYMLIB		VOLSER=	[D=]	

Figure 2-2. JCL Procedure Parameter Summary

Note: for the following procedures the default Q value is other than XYZ:

LKEDPL1 - (null); MODLIB,OBJLIB,SYMLIB - SCR; OPSCN - REL.

The keyword symbolic parameters used are as follows:

Q

Common to all Intercomm procedures, this parameter defines the characters completing various library names used in the procedure. For example, if Q=TST is coded for a procedure which uses both symbolic and load module libraries, the names SYMTST and MODTST are generated by the procedure. One to five alphanumeric characters may be specified. The default is XYZ.

U

Common to all Intercomm source update, compile, assembly, and linkedit procedures, this parameter defines the characters completing the library name of the data set placed after the Q data set in a SYSLIB concatenation stream. One to five alphanumeric characters may be specified. The default is USR.

P

Common to all Intercomm Procedures, this parameter specifies a library name common prefix or high-level qualifier. For example, if P=INTERCOM, and Q=TESTS is coded for a procedure using a source library, the name INTERCOM.SYMTESTS is generated by the procedure. One to eight alphanumeric characters may be specified, the first of which must be alphabetic. If multiple qualifiers are used, then the parameter value must be in quotes, that is, P='A.B', and more than eight characters may be coded. The default is INT.

NAME

For those procedures which use a symbolic library, this parameter is coded to specify the name of a particular member (source program) to be assembled, printed, etc. It may be omitted if an override SYSIN DD statement is present in the JCL. The default is INVALIDNAME.

OMOD

For those procedures using an object module library, this parameter is coded to specify a particular name for the input or output object module. The default is GO.

LMOD

For those procedures using a load module library, this parameter specifies a particular name for the linkage editor output module. It may be omitted if a NAME statement is present in the linkedit input control stream. The default is GO.

S

For utility procedures (compressing, printing, punching) requiring control statement input, this parameter specifies the prefix of the PROCLIB containing the control statements. For example, S=SYS1 specifies the system procedure library SYS1.PROCLIB. The default is INT.

T

For certain procedures (printing, punching, patching) applicable to more than one type of library, T=SYM, OBJ, or MOD may be specified to indicate the type of library. The default is SYM.

D

Indicates the disposition of the output library data set as follows: for procedures which can optionally create a temporary data set, D=MOD must be coded to specify this processing option; when library creation procedures are used to add or replace members, D=MOD, OLD, or SHR may be coded. The default is OLD.

SYSGO

For assemblies only, to provide the name of a temporary partitioned data set which will receive an output object module from the assembly. The data set is deleted at end of job. If not specified, no object output is produced. If a qualified data set name, or a temporary name (starting with &&), is used, enclose the name in quotes. The default is NULLFILE.

VOLSER

For library creation procedures requiring specification of the volume serial number on which the library is to reside. One to six alphanumeric characters may be specified. The default is INT001.

PARM2

For PL/I procedures, this allows specification of additional compiler parameter (PARM='....') information without changing the parameter default values specified in the procedure (which would cause a reversion to installation SYSGENed defaults). Specify as PARM2=',parm[,...]'.

PL1

For the LKEDPL1 procedure to linkedit a dynamically loaded PL/I subsystem and/or subroutine, this provides the module name to be used in the linkedit step (LKED1) execution to resolve all external PL/I references (needed when Intercomm's dynamic linkedit not used).

OVLY

For LKEDPL1, denotes whether the resulting load module is to be an overlay structure (default is ',OVLY'). If OVLY= (no value), then an overlay structure is not created.

INPUT

For LKEDPL1, specifies the prefix of the lowest level name of the installation load library used to resolve external PL/I subroutine references via a LKED1.SYSIN statement such as INCLUDE PL1LIB(name). The default is MOD. Thus if the defaults are used for the P, INPUT and Q parameters, the PL/I subroutine library data set would be INT.MODXYZ.

The following parameters are explained under examples of the applicable procedures:

BLKSIZE, DSN, INDSN, OUTDSN.

2.3.1 Step Names

The following naming conventions apply to multistep procedures:

Step Name	Function
LIB	source update
ASM	assembly
COB	COBOL compile
LKED	linkedit
PL1L	PL/I-F compile
PLI	PL/I-optimizer compile

2.3.2 JCL Procedures for Source Updates, Compiles, Assemblies, Linkedits

```
//    EXEC ASMOC,Q=xxx,NAME=source-member,OMOD=object-member[,D=disp]
```

Assemble the source program on SYMxxx, placing the object module on OBJxxx using the OMOD name. If the OBJxxx data set is created and used in subsequent steps of the same job, then it is deleted at the end of the job. D=MOD must also be coded to specify this option.

```
//    EXEC ASMOC,Q=xxx,OMOD=object-member
(Source program deck)
```

Assemble the input stream program (using library SYMxxx for macro, etc., definitions) and store the object module on OBJxxx using the OMOD name.

```
//    EXEC ASMPC,Q=xxx,NAME=source-member[,SYSGO='output-data-set']
```

Assemble the named source program. No object output is produced unless SYSGO='output-data-set' is coded. If a cataloged sequential data set is named, the object module is added at the end of the data set. Otherwise a temporary sequential data set is created and used in subsequent steps of the same job, then is deleted at the end of the job. A "temporary" data set name may be specified, but '&&LOADSET' should not be used.

```
//    EXEC ASMPC,Q=xxx
//SYSGO DD SYSOUT=B
(source program deck)
```

In this example, an input stream source deck is being assembled, and the object output is to be punched instead of being written to a temporary data set. The Q=xxx parameter still defines a library to be used for macro definitions, COPY members, etc.

```
//    EXEC ASMPC,Q=xxx,NAME=source-member,SYSGO='library(member)'
```

In this example, the object program is added to or replaces an existing member of the cataloged partitioned data set (library), which must have 80-byte logical records.

```
//    EXEC ASMPCL,Q=xxx,NAME=source-member,LMOD=load-module-name
```

Assemble and linkedit the named source member from SYMxxx, creating or replacing the named load module on MODxxx. This statement may be followed by an input stream source deck, in which case the NAME parameter may be omitted. If linkage editor control input is required, it must follow a //LKED.SYSIN DD * statement. If the condition code from the assembly step is greater than 4, the linkedit step is bypassed.

```
//    EXEC COBPC,Q=xxx,NAME=COBOL-source-member
```

Analogous to ASMPC, for COBOL-F compilation.

```
//    EXEC COBPCL,Q=xxx,NAME=COBOL-source-member,LMOD=load-module-name
```

Analogous to ASMPCL, for COBOL-F compilation and linkedit.

```
//    EXEC COBUPC,Q=xxx,NAME=COBOL-source-member
```

Analogous to COBPC, for ANS COBOL compilation.

```
//    EXEC COBUPCL,Q=xxx,NAME=COBOL-source-member,LMOD=load-module-name
```

Analogous to COBPCL, for ANS COBOL compilation and linkedit of resident, overlay, or dynamically loaded (if Dynamic Linkedit used) programs.

```
//    EXEC COBUPCLD,Q=xxx,NAME=COBOL-source-member,LMOD=load-module-name
```

Analogous to COBUPCL, for ANS COBOL compilation and linkedit for a dynamically loaded program and including all needed COBOL load modules from SYS1.COBLIB. If Dynamic Linkedit is used (see Chapter 3), then use COBUPCL. Linkage editor control cards should be added to LKED.SYSIN for the subsystem load module name, and for INTLOAD. For example:

```
//  EXEC COBUPCLD,Q=USR,NAME=COBPROG,LMOD=COBPROG
//LKED.SYSIN DD *
      INCLUDE SYSLIB(COBPROG,INTLOAD)
      ENTRY COBPROG
      NAME COBPROG(R)
```

```
//    EXEC FORTLINK,Q=xxx,NAME=source-member,LMOD=load-module-name
[,S=PDSprefix,S1=PDSname]
```

where S and S1 default to SYS1.FORTLIB (the library containing IEYFORT, the Fortran compiler and Fortran subroutines for the linkedit). This procedure executes a compile and linkedit of a Fortran module.

// EXEC LIBCOBDL,Q=xxx,NAME=COBOL-source-member,LMOD=load-module-name

Analogous to LIBECOBL, for ANS COBOL source member update, compilation, and linkedit for a dynamically loaded program including all needed COBOL load modules, when Dynamic Linkedit not used.

// EXEC LIBE,Q=xxx
(control statements and data for program IEBUPDTE)

Execute the IBM utility program IEBUPDTE to change symbolic library SYMxxx. This program is described in the IBM Utilities manual, and permits an individual source member to be changed, added, or replaced. The member named in the utility control statement is searched for in the named library and the system update (LIB) and release (REL) libraries, so that it is possible to update a source program onto a private library without first copying the program from one library to the other.

Control statement and data examples:

```
//   EXEC LIBE,Q=USR
./   CHANGE NAME=PROG1
*    THIS IS A REPLACEMENT  FOR  THE STATEMENT  NUMBERED  00459370
```

```
//   EXEC LIBE,Q=USR
./   REPL NAME=PROG2,LIST=ALL
./   NUMBER NEW1=10000,INCR=1000
     (replacement deck for PROG2)
```

// EXEC LIBEASM,Q=xxx,NAME=source-member,OMOD=object-module[,D=disp]
(control statements and data for program IEBUPDTE)

Update and assemble the source program, producing an object module on the named library. The control input is normally an add, replace, or change for the member to be assembled. If the update is not successful (any IEBUPDTE diagnostic giving a nonzero return code), the assembly is not performed.

If data set OBJxxx is not cataloged, a temporary data set is created and used in subsequent steps of the same job, then is deleted at the end of the job. D=MOD must also be coded to specify this option.

// EXEC LIBECOB,Q=xxx,NAME=COBOL-source-member,OMOD=object-module
[,D=disp]

Analogous to LIBEASM, for ANS COBOL source member update and compilation.

// EXEC LIBECOBL,Q=xxx,NAME=COBOL-source-member,LMOD=load-module

Analogous to LIBELINK, for ANS COBOL source member update, compilation and linkedit of resident, overlay or dynamically loaded (if Dynamic Linkedit used, see Chapter 3) programs.

// EXEC LIBELINK,Q=xxx,NAME=source-member,LMOD=load-module-name
(control statements and data for program IEBUPDTE)

Update, assemble, and linkedit the source program, creating or replacing the named load module. If the update is not successful, the assembly and linkedit are not performed. If the assembly is not successful (return code greater than 4), the linkedit is not performed. Any linkage editor control input must be preceded by the statement //LKED.SYSIN DD *.

// EXEC LKEDE,Q=xxx,LMOD=load-module-name[,OMOD=object-module-name]

Linkedit a program for subsequent execution, storing the load module on library MODSCR. If this library is not cataloged, it may be created and used in subsequent steps of the same job, then will be deleted at the end of the job; specify D=MOD in this case (the default is SHR). Linkage editor control input may follow this statement; if no control input is provided, then OMOD=object-module must be coded to specify an object module on OBJxxx as input.

Control statement examples:

INCLUDE OBJLIB(omod1,omod2,...)	include object modules
INCLUDE SYSLIB(lmod2,lmod3,...)	include load modules
INCLUDE ddname(...)	data set defined on added DD statement

Multiple load modules may be processed in one execution of the linkage editor by interspersing linkage editor NAME control statements with input control statements. The LMOD parameter is not required in this case. If object module library OBJxxx was created in the same job by an assembly or compilation procedure (see ASMOC, COBOC, LIBEASM), then, if OMOD parameter is not specified, precede any control input by: //SYSLIN DD *

// EXEC LKEDO,Q=xxx,LMOD=load-module-name[,OMOD=object-module-name]

Linkedit one or more object and/or load modules, placing the load module on library MODxxx. Refer to procedure LKEDE for the remainder of the description of this procedure. Override the SYSLMOD DD statement if MODxxx does not exist.

// EXEC LKEDP,Q=xxx,LMOD=load-module-name

This procedure is analogous to procedure LKEDO, but no object module data sets are defined or made available for inclusion.

```
//    EXEC LKEDT,Q=xxx,LMOD=load-module-name
```

Analogous to procedure LKEDE, but with no object module data sets defined. The load module is placed in MODSCR.

NOTE: Procedures LKEDT and LKEDP define concatenations of private library, USR, LIB, and REL for the call library SYSLIB; in addition, procedures LKEDE and LKEDT specify the system COBOL and telecommunications libraries (SYS1.COBLIB and SYS1.TELCMLIB), so that included or called Operating System modules will be available to the linkage editor. For LKEDE and LKEDO, Q specifies only the object library suffix; the SYSLIB concatenation sequence starts with USR (U parameter).

```
//    EXEC LKEDPL1,Q=xxx,LMOD=load-module-name,INPUT=library-type,
      PL1=library-name,OVLY=
```

This procedure will linkedit PL/I programs including all required PL/I library subroutines, and then perform a final linkedit to include all necessary Intercomm modules. This is necessary, as during the final linkedit the automatic library mechanism must be disabled, while during the initial linkedit (when PL/I library routines are included) it must be enabled.

There are two steps, LKED1 (the PL/I library step) and LKED2 (the Intercomm step). During LKED1, PL/I programs are included from either a load or object library (or both if additional user libraries are specified) via the INPUT (INPUT=OBJ for object, MOD is default) parameter and using the ddname PL1LIB. In the LKED2 step, Intercomm modules are included from SYSLIB and the PL/I program(s) from the library defined by the ddname PL1. To include the modules from the first step simply code INCLUDE PL1(PL1). The OVLY parameter, if coded, will nullify the overlay option in the second linkedit.

```
//   EXEC LKEDPL1,Q=LIB,LMOD=INTERCOM,OVLY=
//*  OVERLAY NULLIFIED, INPUT=MOD DEFAULT USED
//LKED1.SYSIN DD *
     INCLUDE PL1LIB(PROG1)
     INCLUDE PL1LIB(PROG2,PROG3)
//LKED2.SYSIN DD *
     INCLUDE SYSLIB(Intercomm-modules,...)
     INCLUDE PL1(PL1)
/*
```

```
//    EXEC PLIXPC,Q=xxx,NAME=PL1-source-name[,PARM2=',options']
```

Compile a PL/I-optimizer program, as in PL1LPC. If the source is in-line, NAME need not be specified. PARM2 is as in PL1LPC.

// EXEC PLIXPCL,Q=xxx,NAME=PL1-source-name,LMOD=load-module-name
[,PARM2=',options']

Compile a PL/I-optimizer program and store the load module (without the PL/I library subroutine modules referenced) under the name specified in LMOD (GO used if LMOD absent); for resident, overlay, or dynamically loaded (when Dynamic Linkedit used) programs. NAME need not be specified if source is in-line. PARM2 is as in PL1LPC.

// EXEC PL1LOC,Q=xxx,NAME=PL1-source-name,OMOD=object-module
[,PARM2=',options']

Compile a PL/I-F program and generate an object module stored in OBJxxx under the name specified for OMOD (GO used if OMOD omitted). NAME need not be specified if the source is in-line. PARM2 is as in PL1LPC.

// EXEC PL1LPC,Q=xxx,NAME=PL1-source-name[,PARM2=',options']

Compile a PL/I-F program from SYMxxx. If the source is in-line, NAME need not be specified. If additional PARM options are required, code PARM2=',options' (for example, PARM2=',LIST').

// EXEC PL1LPCL,Q=xxx,NAME=PL1-source-name,LMOD=load-module-name
[,PARM2=',options']

Compile a PL/I-F program and store the load module (without the PL/I library subroutine modules referenced) under the name specified in LMOD (GO used if LMOD absent); for resident, overlay, or dynamically loaded (when Dynamic Linkedit used) programs. NAME need not be specified if source is in-line. PARM2 is as in PL1LPC.

2.3.3 JCL Procedures for Utility Executions

The following procedures can be used to perform common utility operations (data set copy, data set member print/punch/patch/scan, library creation). The IBM Utilities manual describes the functions of each program in detail. Some of the procedures must be modified by the user to specify appropriate volumes for a given installation. The P and Q override parameters may be used (except where noted), but the U override parameter does not apply.

// EXEC COMPRESS,DSN='data-set-name'[,S=Proclib-prefixname]

Compress an individual library (using utility program IEBCOPY), and release any excess space available in the data set after compressing. Control statement input for this procedure is contained in the released member COMPSYS which must be put on the PROCLIB specified by the additional parameter S=prefix. If the system procedure library is used, specify S=SYS1 (the default is INT).

```
//    EXEC    COPY,INDSN='INT.SYMCHG',OUTDSN='INT.SYMLIB'
      COPY    INDD=SYSUT1,OUTDD=SYSUT2
      SELECT MEMBER=((PROGX,,R))
```

In this example, a member of a private source library (SYMCHG) is copied into SYMLIB. By supplying additional DD statements and control statements, more than one operation may be done in a single step.

Note: the COMPRESS and COPY procedures do not use the Q and P parameters.

```
//    EXEC PMIPCH,Q=xxx,NAME=source-member
```

Punch the named member of library SYMxxx.

```
//    EXEC PMIPCH,Q=xxx,NAME=object-module,T=OBJ
```

Punch the named member of library OBJxxx.

```
//    EXEC PMIPRT,Q=xxx,NAME=source-member
```

Print the named member of library SYMxxx.

NOTE: PMIPCH and PMIPRT use the IBM utility program IEBPTPCH; control statements for these procedures are contained in the released members PMIPCH1 and PMIPRT1 which must be put on the PROCLIB specified by the additional parameter S=prefix. If the system procedure library is used, specify S=SYS1 (the default is INT).

```
//    EXEC PATCH,Q=xxx[,T=library-type]
(control statements for program IMASPZAP)
```

Print and/or change selected data in load modules or object modules, using the IBM utility program IMASPZAP (described fully in the IBM Service Aids manual).

Object modules may be ABSDUMPed and the desired data located before changes are made. If the IMASPZAP program was not included in the operating system link library, a JOBLIB or STEPLIB DD statement is required. A STEPLIB DD statement may be added to the procedure if necessary. T defaults to MOD.

Control statement examples for IMASPZAP:

```
DUMP  (T)   member        [csect]
NAME        member        [csect]
VER         hex-location  hex-data,hex-data,.....
REP         hex-location  hex-data,hex-data,.....
```

// EXEC OPSCN,Q=xxx,NAME=source-member

This procedure executes the Intercomm-supplied utility OPSCAN which scans an Assembler source library member (or sequential data set) and selects all statements having a recognizable operation code field other than standard instructions. The selected statements may be directed to a printer, and will include all macro instructions (Intercomm and Assembler), CALLs, COPY references, conditional assembly statements, entry points, external references, and control sections, as well as other significant details.

Standard instructions are comment statements, machine operation codes (including privileged operations, SPM, TS, and floating-point feature instructions), selected extended mnemonic operation codes (BNE, BH, B, etc.) and selected Assembler operation codes (DC, EQU, CNOP, USING, EJECT, etc.)

The operation code scan accommodates free-form statements as specified for the OS/VS Assembler Language. Continuation lines of the selected statements are also printed.

// EXEC SYMLIB,Q=xxx,VOLSER=serial

Create and catalog a source library named SYMxxx, to be allocated on the specified volume. Space parameters supplied with the procedure allocate 10 directory blocks and one cylinder initially, with two additional cylinders obtained (up to fifteen times) each time the data set becomes filled. DCB information is copied from data set SYMLIB, which must be cataloged and mounted when this step is run. The D parameter defaults to NEW, the Q parameter to SCR.

// EXEC MODLIB,Q=xxx,VOLSER=serial

Analogous to procedure SYMLIB, for creating and cataloging load module library MODxxx. Q defaults to SCR and D to NEW.

// EXEC OBJLIB,Q=xxx,VOLSER=serial[,BLKSIZE=block-size]

Analogous to procedure SYMLIB, for creating and cataloging object module library OBJxxx. An empty member named GO is created to ensure proper functioning of the linkage edit steps of Intercomm-supplied JCL procedures. DCB information is specified in the procedure. A block size of 400 is the default, as larger blocks cannot be read by all linkage editors. The D parameter defaults to NEW, and Q to SCR.

Intercomm utilities for log (journal) printing and analysis, data set creation and loading, BTAM simulator input creation, source member compares, etc. are described elsewhere in this and other Intercomm manuals. Additionally, system cross-reference and maintenance utilities are described in the ASMF Users Guide.

2.4 SYSTEM INSTALLATION AND MAINTENANCE RESPONSIBILITIES

In any on-line system environment, it is necessary to develop a distribution of responsibility to installation personnel involved with the ongoing operation of the system. Three different user categories of Intercomm personnel are required:

- The System Manager(s) - System programmers responsible for coordination of all system specifications, system program maintenance, and operating procedures.
- The Application Group(s) - Project leaders and programmers responsible for design and implementation of application subsystems.
- Central Location Operations Staff - Responsible for the actual scheduling and operation of the central CPU.

Many responsibilities overlap in these functional areas. An installation must be flexible and above all establish orderly communications methods between the user personnel. Each Intercomm installation must develop its own distribution of responsibilities for its personnel depending on the scope of the on-line system. Requirements obviously vary from a staff of three to hundreds of associated programmers, analysts, system programmers, operators, management, etc.

In general, the responsibility for maintaining the Intercomm System lies in the areas of:

1. Intercomm System Program Maintenance via the ASMF Facility
2. Table Maintenance
3. Execution Load Module Maintenance
4. Procedures for Testing and Live Execution
5. System Tuning
6. Problem Reporting
7. Backup and Recovery Procedures

The following list represents a suggested set of guidelines in assignment of responsibilities for each category of installation personnel.

2.5.1 The Intercomm System Manager(s)

- General liaison with SDA

  -- Documentation updates and new editions
  -- Microfiche listings and updates of Intercomm source modules
  -- Early Warnings - monthly publication of outstanding problem reports and solutions
  -- Technical Information Bulletins - non-product problem resolution suggestions
  -- SM (system modification) maintenance of Intercomm system
  -- New release distribution
  -- Problem reporting, tracking, and resolution

- Initial system installation

- Production system generation and maintenance

  -- Definition of network configuration to Intercomm
  -- Definition of subsystems (applications) to Intercomm
  -- Ongoing system tuning as production environment changes
  -- Application and testing of official and experimental SMs
  -- Dump analysis and problem solution

- Maintenance of Intercomm libraries and tables (may include modifications to Intercomm and/or user exit routines for startup, restart, closedown, etc.)

- Control and coordination of terminal test sessions
  -- Add new application modules to linkedit
  -- Add new table entries to system tables
  -- Relinkedit Intercomm test system
  -- Distribute test session output (snaps, dumps, log, etc.)

- Coordination of live (production) system with application project leaders and operations personnel
  -- Installation standards maintenance
  -- Update live system with tested modules and tables
  -- Develop operational procedures as required
  -- Create and maintain a "run book" for operations personnel
  -- System expansion planning

- Analysis of system messages, log and statistics reports from live system for system tuning and problem reporting

- Development of procedures for system backup and restart

- Intercomm education coordination for system and applications staff

2.5.2 The Application Group(s)

- Maintenance of existing (live) application programs
- Development, coding, and comprehensive testing of new applications
- Assign specific identifiers following standards provided by the System Manager(s) for: verbs (transaction identifiers), subsystem codes and entry point names, mapping names, and other required table specifications
- Communicate to System Manager(s) when table maintenance is required for testing: new verbs, new subsystems (program modules), new utility table entries, etc.
- Communicate to System Manager(s) when a new module is to be added to the live system (requires a linkedit of production module)

2.5.3 Central Location Operations

- Start system selection of options (for example, JCL considerations) under direction of System Manager(s)
- Notify System Manager(s) immediately in the event of hardware or software failure and prepare "trouble" report stating cause of failure and corrective action.
- Close down system at direction of System Manager(s)
- Start log printing and analysis procedures, or any related off-line jobs to be executed after closedown or failure
- Restart system after failure at direction of System Manager(s)
- Periodically back up disk packs containing system libraries

2.6 STANDARDS

In planning an orderly Intercomm installation, the System Manager(s) and Application Group(s) may wish to standardize certain conventions for Intercomm libraries, programs and identifiers for Intercomm transactions and associated table specifications.

Intercomm library naming conventions are described in full in this chapter; program naming conventions must be controlled by the System Manager(s) to avoid duplications. Additionally, control must be exercised over file DD statement and data set names, terminal names, Store/Fetch and DDQ key names, etc.

Several different applications may be operating under the control of Intercomm and each of these applications may consist of several different transactions. For example, an order entry application may have different transactions for shipment, receipts, back order processing, stock status, etc.

A transaction under Intercomm has the following components:

- Input message from terminal
- Processing program(s) (subsystems and subroutines)
- Output message to terminal
- Data file(s) and/or data base access

The following basic identifiers are required in the Intercomm system to control (direct) the processing of that transaction:

1. Input message verb (transaction code)
2. Subsystem code and associated program entry point name
3. Message Mapping Utility map group definitions existing as members in this utility's related files and referenced by application subsystems.
4. File DD statement(s) and data set names.

The System Manager(s) may define standards for coding verbs, subsystem codes, program names, MMU map group names, and file names (if applicable). Assume an installation has four application areas: A, B, C, D. The System Manager(s) might define the following standards for basic identifiers:

Application Identifier	A	B	C	D
Verb (4 characters)	AAxx	BBxx	CCxx	DDxx
Subsystem Code (2 1-byte values)	A{x } {nnn}	B{x } {nnn}	C{x } {nnn}	D{x } {nnn}
Program Entry Point name (8 characters)	AAxxxxxx	BBxxxxxx	CCxxxxxx	DDxxxxxx
Map Group Name (1 to 7 characters)	MGAAxxx	MGBBxxx	MGCCxxx	MGDDxxx

where x is any character and nnn is any number (from 0 to 255) selected by the application project leader.

2.7 SYSTEM CONTROL FUNCTIONS AND TABLES

System Control Functions comprise those areas of table specification and related program logic which control the general operation of the Intercomm environment. The System Parameter List (SPA), discussed in Chapter 3, "Message Management," includes specification of many control variables affecting Intercomm execution. In general, these variables consist of time-delay values (indicating such things as checkpoint intervals, statistics intervals, etc.), control values (such as subsystem dispatching, security, message logging and message volume thresholds, etc.) and indicators controlling program logic (mode of operation, subtasking, etc.).

Intercomm Dispatcher routines are discussed in Chapter 4. Other system features connected with Intercomm installation, linkedit and execution are described in Chapters 3, 7 and 8. Implementation of the Resource Management functions of Intercomm is discussed in Chapter 5. The File Handler is described in Chapter 6. Edit and Output specifications are described in Chapter 3 and the Utilities Users Guide. Logging and restart/recovery specifications are discussed in Chapters 9 and 12, security options in Chapter 10, and system tuning recommendations in Chapter 11. Specifications for Front End interfaces and for special features are described in the applicable manuals.

Figure 2-3 lists the Intercomm global tables and corresponding SET symbol tables which may be modified by the user as the various Intercomm support features are utilized. Before a new installation, or a reinstallation, of Intercomm, the SET tables must be moved from SYMREL to SYMLIB and then modified according to expected user needs, or the existing installation. For a new installation, it is primarily necessary to modify SETGLOBE for the operating system in use, the type of Front End to be used, and the types of file access to be used. SETENV is described in the BTAM Terminal Support Guide and may optionally be modified to suppress support for teleprocessing devices which will not be installed. However, if a VTAM Front End is used exclusively, SETENV does not need to be modified as it applies primarily to BTAM/TCAM Front Ends. The DDQ (see Dynamic Data Queuing Facility and Log Analysis (see Chapter 12) tables provide recommended default settings and need only be adjusted to conform to existing installation specifications, or as the facilities are used in a production environment.

GLOBALS	SETTINGS	FUNCTION
INTGLOBE	SETGLOBE	Systemwide Support Requirements
ENVIRON	SETENV	Front End Support Requirements
DDQENV	DDQENV	DDQ Facility Requirements
LOGDCLGB	LOGSETGB	Log Analysis Utility Requirements

Figure 2-3. Intercomm Global Tables

2.7.1 System Global Tables (INTGLOBE, SETGLOBE)

The set of global specifications which control assembly of the SPA and other system routines are the member INTGLOBE defining globals indicating requirements for specific Intercomm features, and the member SETGLOBE which provides user assigned values for the defined globals. In general, these specifications pertain to the operating system, interregion communication, resource management options, data base management system interface requirements, File Handler options, EDIT and OUTPUT options, Dispatcher specifications, etc.

Figures 2-4 and 2-5 illustrate the members INTGLOBE and SETGLOBE as released. As these members vary from release to release, the user's Intercomm Support Manager should examine a listing of these control variables prior to effecting any change and subsequent reassembly of the System Parameter List, and other system programs conditionally assembled with these members. A global cross-reference program (IAIMGOCR) is available to Intercomm users with Product Maintenance agreements, to facilitate determination of which modules require reassembly when a SETGLOBE setting is changed (see ASMF Users Guide). A general list of affected system modules is provided in the Installation Guide.

```
********************************************************************** X
      INTGLOBE - GENERAL SYSTEM FEATURES:
GBLB  &VSSYSTM           ON IF RUNNING UNDER VS1 OR VS2
GBLB  &SYS370            USE OF S/370 INSTRUCTION SET
GBLB  &MVS               VS2 RELEASE 2 OR MORE.
GBLB  &VS2               SVS
GBLC  &MRSVC          INTERCOMM INTERREGION SVC (MRS, ESS, VS, MVS)
GBLA  &FASTSVC           FAST-SNAP SVC NUMBER
GBLC  &INTSVC            DATA BASE INTERREGION SVC                    X
      FRONT-END CHARACTERISTICS:  USED IN BTAM/VTAM MODULES
GBLB  &BTAM              BTAM (INC. GFE)  CONFIGURATION
GBLB  &VTAM              VTAM             CONFIGURATION
GBLB  &TIMSTMP           TIME-STAMP ON RESPONSES TO F.E. CMD          X
      RESOURCE MANAGEMENT:
GBLB  &RM                RESOURCE AUDITING
GBLB  &RMSTATS           RM STATISTICS GATHERING.
GBLB  &RMACCT            BUCKET ACCOUNTING SWITCH.
GBLB  &RMPOOLS           SUPPORT USER POOLS.
GBLB  &RMINTEG           RESOURCE MGMNT CORE INTEGRITY CHCK.          X
      DISPATCHER:
GBLA  &NUMWQES           NUMBER OF WORK QUEUE ELEMENTS                X
```

Figure 2-4. INTGLOBE (Page 1 of 2)

```
      FILE HANDLER:
GBLB  &IAM                IAM FILES USED
GBLA  &RPTINTV            FILE STATISTICS REPORT INTERVAL
GBLA  &FHSTATS            NUMBER OF DSCT STATISTICS BUCKETS
GBLB  &ISAM               ISAM FILES USED
GBLB  &AMIGOS             AMIGOS FILES USED
GBLB  &VSAM               VSAM FILES USED
GBLB  &DYNALOC            DYNAMIC-ALLOCATION ROUTINES DESIRED
GBLB  &VSISAM             ISAM/VSAM COMPATIBILITY REQUIRED              X
      EDIT UTILITY:
GBLB  &DELCHNG            NO CORRECT/CHANGE FACILITY USED
GBLB  &EDERRS             NO MAXIMUM FOR EDIT ERRORS SENT
GBLA  &EDERMAX            MAXIMUM NUMBER OF EDIT ERRORS                 X
                          (USED ONLY IF &EDERRS=0)
GBLB  &OPTRPT             SEND ERRORS FOR OPTIONAL PARMS                X
      OUTPUT UTILITY:
GBLB  &TCAM               FRONT END IS BASIC TCAM ONLY
GBLB  &DDQBACK            DYNAMIC DATA Q'S - AUTO INPUT
GBLB  &BROAD              NO BROADCAST GROUPS
GBLB  &RPTBLE             NO REPORTS TO TAPE
GBLB  &ALTRPT             NO ALTERNATE REPORTS
GBLB  &OUTEXIT            NO USER OUTPUT EXIT                           X
      DL/I SUPPORT:
GBLB  &DLI                DL/1                                          X
      TOTAL SUPPORT:
GBLC  &TOTDESC            TOTAL DATA BASE DESCRIPTOR
GBLA  &TOTMOD             SETTING:1 IF ATTACHED, 2 IF SEP TOT REG
GBLC  &TOTSVC             TOTAL INTERREGION SVC NUMBER                  X
      RJE FACILITY:
GBLB  &RJEWTO             INTR RJE TO INFORM OP OF EACH JOB
GBLC  &RJECLSA            OUTPUT CLASS TRANSFORMATION FOR CL A
GBLC  &RJECLSB            OUTPUT CLASS TRANSFORMATION FOR CL B
GBLB  &AUTORDR            ON IF RJE IS TO AUTO START A RDR
GBLC  &RDRNAME            READER NAME TO BE USED
GBLC  &RDRID              RJE RDR ID-('.S','.P2',ETC.)
GBLA  &NUMJOBS            JOB THRESHOLD FOR AUTO START                  X
      MULTIREGION SUPPORT:
GBLB  &MULTREG            MULTI-REGION SUPPORT REQUESTED                X
      LOGINPUT FACILITY:
GBLC  &GENTERM            DUMMY TERMINAL-ID
GBLA  &LOGINTM            LOGINPUT DISPATCH INTERVAL
GBLA  &LGINRTD            LOGINPUT REAL-TIME DIVISOR
*******************************************************************************
```

Figure 2-4. INTGLOBE (Page 2 of 2)

```
*************************************************************************X
               SETGLOBE - GENERAL SYSTEM FEATURES:
&VSSYSTM SETB  1                  DEFAULT TO VS
&SYS370  SETB  1                  USE 370 INSTRUCTIONS
&VS2     SETB  0                  DEFAULT TO NOT VS2 (SVS)
&MVS     SETB  1                  DEFAULT TO MVS
&MRSVC   SETC  '013'              INTERCOMM INTERREGION SVC NOT USED
&FASTSVC SETA  13                 FAST-SNAP SVC NOT USED
&INTSVC  SETC  '013'              DATABASE INTERREGION SVC NOT USED
&VS2     SETB  (&VS2 OR &MVS)          .VS AND S/370
&VSSYSTM SETB  (&VSSYSTM OR &VS2)      .GLOBAL INTER-
&SYS370  SETB  (&SYS370 OR &VSSYSTM)   .DEPENDENCIES                    X
               FRONT-END CHARACTERISTICS:
&BTAM    SETB  1                  BTAM FRONT-END IS IN USE
&VTAM    SETB  1                  VTAM FRONT-END IS IN USE
&TIMSTMP SETB  0                  NO TIMSTAMPS ON F.E. CMD RESP         X
               RESOURCE MANAGEMENT:
&RM      SETB  1                  RESOURCE MANAGEMENT
&RMSTATS SETB  1                          STATISTICS
&RMACCT  SETB  1                          ACCOUNTING
&RMPOOLS SETB  1                          CORE POOLS
&RMINTEG SETB  0                          CORE POOL INTEGRITY CHECK
&RM      SETB  (&RM OR &RMINTEG)  INTEG CHECK REQUIRES RCBS              X
               DISPATCHER:
&NUMWQES SETA  120                NUMBER OF WORK QUEUE ELEMENTS         X
               FILE HANDLER:
&RPTINTV SETA  600*300            600 SECS = 10 mins
&FHSTATS SETA  5                  NUMBER OF DSCT STATISTICS BUCKETS
&ISAM    SETB  1                  ISAM FILES USED
&AMIGOS  SETB  0                  AMIGOS FILES NOT USED
&IAM     SETB  0                  DEFAULT - NO IAM SUPPORT
&ISAM    SETB  (&ISAM OR &AMIGOS OR &IAM) ISAM IF AMIGOS OR IAM
&VSISAM  SETB  1                  ISAM/VSAM COMPATIBILITY
&VSAM    SETB  1                  VSAM FILES USED
&VSAM    SETB  (&VSAM OR &VSISAM)    FORCE S/370 & VS
&VSSYSTM SETB  (&VSSYSTM OR &VSAM)   IF VSAM OR VSISAM
&SYS370  SETB  (&SYS370 OR &VSSYSTM)
&DYNALOC SETB  1                  GENERATE DYNAMIC-ALLOCATION ROUTINESX
               EDIT UTILITY:
&DELCHNG SETB  1                       NO CANCEL/CORRECT FACILITY
&EDERRS  SETB  0                  SEND NO MORE THAN &EDERMAX ERROR MSGS
&EDERMAX SETA  5                       MAXIMUM NUMBER OF ERRORS/MESSAGES
&OPTRPT  SETB  0                  SUPPRESS ERROR MSG IF PARM IS OPTIONAL  X
```

Figure 2-5. SETGLOBE (Page 1 of 2)

```
                OUTPUT UTILITY:
&TCAM     SETB  0                   FRONT END IS NOT BASIC TCAM
&DDQBACK  SETB  0                   DEFAULT TO NO DDQ AUTO INPUT
&BROAD    SETB  0                   BROADCAST GROUPS IN USE
&RPTBLE   SETB  0                   REPORTS TO TAPE IN USE
&ALTRPT   SETB  0                   ALTERNATE REPORTS IN USE
&OUTEXIT  SETB  1                   NO USER OUTPUT EXIT                  X
                DL/I SUPPORT:
&DLI      SETB  0                   DL/I NOT IN USE                      X
                TOTAL SUPPORT:
&TOTDESC  SETC  'XXXXXX'            TOTAL DATA BASE DESCRIPTOR
&TOTMOD   SETA  1         SETTING:  1 IF ATTACHED, 2 IF SEP TOT REG
&TOTSVC   SETC  'NUL'               NO INTERREGION COMM NECESSARY        X
                RJE FACILITY:
&RJECLSA  SETC  'M'                 DEFAULT TRANSFORMATION FOR CLASS A
&RJECLSB  SETC  'N'                 DEFAULT TRANSFORMATION FOR CLASS B
&RJEWTO   SETB  1                   DEFAULT
&RDRNAME  SETC  'RJERDR'            DEFAULT
&RDRID    SETC  '.S'                DEFAULT
&NUMJOBS  SETA  10                  DEFAULT                              X
                MULTIREGION SUPPORT:
&MULTREG  SETB  1                   MULTIREGION SUPPORT REQUESTED        X
                LOGINPUT FACILITY:
&GENTERM  SETC  '$$$$$'             M.S.G. OR LOGINPUT TID
&LOGINTM  SETA  3                   .3 SEC TO DISP LOGINPUT
&LGINRTD  SETA  5                   LOGINPUT REAL-TIME DIVISOR
*************************************************************************
```

Figure 2-5. SETGLOBE (Page 2 of 2)

2.7.2 System Control Tables

As described in Chapter 1, there are several tables which are required for the proper functioning of the Intercomm teleprocessing monitor. Some of these tables must contain entries for Intercomm system control and command processing routines. As listed in Figure 2-6, such tables are released with the Intercomm recommended entries and contain a COPY statement to copy in a user-coded table of additional installation-dependent entries at assembly time. The user COPY member for the table should be stored on SYMUSR and may thus be carried to new releases without affecting system requirements. The load module may reside on MODUSR or MODLIB.

TABLE	USER COPY MEMBER	FUNCTION
BTVRBTB	USRBTVRB	Front End Verb Table
INTSPA	USERSPA	System Parameter Area
INTSCT	USRSCTS	Application Subsystems
REENTSBS	USRSUBS	System and User Subroutines
PMIVERBS	USRVERBS	Edit Facility Control Table

Figure 2-6. Intercomm Tables with User COPY Members

The tables listed in Figure 2-6 are all described in Chapter 3. Entries may be deleted (if function not used) or modified for all tables except REENTSBS. Subsystem codes for system verbs and subsystems should not be modified, and are also listed in Chapter 3.

Sample tables are provided on SYMREL for many tables, which may be replaced or modified as necessary for a specific installation. Such sample tables include:

BTAMSCTS	Front End Terminal Queues (BTAM/TCAM)
FENETWRK	Front End Network Definitions (BTAM)
VTSAMP	Sample VTAM Front End Tables
DDQDSTBL	DDQ Facility Table
IXFDSCTn	Data Set Control Table
LOGCHARS	MMU Device Processing Definitions
MMUVTBL	MMU Vector Table
MRMCT	Multiregion Communication Table
NEWPOOLS	Resource Management Pools Table
PADDTBLE	Edit Utility Pad Characters
PAGETBLE	Page Facility Terminal Table
PMIBROAD	Broadcast Terminal Table
PMIDEVTB	Back End Device Characteristics Table
PMIFILET	File Tables (Change/Display Utility)
PMIRDT00	Multiregion Description Table
PMISTATB	Back End Terminal Definitions
PTRNTBL	Output Utility Editing Patterns
RPT.....	Output Utility System Reports (1-50)

These tables are further described in this manual or in the applicable facility manuals. See also Appendix A.

Chapter 3

MESSAGE MANAGEMENT

3.1 INTRODUCTION

This chapter defines table specifications for user-written message processing application programs, which under Intercomm are called subsystems. Based upon resource requirements and user-coded table specifications, all subsystems in concurrent execution affect one another's throughput and response time. Procedures to optimize system performance are described, along with techniques for implementing message processing control facilities.

In particular, this chapter documents the following subjects:

- General message flow and cancellation processing
- The Front End Verb Table
- Back End table specifications for message utilities
- Message processing facilities
- The System Parameter Area
- The Subsystem Control Table
- Subsystem processing specifications
- Subsystem residency considerations
- Subsystem interfaces and linkedit considerations
- Subroutine interfaces and linkedit considerations
- Generalized subtasking
- Time controlled message processing

In addition to other referenced documentation, this chapter is to be used in conjunction with the following Intercomm manuals:

- Basic System Macros
- BTAM Terminal Support Guide
- COBOL Programmers Guide
- Utilities Users Guide
- PL/1 Programmers Guide
- Message Mapping Utilities
- Assembler Language Programmers Guide

3.2 GENERAL MESSAGE FLOW

The Intercomm BTAM/VTAM or TCAM Front End interface acts as a message handler between the terminal network and the Subsystem Controller in the Intercomm Back End which controls processing by application programs. The Front End receives messages from terminals, formats message headers, validates transactions and routes them for Front End command processing, or to the appropriate subsystem. Once a response has been generated, the Front End will prefix, insert and/or append terminal control characters, as required, queue the message for the proper terminal, and transmit it to the destined device. Intercomm facilities for editing and formatting messages are the Message Mapping Utilities for mapping input and output messages, or the Edit Utility for input messages and the Output Utility for output messages. Additionally, a Change/Display Utility is provided for display and/or update of user files, which itself interfaces with the Edit and Output Utilities.

3.2.1 Input Messages

To allow the Intercomm Front End to process a message from a terminal, all input messages received by Intercomm must follow the standard Intercomm format:

```
verb$text@
```

verb

represents the transaction code. It must be one to four alphameric characters, and is defined in the Verb Table used by the Front End to validate the incoming message. Once the validity of a verb is established, a standard message header is prefixed to the message text.

If the subsystem does not use Message Mapping Utilities, then the Edit Utility may be used to preedit the message text to remove all terminal/format-dependent characteristics. In all cases, the input message is passed to the Back End via Queue Management routines. Messages not edited prior to queuing for subsystem processing may be edited prior to transferring control to the subsystem (COBOL, PL/1), or on request from the subsystem (Assembler Language). Alternatively, any subsystem may perform its own editing, or use the MMU subroutine MAPIN.

$

indicates a separator character. This may be:

- A special graphic character (comma, etc.)
- A New Line character
- A device-dependent carriage-return/line-feed character (CR/LF)

This systemwide separator character is defined at Intercomm installation time in the System Parameter List SPALIST macro, SEP parameter. It must also be defined by the global &SEPCHAR for the BTAM or TCAM Front End in the member SETENV.

text

indicates optional text data.

@

indicates End-of-Transmission (EOT, EOB, etc.). The particular character will depend on the hardware characteristics of the transmitting terminal.

The message may consist of only a verb with no text data following. In this case, no separator character is necessary. Alternate methods for providing the input verb are described in Section 3.3.4, "Locked Verb Facility," and in the BTAM Terminal Support Guide for certain terminals where special keys can signify a verb request, such as the 3270 AID key processing and the ATTN key on a 2741. Support for AID processing is also provided via the TCAM and VTAM interfaces.

When Intercomm is unable to determine a verb (message routing) for an input message, that message is discarded and the following message is returned to the transmitting terminal:

NO VERB FOUND IN PREVIOUS MESSAGE STARTING xxxx

where xxxx is the first four characters received from the terminal.

3.2.2 Output Messages

Messages for transmission to the network, created by internal Intercomm processing or by the various subsystems, are passed via FESEND to the Front End and placed in terminal queues to await transmission. Figure 3-1 illustrates the relationship between the Intercomm components and the message queues.

The Intercomm Front End utilizes the Queue Management Routines of Intercomm to control all message queuing. If a terminal becomes nonoperational before message transmission is complete, the Intercomm Front End will either requeue the message or reroute it to an alternate terminal (if specified). A system control command (TDWN) is available to dynamically assign alternate devices. When an alternate device assignment exists, all subsequent output messages for the down terminal will be placed directly on the queue for the alternate terminal by the Front End queuing routines.

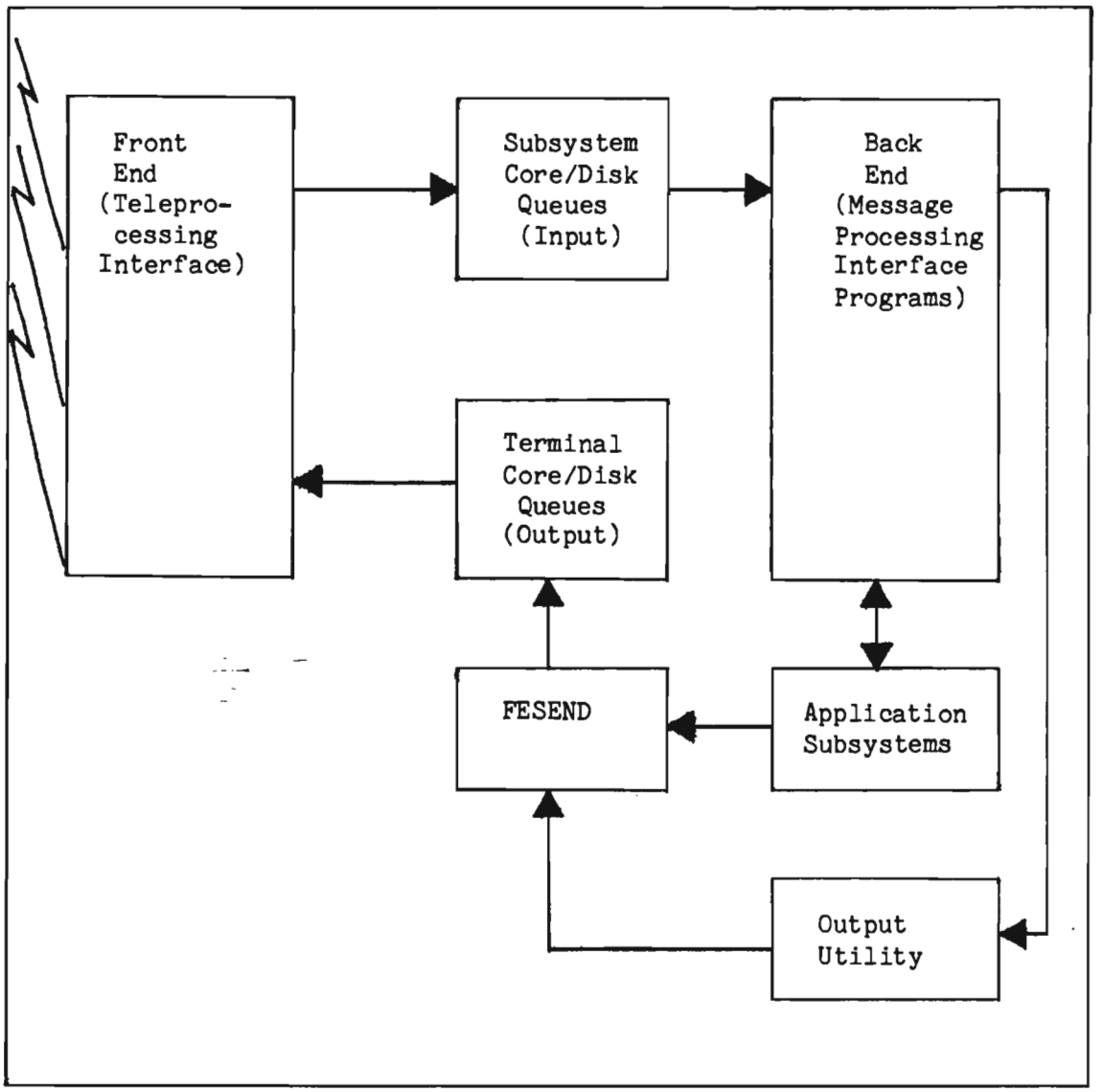


Figure 3-1. Front End/Back End Communication via Message Queues

All output messages must have message-ending characters (EOT/EOB/ETX, or other value, as appropriate to the device) coded at the end of the message. This character may be provided via:

- Output/MMU message formatting utilities, based on coding of the terminal's Back End DEVICE macro, EOT and/or EOB parameters
- Coded by the subsystem before passing the message to the Front End via FESEND (or FESENDC); see Programmers Guides.
- Added/replaced in the BTAM/TCAM Front End via the terminal's BDEVICE macro, ENDCHAR and/or LAST parameters
- Automatically suffixed, depending on device type, by the VTAM Front End, if appropriate.

3.2.3 Message/Subsystem Cancellation Processing

The following subsections describe cancellation processing in terms of message flow.

3.2.3.1 Message Cancellation User Exit--USRCANC

In certain situations, messages must be cancelled by the Subsystem Controller to prevent slowdown or failure of the entire system. The USRCANC routine, released as member PMICANC, is used to inform the terminal operator of this situation. The released USRCANC Csect may be modified to handle particular cases in a manner suitable to specific subsystems.

The USRCANC user exit will be called by the Subsystem Controller (SYCT400) when a message is cancelled for one of the following reasons:

- Program check or time-out (system return code is X'FF')
- I/O error (subsystem return code is X'12')
- No core available to process message or other unrecoverable error such as an output mapping error (subsystem return code is X'08')
- Subsystem stopped due to previous message cancellations (return code not applicable)

The error condition return code is duplicated into the logged message header, the address of which is in the fourth parameter passed to USRCANC (for all but the last reason).

Two types of calls can be issued by the Subsystem Controller to the USRCANC routine. The first is exercised when the message is cancelled due to an error condition. The second is issued if the subsystem assigned to process the message is not allowed to process further messages. This second condition arises only if a message has previously been cancelled and the user has chosen to exercise the SYCTTBL macro CANC parameter to stop the subsystem from further message processing.

3.2.3.2 Message Cancelled Condition

USRCANC is called with register 1 pointing to a parameter list that contains the following four addresses:

1. Address of message which was being processed
2. Address of SPALIST
3. Address of the Subsystem Control Table entry for the subsystem processing the message
4. Address of the logged message header (MSGHCON+1, that is, MSGHRETN, contains the Subsystem Controller return code value)

The first address above may point to an invalid location (or be zero) because the subsystem or MMU MAPIN processing may have freed the area before control was passed to USRCANC. If the subsystem frees the message area, then the message address in the parameter list must be set to binary zero. If MMU frees the message, it will set the message address to zero.

The released USRCANC routine generates and transmits an error message to advise the operator at the sending terminal that the message has been cancelled. This error message will indicate the reason for cancellation. (See the cancellation reasons above.) For a message cancelled condition, the USRCANC routine does not free the input message or any other area. Standard linkage conventions must be used.

3.2.3.3 Subsystem Stopped Condition

If a message was previously cancelled and the user has coded CANC=STOP on the associated SYCTTBL macro to cancel the subsystem, the parameter list passed via register 1 to the USRCANC routine will contain only the first three addresses listed above for the message cancelled condition. Called in this manner, the released USRCANC generates and transmits an error message to the sending terminal, then frees the message area and zeros the address in the parameter list, and finally returns a nonzero return code in register 15.

If the user modifies USRCANC and desires the message to be processed despite the CANC option, the return code must be F'-1' and the message may not be freed by USRCANC. Standard linkage conventions must be used.

3.3 THE FRONT END VERB TABLE

Incoming transactions from a teleprocessing device are identified by a transaction code, which under Intercomm is called a verb. Verbs are defined in the Front End Verb Table (BTVRBTB) via coding of a BTVERB macro for each user transaction code, and each system control command. Each BTVERB macro relates a verb to the subsystem which is to process the transaction via user-coded subsystem identifiers, called receiving subsystem codes. These codes are placed in the Intercomm message header constructed for the incoming message, and are subsequently used to search the subsystem table during message routing processing. See Appendix B for a detailed description of the Intercomm message header. Although the verbs must be unique, more than one verb may be processed by a specific subsystem, by specifying the same subsystem identifier codes.

3.3.1 Entries in The Verb Table

One BTVERB macro must be coded for each four-character verb to be accepted by the system. The macro parameters specify the actual verb, the receiving subsystem code of the message processing subsystem, message editing requirements, etc. To signify the end of the table, the last coded BTVERB macro must be followed by a PMISTOP macro. User verbs should be coded in a copy member USRBTVRB which is copied into the released BTVRBTB at assembly time, as illustrated in the BTVRBTB in Figure 3-2, or may be coded after Intercomm verbs, but before the PMISTOP. Intercomm verbs are called system commands and are all described in System Control Commands.

Assembly of the Front End Verb Table also produces an index (Csect BTVRBNDX) to BTVERB entries, providing a binary search capability via the module BINSRCH. This facility allows verbs to be grouped in any convenient order, such as by application area.

If more than 1000 BTVERB macros are defined, the global values (released as 1000) in FEMACGBL must be reset to the higher number desired to allow sorting of the greater number of verbs for the verb index. Additionally, use of Assembler H and/or a larger region size may be required for the assembly step of BTVRBTB.

```
BTVRBTB      CSECT
*
*         FRONT END
*
         BTVERB  VERB=TDWN
         BTVERB  VERB=TPUP
         BTVERB  VERB=STLN
         BTVERB  VERB=SPLN
         BTVERB  VERB=STLG
         BTVERB  VERB=SPLG
         BTVERB  VERB=STPL
         BTVERB  VERB=SPPL
         BTVERB  VERB=LOCK
         BTVERB  VERB=UNLK
         BTVERB  VERB=RLSE
         BTVERB  VERB=FLSH
         BTVERB  VERB=QHLD
         BTVERB  VERB=QRLS
         BTVERB  VERB=RVRS
         BTVERB  VERB=STAT                    BTAM/TCAM/GFE STATUS
*
*        SYSTEM COMMANDS
*
         BTVERB  VERB=NRCD,SSC=J              NORMAL CLOSEDOWN
         BTVERB  VERB=IMCD,SSC=J              IMMEDIATE CLOSEDOWN
         BTVERB  VERB=SECN                    CONTROL TERM. SECURITY ON
         BTVERB  VERB=SECF                    CONTROL TERM. SECURITY OFF
         BTVERB  VERB=DSPL,SSC=H,EDIT=YES,CONV=18000     DISPLAY
        BTVERB  VERB=CHNG,SSC=H,EDIT=YES,CONV=18000     CHANGE
        BTVERB  VERB=SWCH,SSC=B              MESSAGE SWITCHING
        BTVERB  VERB=SNBK,SSC=W              ECHO INPUT MESSAGE
        BTVERB  VERB=LOAD,SSC=L,SSCH=L,CONV=36000  LOADSCT SUBSYSTEM
        BTVERB  VERB=COPY,SSC=C,SSCH=C       COPY SUBSYSTEM - 3270'S
        BTVERB  VERB=FHST,SSC=R,CONV=36000   FILE STATISTICS DISPLAY
*
*        GPSS VERBS
*
         BTVERB  VERB=FILE,SSCH=G,SSC=P,CONV=36000
         BTVERB  VERB=TALY,SSCH=G,SSC=P,CONV=18000
         BTVERB  VERB=STRT,SSCH=G,SSC=P
         BTVERB  VERB=STOP,SSCH=G,SSC=P
         BTVERB  VERB=SNAP,SSCH=G,SSC=P,CONV=36000
         BTVERB  VERB=ABND,SSCH=G,SSC=P,CONV=36000
         BTVERB  VERB=LTRC,SSCH=G,SSC=P         START/STOP LINE TRACE
*
*        MMU COMMAND
*
         BTVERB VERB=MMUC,SSCH=M,SSC=M,CONV=18000
```

Figure 3-2. Released BTVRBTB (Page 1 of 2)

```
*
*       PAGE FACILITY COMMANDS
*
        BTVERB    VERB=PAGE,SSC=P,EDIT=YES,CONV=36000
        BTVERB    VERB=SAVE,SSC=P,EDIT=YES,CONV=36000
*
*       VTAM VERBS
*
        BTVERB    VERB=STLU
        BTVERB    VERB=SPLU
        BTVERB    VERB=RSLU
        BTVERB    VERB=VTCN
        BTVERB    VERB=VTST                    VTAM STATUS
**************************************************************************
*                                                                        *
*     A D D   U S E R   V E R B S   H E R E   V I A   C O P Y            *
*                                                                        *
**************************************************************************
        COPY      USRBTVRB
        PMISTOP
        END
```

Figure 3-2. Released BTVRBTB (Page 2 of 2)

The following illustrates a USRBTVRB (as released on SYMREL for use by new installations):

```
*
*     MULTIREGION COMMANDS
*
      BTVERB      VERB=COMM,SSC=K,CONV=18000
      BTVERB      VERB=LOKR,LOCKEXE=YES
      BTVERB      VERB=ULKR,LOCKEXE=YES
*
*     EXTENDED SECURITY COMMAND
*
      BTVERB      VERB=SECU,SSC=E
      TITLE      'APW CLASS WORKSHOP S/S VERBS'
      BTVERB      VERB=APW1,SSCH=A,SSC=1
      BTVERB      VERB=INQ1,SSCH=A,SSC=1
      BTVERB      VERB=UPT1,SSCH=A,SSC=1
      BTVERB      VERB=NEW1,SSCH=A,SSC=1
      BTVERB      VERB=APW2,SSCH=A,SSC=2
      BTVERB      VERB=INQ2,SSCH=A,SSC=2
      BTVERB      VERB=UPT2,SSCH=A,SSC=2
      BTVERB      VERB=NEW2,SSCH=A,SSC=2
      .
      .
      .
```

3.3.2 Short Verbs

Intercomm provides a facility to allow verbs with a length of one, two or three characters to be accepted, instead of only verbs of the standard four-character length. These short verbs are padded on the right with Xs before the verb is validated against the Verb Table. The BTVERB entry for each short verb must contain the X padding.

3.3.3 Priority Verbs

Certain verbs may be specified as high-priority by coding HPRTY=YES in the BTVERB macro. The input message header will then be flagged so that the message will receive high-priority treatment on any subsystem or Front End queue which specifies the priority-queuing facility (via the PRYMSGS parameter of the SYCTTBL macro). Any messages generated in the course of processing these high-priority input messages will also receive high priority if message processing program logic is such that input message headers are copied before altering to create output message headers. The MSGHUSR byte in the input message header is set to a character P to identify priority verbs; subsystems altering or omitting this value will cause a message to lose its priority status on transfer to another queue.

3.3.4 Locked Verb Facility

For certain terminals where prefixing a message with a verb may be impractical, Intercomm provides a facility for locking the terminal to a verb. The verb is automatically inserted by the Front End for each message from the designated terminal. This may be accomplished by one or more of the following:

- Specifying LOCK=verb on the terminal descriptor (BTERM/LCOMP/LUNIT) in the Front End Network Table.
- Specifying AUTOLOK=YES on the verb descriptor (BTVERB)
- Issuing the LOCK system control command from another terminal or a subsystem.

Subsequent unlocking of the terminal from a specific verb may be accomplished dynamically by issuing the UNLK system control command.

When the LOCK parameter is specified via the terminal descriptor, the terminal is automatically locked to the specified verb at startup; therefore the first message input from the terminal does not need a verb. That message, and all subsequent messages, will automatically have the designated verb (and system separator) inserted between the Intercomm message header and the message text before queuing. When AUTOLOK is requested via the BTVERB macro, only the first message

requires a verb; subsequently the terminal is locked. Issuing the LOCK/UNLK system control commands may be done before terminal input is begun or to alter subsequent locked verb processing (status). The latter case applies particularly to restarted messages; the processing subsystem must issue an internal LOCK command if terminal locking is required for subsequent input.

Certain verbs may be defined as lock-exempt; that is, even if the terminal is locked to another verb, when the exempt verb is entered from the terminal, it is to be processed instead of the locked verb. This is designated by coding LOCKEXE=YES for the BTVERB macro, and is the default for certain system control commands. The LOCKEXE and AUTOLOK parameters of BTVERB are mutually exclusive. When executing under Multiregion, LOCKEXE also exempts terminal/region locking.

3.3.5 Conversational Verbs

An installation may optionally define certain terminals as conversational terminals and certain verbs as conversational verbs. If a conversational verb arrives from a conversational terminal, the terminal is quiesced (taken out of the polling list) and further input is ignored until a message has been written back to the terminal. This prevents a terminal from having more than one input message begin processing at one time. A routine is scheduled on a time interval to issue a time-out message to the terminal in the event that the subsystem to which the verb was directed does not respond within the specified time. The time limit for each verb is defined on the BTVERB macro. The presence of a nonzero time limit indicates a conversational verb. In normal operation, if a response does come back from the subsystem before the specified interval expires, the scheduled routine is cancelled. Conversational mode processing controls input messages only. Response to a conversational verb from a conversational terminal could be more than one output message.

This facility is implemented as follows:

1. Set the &CONVER global in SETENV to 1 if BTAM/TCAM used, and reassemble the BTAM Front End modules.

2. Code CONV=YES for all terminal BTERM/LUNIT/LCOMP macros for which this processing is desired.

3. Code the CONV parameter with the time-out value on the conversational verb's BTVERB macro.

If this facility is used in conjunction with the CONVERSE facility (described in the Programmers Guides), the time interval on the conversational verb should be slightly larger than the time interval passed from the application program to CONVERSE. Use of the CONVERSE facility is not recommended if message restart is used.

3.3.6 Separate Assemblies of Verb and Network Tables

Normally, the Front End Verb Table is coded with the Front End Network Table as one module. In cases where frequent changes of entries in the Front End Verb Table occur, or either table becomes very large, it may be coded and assembled as a separate module. The Csect and member name for the verb table must be BTVRBTB. Internal Csect or entry point names, generated by the first occurrence of a macro designating a major component, are used for accessing the Network Table, which may have any Csect name if assembled separately. When assembled separately, the load module name for the Network Table must be specified on the Intercomm linkedit generation ICOMLINK macro via the FETABLE parameter. The BTVRBTB is automatically included. In a Multiregion environment, these tables are included only in the control region. Sample Front End terminal tables are illustrated in the BTAM/TCAM/VTAM Terminal Support Guides.

3.4 BACK END TABLE SPECIFICATIONS FOR THE UTILITIES

The Intercomm utilities (Edit, Output, Change/Display, and the Message Mapping Utilities) are documented in the Utilities Users Guide and Message Mapping Utilities. This section describes specifications for the utilities of a nonapplication-oriented nature, that is, systemwide table specifications controlling the use of the utilities. In a Multiregion environment, these tables are required in the control region, and in each satellite region which uses the utilities and/or Intercomm subsystems. These tables are also required in a simulated or Test Mode Intercomm system. The following describes tables used by all the utilities, plus additional tables unique to the individual utility.

3.4.1 Station Table

The Station Table is core-resident in a Csect named PMISTATB. The table is created and maintained by the user. Individual entries in the table are created by use of the STATION macro (one for each device defined in the Front End Network Table). The end of the table is indicated by four bytes of hexadecimal 'FF', generated by the PMISTOP macro. Assembly of the Station Table produces a binary search index by terminal names (Csect STATINDX). The location in core of the PMISTATB Csect is pointed to by a V-type address constant in the field SPASTATB of the System Parameter Area. The member PMISTATB on SYMREL contains a sample Station Table which may be updated or replaced by the system manager to define the network configuration for the utilities.

The Station Table effectively creates five-character logical names for each terminal in the system, and relates that terminal to the device characteristics defined in the Device Table. General device characteristics for an individual terminal may be overridden by coding a DVMODIFY macro after the PMISTOP in the Station Table, and specifying the label of that DVMODIFY via the corresponding STATION macro.

The Station Table structure is as follows:

```
PMISTATB   CSECT
           STATION . . .
           STATION . . .
           STATION . . .
           STATION . . .
           STATION . . .
              .
              .
              .
           PMISTOP
           END
```

To add a new terminal to the system, the Station Table must be modified by adding a STATION macro entry before the PMISTOP macro. The Station Table is accessed by all the utilities, and for additional internal Intercomm functions, and therefore is required in all regions. If more than 1000 STATION macros are coded, the global table FEMACGBL must be modified as described for the BTVRBTB in Section 3.3.1.

3.4.2 Device Table

Created and maintained by the user, the Device Table is resident in a Csect named PMIDEVTB. Individual entries (one per terminal type) are created by use of the DEVICE macro (specifying message editing and formatting control characteristics of each device type). The end of the table is indicated by four bytes of hexadecimal 'FF', generated by the PMISTOP macro. The location in core of the PMIDEVTB Csect is pointed to by a V-type address constant in the field SPADEVTB of the System Parameter Area.

The member PMIDEVTB on SYMREL contains a sample Device Table which may be updated or replaced by the system manager to define the installation device types. A user-assigned device type (DEVICE macro, TYPE parameter) is referenced by the STATION macro, IOCODE parameter. The Device Table structure is as follows:

```
PMIDEVTB   CSECT
           DEVICE . . .
           DEVICE . . .
           DEVICE . . .
           DEVICE . . .
           PMISTOP
           END
```

To add a new device type to the table, code the necessary DEVICE macro before the PMISTOP, then reassemble and relinkedit. The Device Table is accessed by all the utilities, and also by internal Intercomm functions, and therefore is required in all regions.

3.4.3 Broadcast Table

The Broadcast Table is core-resident in a Csect named BROADCST and linkedited with the member name PMIBROAD. The table is created and maintained by the user. Each entry in the Broadcast Table represents one broadcast group. The end of this table is indicated by four bytes of hexadecimal 'FF', generated by the PMISTOP macro.

The member PMIBROAD on SYMREL contains a sample Broadcast Group Table which may be updated or replaced by the system manager. The Broadcast Group "TOALL" is used by the optional modules USRSTART and USRCLOSE to send a message to all terminals in the group at startup and closedown time.

The Broadcast Table is defined by the BCGROUP macro. The broadcast group name (five bytes) is followed by a specification of the terminals within the group. A message destined for a broadcast group (MSGHTID in the header) will cause a message to be passed to the Front End for each terminal in the group. Therefore, all terminals in a broadcast group must be of the same device type. The Broadcast Table is accessed by the Output Utility, Message Mapping Utilities, and the Intercomm Front End.

In the following sample Broadcast Table (released as member PMIBROAD), one broadcast group is defined:

```
BROADCAST CSECT
          BCGROUP GROUP=TOALL,TERMS=(CNT01,TEST1)
          PMISTOP
          END
```

An optional routine, BROADRTN, will assist in smoothing the storage requirement peaks when processing broadcast messages. If included, BROADRTN will generate one message at a time with a small time delay before generating the next message. If BROADRTN is used, the module must be in the resident portion of Intercomm, and in the same region as the Output Utility.

3.4.4 Message Mapping Utilities Requirements

The Message Mapping Utilities provide input message editing and output data formatting capabilities to Intercomm subsystems through callable subroutines. MMU allows a unified specification of input and output formatting requirements, and provides simplified format (screen template) generation and data insertion. It can be used instead of the Edit and Output Utilities.

MMU includes all processing options of the Edit and Output Utilities, in addition to control and attribute character insertion. MMU also provides a means of generating symbolic versions of message data areas which can be copied into the application source module for ease of definition and reference.

Tables required by MMU include the Device Table and Station Table and, optionally, the Broadcast Table. General device characteristics may be overridden for an individual terminal via the DVMODIFY macro coded in the Station Table after the PMISTOP. Additional design and implementation considerations for MMU are documented in Message Mapping Utilities.

3.4.5 Edit Utility Requirements

The Edit Control Table (ECT) contains all information necessary to perform editing of a message by the Edit Utility. The Edit Control Table is a variable-length table created and maintained by the user, as described in the Utilities User Guide.

The table resides in core, in a separate Csect labeled VERBTBL. The member PMIVERBS on SYMREL contains required ECT entries for the Intercomm verbs which require Edit Utility processing. User table entries may be added to this member via COPY member USRVERBS, or an entirely new VERBTBL Csect may be created. In either case, care must be taken to ensure that each new entry has been thoroughly tested prior to execution in production mode. Disk-resident table entry references are coded within the core-resident table. Each disk-resident entry is assembled and linkedited individually, for loading to the VRB000 data set via the File Load Utility (PMIEXLD). A DD statement for VRB000 must be included with execution JCL, if disk-resident entries are used.

The Intercomm system manager must define the systemwide field separator character used by the Edit Utility in scanning a message text for field delimiters. This same character is used by the Intercomm Front End to separate the verb from other message text. The SETENV global specification for &SEPCHAR in a BTAM/TCAM Front End must correspond to coding of the SEP parameter of the SPALIST macro to ensure consistent operation.

User-coded edit subroutines may be added, but must be coded in Assembler Language. If used, the system manager must code the SPALIST macro EDITRTN parameter to indicate the highest-numbered edit routine in use. Coding specifications are in the Utilities Users Guide.

In addition to controlling the table specifications for the Edit Utility and ensuring their validity in the production environment, the system manager may control optional edit features via conditional assembly. The globals listed below control conditional assembly of the member PMIEDIT. The globals are defined in the member INTGLOBE and specified in the member SETGLOBE.

Global Definition (INTGLOBE)	Option Defined	Default Specification (SETGLOBE)
&EDERRS &EDERMAX	&EDERRS code specifies that the maximum number of error messages per input verb is limited by &EDERMAX. To suppress this feature, use &EDERRS SETB 1.	SETB 0 SETA 5
&OPTRPT	&OPTRPT code specifies that error messages for non-required fields are not generated. To get error messages use &OPTRPT SETB 1.	SETB 0
&DELCHNG	&DELCHNG code controls the CANCEL/CORRECT feature for keyword input. To activate this feature, use &DELCHNG SETB 0.	SETB 1

The Edit Control Program (PMIEDIT) must be a resident module, but the edit subroutines (Intercomm or user-supplied) may be resident, linkedited as part of an Overlay Region A subsystem group to be resident only when the subsystem which requires their use is loaded, or linkedited within the Intercomm Transient Subroutine Overlay Region. Certain constraints apply in this latter case with respect to situations where one subroutine calls another; all called subroutines must be linkedited in the same load segment as the calling subroutine.

3.4.6 Output Utility Requirements

The Output Utility (PMIOUTPT) is defined by three Subsystem Control Table entries in the member INTSCT. This allows routing of messages to the Output Utility via three subsystem codes and corresponding subsystem queues. Subsystem U is for standard full messages; V is only for segmented messages, and N is for messages to the control terminal.

If segmented messages are processed by the Output Utility, (that is, a series of messages destined for the same terminal, identified by message header VMI=X'51', X'52', X'5C', or X'53' for each segment of the message text) the System Manager must be aware of three parameters on the SPALIST macro controlling message processing:

- DTIMS, which is the delay time between attempts to check the availability of the terminal to assign it to a "segmented message in progress" condition by the PMIDVASN module.
- NTIMS, which is the maximum number of attempts that are to be made to assign a terminal to a "segmented message in progress" condition when a terminal is already busy with other segmented message processing.
- TIMS, which is the time value (multiplied by two minutes) which specifies allowable time between processing of the VMI=X'51' and VMI=X'53' messages; that is, the duration allowed for device assignment to a "segmented message in progress" condition. If a time-out occurs, an error message is routed to the destination terminal indicating SEGMENTED MESSAGE TIMEOUT.

The following globals (defined in INTGLOBE and specified in SETGLOBE) control conditional assembly options of the Output Utility.

Global	Option Defined	Default
&TCAM	Basic TCAM Front End (SETB to 1 to activate this facility)	SETB 0
&DDQBACK	DDQ Automatic Subsystem Input (SETB to 1 to activate this facility)	SETB 0
&BROAD	Broadcast Groups in use (SETB to 1 to suppress this facility)	SETB 0
&RPTBLE	Batch Report Table Facility (SETB to 1 to suppress this facility)	SETB 0
&ALTRPT	Alternate Format Table Facility in use (SETB to 1 to suppress this facility)	SETB 0
&OUTEXIT	User Output Exit USROTEDT not used (SETB to 0 to activate this facility)	SETB 1

3.4.6.1 Adding Output Format Table Entries

User-generated Output Format Table (OFT) entries may be added to the Intercomm system as either core-resident or disk-resident. Each user entry is identified by the name RPTOnnnn, where nnnn is in the range 0051 to 9999. Numbers 1-50 are reserved for Intercomm use. Individual table entries (REPORTs) must be assembled and linkedited separately. These table entries must not use the Csect name PMIRCNTB nor include a PMISTOP macro. Generation of OFTs is described in the Utilities Users Guide.

Two members are contained on SYMREL to facilitate linkedit of OFT entries for the core-resident table: (1) PMIRCNTB--Table Heading (Csect name PMIRCNTB); and (2) PMIRCEND--Table End (PMISTOP macro). In an Intercomm linkedit generated by the ICOMLINK macro, these members bracket the common system OFT entries RPT00043 and RPT00045 which should be resident. Other Intercomm OFT entries may be made resident, if desired. See also installation of system command verbs requiring REPORTs, as described in System Control Commands.

The following linkedit control statements are used to construct the core-resident OFT (entries do not have to be in numeric sequence):

```
INCLUDE SYSLIB(PMIRCNTB)          BEFORE ALL RESIDENT REPORTS
INCLUDE SYSLIB(RPT00043)
INCLUDE SYSLIB(RPT00045)
INCLUDE SYSLIB(RPT000nn)
   .
   .
INCLUDE SYSLIB(RPT0nnnn)
INCLUDE SYSLIB(PMIRCEND)          AFTER ALL RESIDENT REPORTS
```

Disk-resident OFT entries have no entry in the core-resident table. They are loaded to the BDAM data set RCT000 via the File Load Utility (PMIEXLD) for access at execution time. A DD statement for RCT000 must be present in the Intercomm execution JCL. Many Intercomm error and statistical messages are produced via OFT numbers 1-50 released as member names RPT00001 to RPT00050 on SYMREL. These table entries are loaded to RCT000 at system installation time. The block size of RCT000 must be a minimum of 1500 to accommodate Intercomm OFTs.

3.4.6.2 Error Messages from the Output Utility

Error messages reflecting problems encountered during message processing by the Output Utility are generated and queued for subsequent processing via the Output Utility. The messages are formatted according to OFT entries which may be disk-resident. Each error message is prefixed with identifying information:

SEQ NO	(Monitor Message Number of message in error)
SSC	(Sending Subsystem Code)
RSC	(Receiving Subsystem Code: U, N or V)
TID	(Destination Terminal of message in error)

Each error message explicitly defines the reason for rejecting the message being processed, for example:

THE FROM IS GREATER THAN THE TO FIELD.

REPORT NUMBER NOT IN MESSAGE.

RCTnnnn IS INVALID. NOT FOUND. (OFT entry missing for nnnn)

See Messages and Codes for a precise listing of Output Utility error messages.

3.4.6.3 Output User Exit--USROTEDT

An optional user-coded exit, USROTEDT, is available in PMIOUTPT. Before logging a message and sending it to the Front End, the Output Utility issues a conditional call (CALLIF) to USROTEDT, if such a routine has been written and included. USROTEDT is also called by FESEND if the subsystem calls FESEND (FESENDC) directly. In a Multiregion environment, if PMIOUTPT is included in a satellite region, USROTEDT should be included only in the control region (called by FESEND). This will prevent it from being called twice. Standard linkage conventions are to be used.

The parameter list passed to USROTEDT via register 1 contains:

1. Address of message
2. Address of System Parameter Area
3. Address of a fullword in which the user-written routine must place a return code.

Any return code other than 0 will cause PMIOUTPT or FESEND to stop the message from being queued for the Front End. If the user wishes to create an entirely new message area, an area of storage may be obtained (via the STORAGE macro) and a new message may be created consisting of header and text. Do not free the storage area occupied by the old message. Change the address of the message in the parameter list to reflect the address of the new message.

To generate the code to call USROTEDT, make sure the global &OUTEXIT was not set to 1 in SETGLOBE when FESEND and PMIOUTPT were assembled for Intercomm installation.

3.4.6.4 Output User Exit--USROUTCK

USROUTCK is a user-coded user exit conditionally called (via CALLIF) by PMIOUTPT. Its purpose is to allow the user to determine if PMIOUTPT is to process the unformatted message, based on installation-dependent criteria. If the message is to be cancelled, USROUTCK must free it before returning to PMIOUTPT. In this case, the user exit is responsible for notifying the terminal that the message was cancelled, if a response is expected.

At entry to USROUTCK, register 8 points to the input message (header). If PMIOUTPT is not to process the message, a nonzero return code must be returned by USROUTCK to PMIOUTPT in register 15; otherwise, a zero return code is required, indicating PMIOUTPT is to process and/or forward the message to FESEND. If the message is cancelled, PMIOUTPT returns immediately to the Subsystem Controller with a zero return code. Standard linkage conventions are to be used.

3.4.7 Change/Display Utility Requirements

The Subsystem Control Table entry for the Change/Display Utility is provided in the released member INTSCT. The SCT defines the CHANGE module as a resident subsystem. The user may redefine the Change/Display entry as a dynamically loaded subsystem. Other modules referenced by CHANGE include DISPLAY, FORMAT, CRUNCH and PTRNTBLE. The UTILITY parameter of the ICOMLINK macro is used to generate the include statements.

All file (format) description records (FDRs) for the Change/Display Utility are disk-resident (ddname DES000) table entries loaded via the File Load Utility (PMIEXLD). See the Utilities Users Guide for coding specifications, a description of application subsystem interface to the CHANGE utility, and the required user-coded CHNGTB table. The DD statement for DES000 must be specified in the Intercomm execution JCL if Change/Display is used. The released PMIVERBS contains required ECT entries for the CHNG and DSPL verbs for this utility.

User files accessed via the utility are defined via the GENFTBLE macro in the Intercomm File Table (PMIFILET). Additional considerations are:

- There must be an entry in the File Table for each Intercomm disk-resident table data set (RCT000, VRB000, DES000, etc.) as well as files accessed via Change/Display.
- The entry in the File Table defines the block size for data set access which must be greater than or equal to the physical block size of the user file data block on disk. If the optional module PMICKFTB is included, these block sizes are verified at startup and dynamically corrected if required.
- The last entry must be followed by a PMISTOP macro.

Following is a sample PMIFILET:

```
PMIFILET  CSECT
          ENTRY    PMIFILTB
PMIFILTB  EQU      *
          GENFTBLE FNAME=RCT000,BLKSIZE=1500,TYPE=BDAM
          GENFTBLE FNAME=DES000,BLKSIZE=750,TYPE=BDAM
          GENFTBLE FNAME=VRB000,BLKSIZE=750,TYPE=BDAM
* BLKSIZE FOR DES000,RCT000,VRB000 CORRESPOND TO INTERCOMM RELEASE
* SPECIFICATIONS.  USER MUST CHANGE FOR LARGER TABLE ENTRIES.
* ADD USER FILE DESCRIPTIONS HERE.
*
          GENFTBLE FNAME=USERFILE,BLKSIZE=xxxx,TYPE=ISAM,DESNUM=7
          PMISTOP
          END
```

3.5 MESSAGE PROCESSING FACILITIES

The following subsections describe other Intercomm facilities for queuing and processing messages.

3.5.1 Message Switching

The standard terminal-requested message switching facility is activated by the SWCH system control command which uses a subsystem for the switching and allows messages to be switched to one or more receiving terminals, as well as to Broadcast Groups.

The Intercomm Front End also provides a Fast Message Switch facility, as it recognizes input messages which contain, in place of the normal verb, the five-character name of the single terminal to which the message should be forwarded. For example, terminal NYC01 sends a message to terminal BOS07 in the following format:

```
BOS07,THIS IS A SWITCHED MESSAGE
```

The message would be routed, completely within the Front End, to terminal BOS07. The receiving terminal name is replaced by the sending terminal name so that the origin of the message is known. The message sent to BOS07 would be:

```
NYC01,THIS IS A SWITCHED MESSAGE
```

As with the standard message switching facility, no reformatting of the message is done. Messages should therefore be switched only to terminals which have hardware characteristics compatible with the sending terminal. For example, a multiline message from a terminal which uses NL (new line) characters should not be switched to a terminal which requires CR/LF (carriage return, line feed) characters.

If the receiving terminal is not active, or is not currently able to receive an output message, the message remains queued until it can be transmitted.

3.5.2 Multimessage Queuing via the Dynamic Data Queuing Facility

The Front End Data Queuing feature operates in conjunction with the Intercomm Dynamic Data Queuing Facility. It enables an application to send to the Front End a dynamic data queue (DDQ) that contains messages to be transmitted to a terminal. Thus, instead of sending one message at a time and having each message queued for Front End transmission, and then dequeued by the Front End, an entire group of messages may be placed on a DDQ and treated as one message.

For implementation of the data queuing feature, refer to the applicable application programmer guides and the Dynamic Data Queuing Facility for further details. In addition to the Dynamic Data Queuing Facility, the Front End Control Message Facility (see below) must be installed in order to use the Front End Data Queuing feature.

The Dynamic Data Queuing Facility is also used for easy, orderly retrieval of segmented input messages, and may be used for queuing of output messages to the Change/Display or Output Utilities.

3.5.3 Front End Control Message Facility

This facility allows application subsystems to generate and transmit control messages to the Front End. Three types are currently defined. A control message (FECM) may be either a feedback-request, a release-request, or a DDQ-identifier for a group of messages collected on a DDQ. For implementation, the module FECMMOD must be included in the Intercomm linkedit.

Feedback-requests, when sent to a terminal, cause the Front End to send a message, containing user-specified text, to a user-specified subsystem. This message, which is sent when all messages in front of the feedback-request message have been transmitted to the terminal, can be used, for example, to determine when a report has actually been printed. The feedback facility also allows synchronization of message transmission with subsystem processing. A subsystem may issue a feedback FECM which signals the Front End to notify the issuing subsystem or another subsystem when a certain output message has been transmitted to a destination terminal. RLSE system control commands may be intermingled with multiple screens being forwarded to a CRT to force a previous screen to be overlaid.

DDQ-identifier control messages designate a DDQ containing messages to be sent to the terminal. These messages, which must be preformatted (VMI=X'67' or X'57'), are read from the DDQ and sent to the terminal. The DDQ, subject to user specification, may be either freed or retained. By retaining the DDQ, the messages may be broadcast; therefore it is a convenient facility to send canned reports or other data. The DDQ may also contain FECMs for other DDQs, or for feedback, mixed in with real output messages (only at the end of the DDQ, if VTAM). DDQ FECMs require dedicated queues for the receiving terminals.

Release-requests, when sent to a terminal, override normal CRT processing logic, which requires a one-for-one correspondence between input and output messages. When the release FECM is processed by the Front End, it causes the next message queued for the CRT terminal to be transmitted immediately, rather than waiting for input from the operator. Under a VTAM Front End, certain protocols (HDFF) may preclude immediate transmission of the next message; see SNA Terminal Support Guide.

3.5.4 Page Facility

The Page Facility provides a browsing capability for CRT output messages that have been collected on a disk data set, rather than being queued for the terminal. A subsystem may request MMU to pass messages to the Page Facility which were formatted by MAPOUT processing, or the subsystem may call the Page Facility directly with messages to be formatted later by the Change/Display Utility and/or the Output Utility.

The first message of the series is always returned directly to the terminal. The terminal operator subsequently uses Page Facility commands to browse and ultimately save or discard the collected messages. Further details are described in Page Facility.

3.5.5 Intermediate Message Data Storage

Two facilities are provided for storage of data by a message processing thread between input messages when an interactive conversation is in progress. These are the Store/Fetch Facility (see the manual of that name), and the CONVERSE facility described in the applicable Programmers Guide. The former provides for storage and retrieval of saved data as data strings in core or on disk. The saved data may consist of tables, counters, message data, or file data, as the strings may be of any length. The CONVERSE facility is used to save and restore the dynamic working storage of reentrant higher-level language subsystems between input messages, that is, while waiting for a response to the last output message. Installation and programming considerations for these facilities are described in the referenced manuals.

3.6 THE SYSTEM PARAMETER AREA (SPA)

The System Parameter Area consists of systemwide variables and system component addresses controlling all message processing functions. These elements are defined in the member INTSPA which contains the following:

- SPA CSECT--the System Parameter List, defined by the SPALIST macro.
- USERSPA: This is an optional user extension to the System Parameter List, with user-defined variables and addresses, coded as a separate source module in SYMUSR.
- SPAEXT: This is the Intercomm extension to the System Parameter List. SPAEXT Csect is also generated by the SPALIST macro, using the EXTONLY=BOTH parameter.

Figure 3-4 illustrates typical JCL which may be used to create INTSPA, or the released member on SYMREL may be modified to user requirements and placed on SYMLIB or SYMUSR.

3.6.1 System Parameter List (SPA Csect)

The System Parameter List is a fixed area of 500 bytes in length. It contains addresses, control information and statistics for the entire Intercomm system. When building the SPA Csect, the System Parameter List is generated by coding the SPALIST macro.

3.6.2 User Extension to the System Parameter List (USERSPA)

The variable-length USERSPA allows definition of user fields or table areas common to all user subsystems. Since all subsystems are passed the address of the SPA as an entry parameter, application subsystems may not alter values within the System Parameter Area. Users must instead add user fields to the SPA Csect via USERSPA. User additions to the System Parameter Area are coded as a separate source module named USERSPA. When the SPALIST macro is assembled, the source module USERSPA will automatically be copied into the System Parameter Area, at a displacement of 500 bytes from the beginning of the SPA (plus X'1F4'), and labeled SPAUSER. The maximum length allowed for USERSPA is 4095 minus 500, or 3595 bytes.

USERSPA should be correctly referenced by application subsystems. For application programmers' use in defining this user extension, source statement library members should be provided in the appropriate language available for copying into the program.

3.6.3 Intercomm Extension to the System Parameter List (SPAEXT Csect)

The SPAEXT Csect is variable in length to allow for continued flexibility in adding systemwide control variables to the System Parameter List.

3.6.4 Separate Assembly of the SPA and the SPAEXT Csects

The number of VCONs required by the addition of USERSPA and/or edit routines may necessitate separate assembly of the SPA Csect and the Intercomm extension to the System Parameter List. The SPALIST macro must be assembled twice, once to generate the SPA Csect and once to generate the SPAEXT Csect. With the exception of the EXTONLY=YES parameter, denoting generation of the SPAEXT, coding of the SPALIST macro parameters must, in both cases, be identical. Currently, approximately 300 VCONs are generated by the combined SPA and SPAEXT Csects, along with VCONs for the Edit Utility routines EDIT0001-0020.

3.7 THE SUBSYSTEM CONTROL TABLE (SCT)

Each subsystem is defined to Intercomm by an entry in the Subsystem Control Table, generated via the SYCTTBL macro coded in the member INTSCT which contains the following:

- SCT Csect containing:
  - -- The Subsystem Control Table (SCT)--individual table entries defining subsystem characteristics and message processing scheduling parameters, defined via the SYCTTBL macro.
  - -- The Subsystem Control Table Overlay and Binary Search Indices, generated via the GENINDEX macro.
  - -- The SCT Extension--automatically generated SYCTTBL extensions for defining dynamically loadable subsystems.
- DYNREQ1 Csect--automatically generated SCT addition for reentrant COBOL subsystems.

Figure 3-3 illustrates the relationship of the SPA, the SCT, and the Overlay Index.

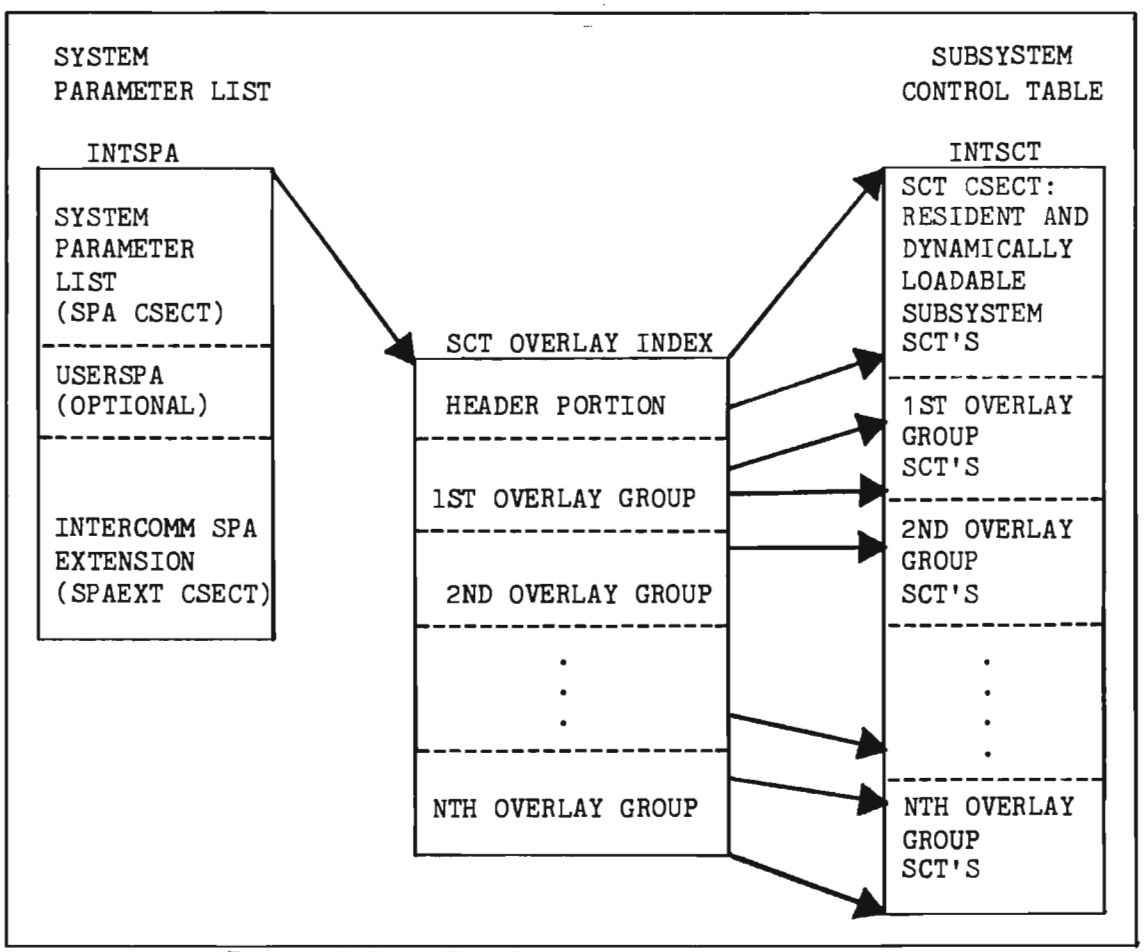


Figure 3-3. The System Control Components

The SYCTTBL macro defines the following for each subsystem:

- Subsystem residency (overlay region, VS execution group, dynamically loadable, or resident)
- Subsystem characteristics (subsystem code, program language, reentrancy, entry point name, subpool requirements, etc.)
- Processing specifications (immediate or threshold, queues, queue overflow, priority, concurrent message processing limits, scheduling, etc.)
- Control parameters (time-out limit, snaps/WTOs required, cancellation criteria, security, restart, etc.)

If more than 1000 SYCTTBL macros are defined in INTSCT, the global values (released as 1000) in FEMACGBL must be reset to the higher number desired to allow sorting of the greater number of subsystems for the binary search index. Additionally, use of Assembler H and/or a larger region size may be required for the assembly of INTSCT.

Figure 3-5 illustrates the released member INTSCT on SYMREL which provides for most of the Intercomm subsystems and indicates where user SCT entries may be inserted via a user-coded copy member USRSCTS. If an overlay structure is not used, the order of SCT entries is immaterial as the Binary Search Index is used by Intercomm to find a particular entry. Figure 3-4 shows JCL to create a USRSCTS and assemble and link the released version of INTSCT which copies USRSCTS.

```
//SPA     EXEC      LIBELINK,Q=USR,NAME=INTSPA,LMOD=INTSPA
./ ADD    NAME=INTSPA
*                   SYSTEM PARAMETER LIST
SPA       CSECT
          SPALIST   A=A,EXTONLY=BOTH,CCNID=CNT01,WTO=NO,                X
                    .
                    other operands as desired
                    .
          END
//*
//SCT     EXEC      LIBELINK,Q=USR,NAME=INTSCT,LMOD=INTSCT
//LIB.SYSIN  DD     *
./ ADD    NAME=USRSCTS
*                   USER SUBSYSTEM CONTROL TABLE ENTRIES
          SYCTTBL
              .
              .
              .
          SYCTTBL
//ASM.SYSIN  DD     DSN=INT.SYMREL(INTSCT),DISP=SHR
//
```

Figure 3-4. Creating the System Parameter Area and SCT

```
SCT      CSECT
         DC    CL8'SCTENTRY'       SCTS BEGIN HERE.
*************************************************************************
*            SCT DEFINITIONS (SYCTTBL'S) FOR INTERCOMM S/S              *
*************************************************************************
U        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=U,LANG=RBAL,TCTV=120,   X
               MNCL=4,DFLN=PMIQUE,BLRI=F,PCEN=10,NUMCL=10,            X
               SBSP=PMIOUTPT,RESTART=NO
V        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=V,LANG=RBAL,            X
               TCTV=120,MNCL=4,NUMCL=10,SBSP=PMIOUTPT,RESTART=NO
N        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=N,LANG=RBAL,TCTV=120,   X
               MNCL=4,DFLN=PMIQUE,BLRI=F,PCEN=10,NUMCL=10,            X
               SBSP=PMIOUTPT,RESTART=NO
J        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=J,LANG=RBAL,TCTV=0,     X
               MNCL=1,NUMCL=2,SBSP=PMICLDWN,PRTY=3,RESTART=NO
LL       SYCTTBL ECB=YES,WTO=NO,SUBH=L,SUBC=L,LANG=RBAL,TCTV=120,     X
               MNCL=4,NUMCL=10,SBSP=LOADSCT,RESTART=NO
MM       SYCTTBL ECB=YES,WTO=NO,SUBH=M,SUBC=M,LANG=RBAL,TCTV=120,     X
               MNCL=4,NUMCL=10,SBSP=MMUCOMM,RESTART=NO
GP       SYCTTBL ECB=YES,WTO=NO,SUBH=G,SUBC=P,LANG=RBAL,TCTV=120,     X
               MNCL=4,NUMCL=10,SBSP=GPSS,LOG=NO,RESTART=NO
B        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=B,LANG=RBAL,TCTV=120,   X
               MNCL=2,NUMCL=2,SBSP=SWITCH,LOG=NO,RESTART=NO
P        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=P,LANG=RBAL,TCTV=120,   X
               MNCL=5,NUMCL=5,DFLN=PMIQUE,BLRI=F,PCEN=5,              X
               SBSP=PAGEMSG,RESTART=NO
W        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=W,LANG=RBAL,            X
               TCTV=120,MNCL=4,NUMCL=10,SBSP=SENDBACK,RESTART=NO
R        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=R,LANG=RBAL,            X
               TCTV=120,MNCL=4,NUMCL=10,SBSP=IXFRPTIQ,RESTART=NO
H        SYCTTBL ECB=YES,WTO=NO,SUBH=000,SUBC=H,LANG=RBAL,            X
               TCTV=120,MNCL=4,DFLN=PMIQUE,BLRI=F,PCEN=10,
               NUMCL=4,SBSP=CHANGE,RESTART=NO
HH       SYCTTBL ECB=YES,WTO=NO,SUBH=H,SUBC=H,LANG=RBAL,TCTV=120,     X
               MNCL=1,DFLN=PMIQUE,BLRI=F,PCEN=10,NUMCL=4,             X
               SBSP=CHANGE,RESTART=NO
CC       SYCTTBL ECB=YES,WTO=NO,SUBH=C,SUBC=C,LANG=RBAL,TCTV=120,     X
               MNCL=4,NUMCL=10,SBSP=COPYSS,RESTART=NO
*************************************************************************
*          SCT DEFINITIONS (SYCTTBL'S) FOR USER SUB/SYSTEMS             *
*************************************************************************
         COPY USRSCTS
         GENINDEX
         PCENSCT
         END
```

Figure 3-5. INTSCT Coding of Intercomm Subsytems

Optionally, control of maximum thread concurrency for a group of subsystems may be implemented by coding a RESOURCE macro prior to all the SYCTTBL macros. The RESOURCE macro is used to provide a systemwide limit on the number of threads that may concurrently access a specific system resource, or is often used to control concurrent access to a data base. It is referenced via the SYCTTBL macro RESOURC parameter.

3.7.1 Coding Subsystem Control Table (SCT) Entries

The SCT defines all subsystems executing under Intercomm. The table entries coded via the SYCTTBL macro must be in the following sequence:

1. Resident and dynamically loadable subsystem entries
2. Entries for subsystems in each Overlay Region A overlay segment (OVLY parameter) or in each VS execution group (EXGRP parameter) if used.

The OVLY parameter defines the subsystem's residency, and is coded according to the following conventions:

- OVLY=0--indicates a resident, or VS execution group, or dynamically loadable subsystem. Default.
- OVLY=1--indicates an Overlay Region B subsystem, to be scheduled by MONOVLY (see Section 3.9.6).
- OVLY=2--indicates an Overlay Region C subsystem, to be scheduled by MONOVLY.
- OVLY=3--indicates an Overlay Region D subsystem, to be scheduled by MONOVLY.
- OVLY=4--indicates a subsystem within an Overlay Region A subsystem group. It must be coded in ascending consecutive order: the first number must be 4; the highest permissable number is 62.

  .
  .
  OVLY=62

Figure 3-6 illustrates a sample coding of SCTs, with resident and Overlay A Intercomm-provided subsystems. More than one subsystem may belong to the same Overlay A group. Each group is delimited by a required label: SCTLRES--for resident (dynamically loadable) subsystems; SCTLOVn--for Overlay A subsystem groups, where n is in the range of 1 to 59 (corresponding to OVLY numbers 4-62).

```
SCT      CSECT
*                                  DSECT DESCRIPTION
         COPY  SCTLISTC
SCT      CSECT
         DC    C'SCTENTRY'         SCTS BEGIN HERE.
SCTRES   DS    OF
B        SYCTTBL SUBC=B,SBSP=SWITCH,OVLY=0,NUMCL=4,LANG=RBAL,MNCL=2
W        SYCTTBL SUBC=W,SBSP=SENDBACK,OVLY=0,NUMCL=4,LANG=RBAL
SCTLRES  EQU   *
*                        O V E R L A Y   A   G R O U P   O N E
H        SYCTTBL SUBC=H,SBSP=CHANGE,OVLY=4,NUMCL=4,LANG=RBAL,MNCL=4, X
               DFLN=PMIQUE,PCEN=10,BLRI=F
HH       SYCTTBL SUBH=H,SUBC=H,SBSP=CHANGE,NUMCL=4,OVLY=4,           X
               LANG=RBAL,MNCL=1,DFLN=PMIQUE,PCEN=10,BLRI=F
SCTLOV1  EQU   *              END OF OVERLAY ONE
*                        O V E R L A Y   A   G R O U P   T W O
U        SYCTTBL SUBC=U,SBSP=PMIOUTPT,OVLY=5,NUMCL=10,               X
               LANG=RBAL,MNCL=4,DFLN=PMIQUE,PCEN=10,BLRI=F
V        SYCTTBL SUBC=V,SBSP=PMIOUTPT,OVLY=5,NUMCL=4,                X
               LANG=RBAL,MNCL=1,DFLN=PMIQUE,PCEN=10,BLRI=F
N        SYCTTBL SUBC=N,SBSP=PMIOUTPT,OVLY=5,NUMCL=4,                X
               LANG=RBAL,MNCL=4,DFLN=PMIQUE,PCEN=10,BLRI=F
SCTLOV2  EQU   *              END OF OVERLAY TWO
*                        O V E R L A Y   A   G R O U P   T H R E E
J        SYCTTBL SUBC=J,SBSP=PMICLDWN,OVLY=6,RESTART=NO,NUMCL=2,     X
               LANG=RBAL,PRTY=3,MNCL=1
SCTLOV3  EQU   *              END OF OVERLAY THREE
*                        O V E R L A Y   A   G R O U P   F O U R
LL       SYCTTBL SUBC=L,SUBH=L,SBSP=LOADSCT,NUMCL=4,OVLY=7,LANG=RBAL
SCTLOV4  EQU   *
*                        O V E R L A Y   A   G R O U P   F I V E
CC       SYCTTBL SUBC=C,SUBH=C,SBSP=COPYSS,NUMCL=4,OVLY=8,LANG=RBAL, X
               RESTART=NO
SCTLOV5  EQU   *
*                        O V E R L A Y   A   G R O U P   S I X
GP       SYCTTBL SUBH=G,SUBC=P,SBSP=GPSS,LANG=RBAL,OVLY=9,NUMCL=4,   X
               RESTART=NO
SCTLOV6  EQU   *
         GENINDEX
         PCENSCT
         END
```

Figure 3-6. Sample Coding of INTSCT with an Overlay Structure

For VS users wishing to code VS execution groups, instead of Overlay Region A subsystem groups, the OVLY parameter is coded as 0, and the EXGRP parameter is used as follows:

EXGRP=4--indicates a resident subsystem within a VS execution group. It must be coded in ascending consecutive order: the first number must be 4; the highest possible number is 62.

:

EXGRP=62

NOTE: If more than one subsystem code is used for the same subsystem (accessed by multiple verbs), then the OVLY or EXGRP parameter value must be the same on each SYCTTBL pointing to that subsystem. Also, subsystem residency must be the same (either resident, or overlay, or dynamically loadable).

VS execution group scheduling is similar to Overlay Region A scheduling except that, instead of the overlay supervisor, the VS paging supervisor is used to invoke loading of the subsystem logic into main storage. See also Chapter 7 on VS installation and page preloading.

Figure 3-7 shows a listing of Intercomm-supplied subsystems and reserved subsystem codes. If no specific value is listed for SSCH, then it must be binary zeros (000--default). Additional subsystems for special feature commands are described in System Control Commands.

SSCH	SSC	Function (Member Name)
	J	Closedown (CLOSDWN3: entry PMICLDWN)
	H	Change/Display Utility (CHANGE)
H	H	Single-thread Display--segmented messages (CHANGE)
	B	Message switching between terminals (SWITCH)
	W	Message echoing (SENDBACK)
L	L	Loading dynamically linkedited modules (LOADSCT)
C	C	Copy processing for BTAM 3270 terminals (COPYSS)
G	P	General Purpose Subsystem (GPSS)
	P	Page Facility (PAGEMSG)
	Q	Checkpointing (CHCKPTSS)
	S	Basic Security processing (PMISIGN)
	T	Fine Tuner processing (FINTUNER)
	U,N,V	Output Utility (PMIOUTPT)
	M	Internal processing (Time Zone, etc.)
	Z	MROTPUT--satellite regions only under MRS
	K	Multiregion commands--control region only (MRCONSS)
	E	Dummy subsystem for ESS processing ($$$$SECU)
D	E	Data Entry Facility (INTBETAI)
A	G	Autogen (ISGEN)
	R	File Handler Statistics (IXFRPT01: entry IXFRPTIQ)
	F	VS page fix/unfix (PMIFIXB)
M	M	MMU command processing (MMUCOMM)

Figure 3-7. Intercomm-Supplied Subsystems

3.7.2 Coding Subsystem Control Table Indices (GENINDEX)

The SCT Indices consist of two elements: the SCT Overlay Index--used for scheduling work for resident and dynamically loadable subsystems, and for overlay or execution groups within the Subsystem Control Table; and the SCT Binary Search Index--used for finding an entry in the Subsystem Control Table. Each Overlay Index entry is three words in length. There is one entry for resident and dynamically loadable subsystem SCTs (OVLY=0), followed by one entry for each overlay group.

As illustrated in Figure 3-3, the System Parameter Area points to the SCT Overlay Index, which in turn is used to locate the individual SCT groups.

As illustrated in Figure 3-5, the SCT Indices are generated at assembly time by coding the GENINDEX macro after all the SYCTTBL entries. However, if multiple overlay group indices for the same Overlay A group are desired, or if no resident or dynamic load SCTs are defined, the SCT Overlay Index must be hand-coded, as described in Appendix C. In this case, the GENINDEX macro must be coded with the parameter OVLYNDX=NO, and is placed after the user-coded Overlay Index.

3.7.3 Coding Overflow Disk Queue Allocations (PCENSCT)

As illustrated in Figure 3-5, the PCENSCT macro is coded prior to the GENINDEX macro. This macro has no parameter and is coded only once. Its function is described in Section 3.8.1.

3.7.4 Adding a Subsystem

In addition to coding the SYCTTBL for a new subsystem, the entire Subsystem Control Table structure may have to be reevaluated to determine the impact of the new subsystem on response time, throughput, and queue space for all subsystems. Also, other table entries may be required in order to test the new subsystem or utilize it in the production environment.

The Front End Verb Table must be updated with the new verb(s) for the added subsystem. Locking, conversational, and other Front End processing parameters may have to be considered, depending on the terminal type(s) being used. Other Intercomm facilities, such as ICOMPOOLs, may be affected, and table or disk-resident entries for the Intercomm utilities may be required.

3.7.5 Subsystem Control Table Verification (CKOVLYNO)

If included in the Intercomm linkedit, the routine CKOVLYNO will be executed at system startup to verify the coding of the SCT Index and individual SCTs. In particular, it checks to ensure that each group of SCTs indicated by an SCT Overlay Index entry does in fact have the SYCTTBL parameter OVLY coded correctly, and that the OVLY group numbers are in ascending and consecutive order. If any table entries are incorrect, startup execution terminates with a user abend code 98.

3.8 SUBSYSTEM PROCESSING SPECIFICATIONS

Subsystem response time and throughput are affected not only by subsystem residency, but also by queue, scheduling and processing limit specifications. These specifications are also defined via SYCTTBL macro parameters for each subsystem.

3.8.1 Subsystem Queue Specifications

A subsystem queue is a list of messages awaiting processing by the subsystem. These messages may be incoming transactions (from a terminal), or passed from another subsystem. These queues are also known as input queues, in contrast to output terminal queues of messages awaiting transmission. Three types of queues may be defined: core queues, high-priority core queues, and disk overflow queues.

At least one type of queue should be defined. The queuing method is controlled by the BLRI parameter. Normally, a priority queue is defined only if more than one verb is processed by the subsystem, and certain verbs (such as those requiring little subsystem processing) should be processed as soon as possible. A subsystem which is not response time dependent or which is activated only periodically would have little use for a core queue because a core queue ties up system resources for holding the message in core. Disk queues are used for overflow from the core queue(s) at high activity periods, or to hold messages when no core queues are defined. The SYCTTBL AUXS parameter is coded when no core or disk queues are defined.

The NUMCL parameter defines the number of elements in a core queue and creates an entry in the internally generated PMICLZZZ Csect which defines the core list (queues) for all subsystems operating under Intercomm. The purpose of the core list is to contain the addresses of all messages that are destined for a subsystem and are still in core. When the core list is full, messages are written to overflow disk queues that are accessed under the file name (JCL DD statement label) specified by the SYCTTBL macro, DFLN parameter.

In addition to the normal core queue, a priority core queue may be defined (by the PRYMSGS parameter of the SYCTTBL macro) for those messages requiring priority processing for fast response time. If the priority queue is full when adding a priority message to a subsystem's queue, it will be added to the end of the normal queue (core or disk). A priority message is recognized by Intercomm when a C'P' is in the message header field MSGHUSR. The P is inserted during Front End verb processing if the BTVERB parameter HPRTY=YES was coded, or if a subsystem initializes MSGHUSR before queueing a message for another subsystem.

The disk queues are contained on BDAM data sets which must be preformatted with dummy records via the Intercomm utility CREATEGF (see Chapter 12). If a disk queue data set is to be shared among several subsystems (PCEN parameter in SYCTTBL), assignment of space is allocated at system startup time by the module CALCRBN, which calculates the appropriate percentage of the actual number of blocks (RBNs) on the data set and rounds that down to the nearest multiple of 8; a minimum of eight RBNs are allocated. If the data set referenced by DFLN is exhausted, an indicative message is issued and startup abends with a user code 44. Queue and block size considerations include message lengths and traffic for a given subsystem, as well as achievement of minimal I/O activity, since messages with lengths greater than disk queue block size are spanned. A maximum of 50 different disk queue data sets may be defined for the combined subsystems in the Subsystem Control Table. The PCENSCT macro, coded after the GENINDEX macro, will print the accumulated percentages per disk queue data set as part of the assembly of the SCTs; the output should be checked whenever a SYCTTBL is added. Typical output generated by the PCENSCT macro is illustrated below.

```
*** ACCUMULATED PERCENTAGES PER DISK QUEUE ***

***  QUEUE NAME          PERCENTAGE ***
*,     QUEUEN               40.0
*,     QUEUEA              100.0
*,     QUEUEC              100.0
*,     QUEUEU              100.0
*,     QUEUEH               80.0
```

3.8.2 Scheduling and Concurrent Processing Limits

SYCTTBL scheduling parameters are SCHED, ECB, and THRSH. Processing limits are defined by the MNCL and RESOURC parameters, which are also directly related to the residency and reentrancy of the subsystem.

3.9 SUBSYSTEM RESIDENCY CONSIDERATIONS

The subsystem identifier, or receiving codes in the Intercomm message header (MSGHRSCH and MSGHRSC fields), is coded for the subsystem in the SUBH and SUBC parameters of the SYCTTBL macro. Each SYCTTBL must have a unique set of codes which are used by the Intercomm subsystem queuing routines to identify the specific subsystem to process a transaction. Once found, the transaction is queued for later dispatch of the subsystem. Dispatch considerations are based not only on systemwide parameters defined for the SPALIST macro, but also on subsystem residency, reentrancy and processing specifications.

3.9.1 Subsystem Reentrancy

Reentrancy is defined to Intercomm by the LANG parameter of the SYCTTBL macro. See the applicable Programmers Guide for criteria for reentrant subsystems under Intercomm which may process more than one transaction (message) at a time (more than one thread dispatched), if permitted by scheduling parameters. High-level language subsystems coded and defined to Intercomm as reentrant may not, however, be linkedited as reentrant.

3.9.2 Resident Subsystems

Definition of a subsystem as resident, dynamically loadable, in an Overlay Region A, or in a VS execution group, is a function of reentrancy, message traffic, message volume and storage requirements. For efficiency, those reentrant subsystems with high volume and/or traffic should be made resident. Subsystems with sporadic or single periods of volume processing could be made dynamically loadable, while those with lower volume but more constant traffic could be defined for an overlay or execution group.

In this discussion, volume represents the possible total number of transactions to be processed during an execution of Intercomm, while traffic represents the number to be processed within a specific time span. Storage requirements for processing of a transaction include not only the program area, but also the dynamic working storage (pool areas).

Subsystem residency is also affected by the processing time required, file and data base access, message formatting, etc., and response time criteria.

Because loading delays are avoided, resident subsystems potentially provide the best response time. They are defined to Intercomm in the OVLY=0 group, as described above. Throughput is controlled by scheduling parameters and also depends on external storage requirements and processing time. Resident subsystems are linkedited with resident Intercomm modules.

3.9.3 Overlay A and Execution Group Subsystems

Depending on scheduling and concurrent processing limits defined for each subsystem within the overlay structure, Intercomm controls the Overlay A processing. An overlay group may consist of one or more subsystems which may be grouped according to reentrancy, programming language, processing time, resource requirements, traffic, volume, etc. Scheduling and concurrent processing limits are relevant, as, once work is dispatched for the group in Overlay A, another group cannot be overlaid into the area until all the dispatched threads have completed processing.

Intercomm controls VS execution group processing, depending on scheduling and concurrent processing limits defined for each subsystem within the VS execution group. An execution group may consist of one or more subsystems which are grouped according to reentrancy, programming language, processing time, resource requirements, traffic, volume, etc. Scheduling and concurrent processing limits are relevant, since once work is dispatched for one execution group no other execution groups will be scheduled until the current group completes its processing. This technique is useful in preventing excessive VS paging overhead when real storage is at a premium; all nonzero EXGRP subsystems are linked as resident in a contiguous group.

Those subsystems which are to be executed from Overlay Region A must be linkedited according to the same structure depicted in Figure 3-6. In other words, all subsystems whose SYCTTBL macro OVLY parameter is coded as 4 must be inserted in the same overlay segment, all OVLY=5 in the same segment, etc. These SYCTTBLs must have OVLY coded in ascending, sequential order.

The following example illustrates a sample Subsystem Control Table with two Overlay A groups defined. Linkedit control cards which relate the OVLY parameter definitions to Overlay A INSERT statements are illustrated in Figure 3-8.

```
* RESIDENT and DYNAMICALLY LOADABLE SUBSYSTEMS
      SYCTTBL -----
      SYCTTBL -----
* OVERLAY A GROUP 1
      SYCTTBL SBSP=SUBSYSA,OVLY=4,---
      SYCTTBL SBSP=SUBSYSB,OVLY=4,---
* OVERLAY A GROUP 2
      SYCTTBL SBSP=SUBSYSC,OVLY=5,---
```

Within one overlay segment, a substructure may be defined for subroutines called by, and linked with, a particular subsystem, as illustrated by OVERLAY AB; SUBX and SUBY in Figure 3-8. The subroutines may not give up control to the Dispatcher (no calls to the File Handler, etc.); if such logic is essential, the subsystem of the called subroutine must be defined as single-thread processing. Otherwise, calls in different message threads processed concurrently for that subsystem will cause the overlay substructure to be "overlaid" by mistake.

The appropriate control cards for eligible Overlay A Intercomm routines may be generated via the ICOMLINK parameter OVLYSTR=YES which also causes inclusion of LOADOVLY in the Intercomm linkedit. For asynchronous overlay loading, also code ASYNCH=YES on ICOMLINK (causes an include for ASYNCH), and code ASYNLDR=YES on the SPALIST macro.

```
//LKED.SYSIN  DD     *
            INCLUDE .
                . required Intercomm modules
                .
            INCLUDE SYSLIB(SUBSYSA)
            INCLUDE SYSLIB(SUBSYSB)
            INCLUDE SYSLIB(SUBSYSC)
            OVERLAY A
                .
                . Intercomm Overlay A modules
                .
            OVERLAY A
                INSERT SUBSYSA
                INSERT SUBSYSB
            OVERLAY A
                INSERT SUBSYSC
            OVERLAY AB
                INSERT SUBX
            OVERLAY AB
                INSERT SUBY
```

Figure 3-8. Sample Linkedit Control Cards for Overlay Region A Subsystems

3.9.4 Dynamically Loaded Subsystems

No special table entries are required for dynamically loadable subsystems, other than the LOADNAM and REUSE parameters on the SYCTTBL macro. If the BLDL parameter indicates YES, the Subsystem Controller searches the STEPLIB or JOBLIB directory only once for the required member location. Thereafter, loading is performed based upon an internally generated list of actual file locations. The system control command, LOAD, must be used to indicate a change in location. Each dynamically loaded subsystem is linkedited independently of the main Intercomm load module.

The subsystem load module consists of the subsystem itself and any called modules (compiler-oriented routines not loaded dynamically by compiler-oriented code) which are not standard Intercomm/user subroutines accessible via REENTSBS. Assembler Language subsystems should load Intercomm facility addresses from the SPA/SPAEXT before calling an Intercomm routine, and use the MODCNTRL macro to access user subroutines defined to Intercomm via REENTSBS (SUBMODS macro). Each dynamically loaded subsystem module is then linkedited with the Intercomm interface INTLOAD (unless dynamic linkedit is used; see below). INTLOAD resolves references to resident Intercomm routines. The LKEDP procedure may be used for the subsystem linkedit, as the following illustrates:

```
//LINKSUBS      EXEC      LKEDP,Q=ABC,LMOD=DYNSUBX
//LKED.SYSIN    DD        *
                ENTRY     SUBSYSX
                INCLUDE   SYSLIB(SUBSYSX)
                INCLUDE   SYSLIB(INTLOAD)
                NAME      DYNSUBX(R)
```

The LOADNAM parameter of the SYCTTBL macro describing the subsystem must then correspond to the LMOD parameter of the LKEDP procedure (name of the module in the load library). If the subsystem is defined under more than one SYCTTBL (accessed by multiple verbs), linkedit with ALIAS names to make each definition unique, but do not link as either reusable or reentrant. This will result in more than one copy loaded in core, which cannot be avoided. The subsystem may, however, be defined to Intercomm as reentrant, if coded as reentrant.

The library used for dynamically loaded subsystems must be defined at execution time (STEPLIB or JOBLIB). Certain restrictions apply if the Dynamic Linkedit facility is used (see below).

Use of dynamically loaded subsystems requires an INCLUDE of the modules LOADSCT, DELOAD, and either ASYNCLDR (MVS, OS/MVT) or VS1LOADR (VS1, OS/MFT) for the resident portion of Intercomm. Coding DYNLOAD=YES (default) for the ICOMLINK macro automatically generates these statements. LOADSCT is used in conjunction with the LOAD command. MAXLOAD is the SPALIST macro system control parameter used with dynamically loaded subsystems.

3.9.5 Dynamic Linkedit Facility

The Intercomm Dynamic Linkedit facility is optionally used in conjunction with dynamically loaded subsystems to allow these subsystems to be linkedited with unresolved references to subroutines and data areas. If these subroutines and data areas are present (and resident) within the main Intercomm load module, the Dynamic Linkedit facility will resolve the references at startup time by "zapping" the load module of each subsystem.

Using this facility, the INTLOAD interface module is no longer to be linkedited with each dynamically loaded subsystem to resolve references to Intercomm resident routines, since they will be automatically resolved by Intercomm.

The Dynamic Linkedit facility is a generalized approach which permits a single copy of a compiler subroutine which is resident within the main Intercomm load module to be used by any loaded subsystem, rather than requiring a separate copy along with each loaded subsystem. Eliminating duplicate copies of subroutines in this manner is particularly useful for COBOL or PL/I loaded subsystems, since a single copy of all the standard library routines used by these languages can be made resident within Intercomm (if not in the Link Pack Area), and thus available to be used by all subsystems.

The Dynamic Linkedit facility is implemented by including the module ICOMDYNL in the main Intercomm linkedit. ICOMDYNL can be placed in the startup overlay. However, if the LOAD system control command is implemented, it must be resident. Coding DYNLINK=YES (default) for the ICOMLINK macro automatically generates the necessary statement. Also, the ICOMCESD and ICOMVCON modules must be separately linkedited with these names, and as nonreentrant, on one of the load libraries specified via STEPLIB or JOBLIB for Intercomm execution.

Additionally, a work file must be provided to Intercomm using the following format:

```
//DYNLWORK   DD   UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(1,1))
```

A listing of Dynamic Linkedit processing results, unresolved External References and WXTRNs will be produced by adding an optional DD statement to execution JCL:

```
//DYNLPRNT   DD   SYSOUT=A
```

If the LOADSCT routine is used to reload a dynamically loaded subsystem which has been relinkedited during Intercomm's execution, LOADSCT will use the Dynamic Linkedit facility to rezap the subsystem.

The following restrictions apply to the use of dynamic linkedit:

- Assembler Language address constants will not be resolved if coded as "label+nn" where "nn" is nonzero and less than or equal to 64K.
- Called programs must be resident in the Intercomm root segment for resolution to take place. This does not apply to dynamically loaded subroutines.
- A VCON referencing a module in an overlay segment will not be resolved. Thus, an Assembler Language program may use CALLOVLY only if it obtains the VCON of the called program from the Intercomm root segment, that is, from the System Parameter Area.
- Load modules on the library which is to be dynamically linkedited may not be executed by any other concurrent job. Since VCONs can only be resolved to point to one region, the load module is therefore executable only in that region.
- All modules to be dynamically linkedited during a given Intercomm execution must reside on one data set defined for STEPLIB or, if no STEPLIB, then on JOBLIB. This library must be contained in one extent. A careful watch of this library space is necessary to ensure that updates do not cause it to exceed one extent. Frequent off-line compresses may be necessary. This library may not be concatenated with others.
- However, if STEPLIB consists of concatenated data sets, the library containing load modules to be dynamically linkedited must be defined by a DD statement with the name DYNLLIB. This library must be a single data set, and must also be concatenated with STEPLIB for subsequent load processing. Code DISP=SHR on both DD statements. This library may not exceed one extent (see above) and may not be shared with any other Intercomm region.
- A combination of loadable subsystems linkedited with INTLOAD and dynamically linkedited loadable subsystems may be used. However, the INTLOAD group may not be on the dynamic link library, but must be on one of the other libraries concatenated to STEPLIB/JOBLIB. The INTLOAD library may be shared across regions.

- Compress of the dynamic link library may not be done while Intercomm is executing.

Because the load modules of dynamically loaded subsystems are modified, they cannot reside on a library shared by another Intercomm region. For efficiency, each dynamic load library should be on a different disk pack. To convert a subsystem from dynamically linkedited and loaded to resident or in the overlay region, the subsystem must be recompiled and relinkedited prior to inserting it into the Intercomm linkedit.

3.9.6 Subsystems Assigned to Overlay Region B, C or D

Some linkage editors limit the number of overlay regions that can be defined in a linkedit. Due to the existence of Intercomm Regions TRAN and SUB, not all of Overlay Regions B, C and D may be usable.

Overlay Regions B, C and/or D are used for subsystems which require no guaranteed response time. The objective of their use is to effectively remove some subsystems from contention for use of Overlay Region A. Subsystems assigned to Overlay Region B, C or D have the following characteristics:

- Input messages are queued by region, instead of by subsystem.
- Subsystem execution is controlled by the Intercomm program MONOVLY.
- Subsystem processing is always single-threaded.
- All subsystems in one overlay region should be coded in the same language.
- A Subsystem Control Table entry (SYCTTBL) is defined for MONOVLY, not the individual subsystem(s).
- An additional Verb Table is required for each overlay region.

There is one Subsystem Control Table entry for each of the Overlay Regions B, C or D, in use. Each defines MONOVLY as the entry point and the OVLY parameter is coded as 1, 2 or 3 for Overlay Region B, C or D, respectively. For example:

```
OVLYB     SYCTTBL   SUBC=B,SUBH=B,OVLY=1,LANG=NBAL,                X
                    SBSP=MONOVLY,NUMCL=2,DFLN=OVLYBQ
```

MONOVLY controls the loading of the appropriate subsystem into the overlay region, based upon the order of messages retrieved from the queue, and a table specification relating the message verb to the subsystem entry point.

Subsystems assigned to Overlay B, C or D, and coded in different languages, should have a Subsystem Control Table entry for an overlay region for each programming language. For example:

```
COBOVLYB    SYCTTBL    SUBC=B,SUBH=C,OVLY=1,LANG=COB,                X
                       SBSP=MONOVLY,NUMCL=2,----
BALOVLYC    SYCTTBL    SUBC=B,SUBH=A,OVLY=2,LANG=NBAL,               X
                       SBSP=MONOVLY,NUMCL=2,---
```

The Intercomm Enqueue/Dequeue facility (PMINQDEQ) is used to force single-threading of the overlay region. Two restrictions exist if more than one language is used per overlay region: the startup module CKOVLYNO may not be used to perform Subsystem Control Table checking; and the conversational control routine, CONVERSE, may not be called by any subsystem assigned to the overlay region.

BTVERB entries in the Front End Verb Table (BTVRBTB) must use the subsystem code assigned to the overlay region via the SYCTTBL macro. An Overlay Region Verb Table is required for each overlay region. This special verb table must have a Csect name of OVLYBTB for Region B, OVLYCTB for Region C, and OVLYDTB for Region D. These Csects are coded by the user, and must include an entry for each subsystem in the particular overlay. Each table entry is twelve bytes in length, as follows:

- Bytes 1-4--the four-character verb associated with a subsystem in the overlay region
- Byte 5--Verb Identifier/Edit Flag: X'00' = editing required; X'01' to X'254' = user VMI value; X'FF' = no editing desired
- Byte 6--X'FF' indicates free the incoming message before calling the subsystem, if desired, else code X'00'
- Bytes 7-8--unused
- Bytes 9-12--the subsystem entry point, coded as a VCON

A fullword of X'FF', generated by the PMISTOP macro, is required at the end of the table. Following is a sample overlay region verb table.

```
OVLYBTB  CSECT
         DC     C'EPKF',4X'0',V(EDITTEST)
         DC     C'EPKV',4X'0',V(EDITTEST)
         DC     C'V250',4X'0',V(EDITTEST)
         DC     C'EDKF',4X'0',V(EDITTEST)
         DC     C'EDPV',4X'0',V(EDITTEST)
         DC     C'EDPL',4X'0',V(EDITTEST)
         DC     C'ED32',4X'0',V(EDITTEST)
         PMISTOP
         END
```

As illustrated below, the Verb Identifier/Edit Flag controls processing of incoming messages via the Edit Utility based upon a test of the message header VMI field.

Message Header VMI Value	Verb ID/ Edit Flag (Verb Table Byte 5) Value	Action
X'FF'	ignored	No editing required. The message text verb is used to locate the table entry defining the subsystem to process the message.
X'00'	X'00'	Same as above. Edit Utility is not called.
X'00'	X'01' to X'FE'	Edit Utility is called prior to giving control to the subsystem.
X'01' to X'FE'	X'01' to X'FE'	Editing is not required. The message header VMI is matched with the Verb Identifier to locate the table entry defining the subsystem to process the message.

The MONOVLY program checks the input verb or the VMI against the table and calls the Edit Utility, if specified by the table entry. It then brings the program into the overlay area and passes control to the program. If the Overlay Region Verb Table is invalid, a message is issued and a Snap 90 is taken; then the overlay monitor returns to the Subsystem Controller with a return code of 4.

If asynchronous loading (ASYNLDR=YES in the System Parameter Area, and the module ASYNCH is present) is being used, the module LOADOVLY must be present. It is a necessary interface between MONOVLY and the Loader Task ASYNCH. To generate the correct linkedit for MONOVLY processing, the following must be coded for the ICOMLINK Macro: MONOVLY=YES, ASYNCH=YES, OVLYSTR=YES and optionally TRANS=YES.

3.10 SUBSYSTEM INTERFACES AND LINKEDIT CONSIDERATIONS

There are no special considerations for coding or linking of Assembler Language subsystems except that they should be reentrant and use the Intercomm facilities described in the Assembler Language Programmer's Guide. Macros supplied by Intercomm to aid in coding Assembler Language programs and subroutines are described in Basic System Macros. Considerations for higher-level language programs supported by Intercomm are described below.

3.10.1 COBOL Subsystem Interfaces

Application subsystems may be coded in COBOL-F or VS or ANS COBOL, and may also be compiled via the CAPEX Optimizer. However, all COBOL subsystems must use the same compiler, because the ILBO subroutines may not be compatible. An Intercomm facility allows COBOL subsystems to operate in a reentrant mode, processing several messages concurrently, as specified by the Subsystem Control Table entry for the subsystem. Certain coding conventions must be followed, as described in the Intercomm COBOL Programmers Guide.

The size of the Dynamic Working Storage in the Linkage Section of a reentrant COBOL subsystem must agree with SYCTTBL macro values. The COBOL Programmers Guide details coding techniques required when the amount of storage freed is less than the amount of storage obtained for the processing of a message. Two SYCTTBL parameters, GET and FREE, are used to specify the amount of dynamic core to obtain on entry to, and free on return from, a reentrant COBOL subsystem. The maximum request for storage via the GET parameter is 64K, less 304 bytes.

The Reentrant Subroutine Table (REENTSBS) must be included for execution of reentrant COBOL subsystems. This table represents a list of Intercomm service routine addresses referenced by a COBOL program parameter list for the reentrant subroutine interface module COBREENT. User additions to this list may begin at decimal offset 104 and be coded in a copy member USRSUBS. User-coded subroutines require an entry in this member and COBREENT must be used to interface to a called subroutine. Additionally, the supplied COBOL program COPY member ICOMSBS must be updated to provide the names and index codes for the added user subroutines.

Figure 3-9 illustrates the standard Intercomm-supplied Reentrant Subroutine Table. REENTSBS must be reassembled and relinked every time an entry is changed or added to USRSUBS.

3.10.2 COBOL Subsystem Linkedit Considerations

To execute COBOL subsystems under Intercomm, the interface modules PREPROG, PMICOBOT, and COBPUT must be included in the Intercomm linkedit (automatic if the ICOMLINK parameter COBOL=YES (default) is coded). Depending on the version and compiler NORES options used, COBOL programs require certain COBOL routines (based on coding logic) to be available from SYS1.COBLIB, either at linkedit time or at execution time. These modules are ILBOSRV, ILBOBEG, ILBOCMM, and ILBOMSG.

In addition, ILBOSTP0 and ILBOSTP1 may be required if they are not entry points within the ILBOSRV module. The modules have several subroutines (indicated by a suffix code) which may or may not be linkedited with them on SYS1.COBLIB, depending on the COBOL version (release) used, and weak external reference specifications in routines of that version. Normally, to cut down on the size of the COBOL load module, an execution time library is required if all COBOL routine external references are not resolved at linkedit time. This execution time library provides COBOL subroutines for the COBOL program only when needed, thus saving space in the user's region via LOADs and DELETEs. For example, ILBOBEG0 and ILBOCMM0 will always be needed, whereas ILBOMSG0 only if an error occurs. If EXHIBIT or READY TRACE is coded, adding an INCLUDE for ILBODSP0 to the Intercomm linkedit may be advisable.

To save space in the Intercomm region, COBOL subsystems should be compiled with the same compiler, using the NORES, and NOTRUNC options. For dynamically loaded COBOL subsystems defined to Intercomm as reentrant (SYCTTBL-LANG=RCOB), use the REUS and NCAL linkedit options. In addition, to save LOAD and DELETE time (if subroutine not in Link Pack), the ICOMLINK parameter RECOBOL=YES (default) should be used to generate INCLUDEs not only for Intercomm routines required for reentrant COBOL (COBREENT, COBSTORF), but also for the most common COBOL subroutines (ILBOSTP0, ILBOBEG0, ILBOCMM0, ILBOMSG0 and ILBOCOM0), and for the Intercomm/user subroutine table REENTSBS.

If following the above recommendation is not possible, due to the COBOL version in use, the user is advised to perform the following steps:

1. Linkedit ILBOSRV0 (PARM='REUS') into a special SRV library, with INCLUDE statements for subroutines ILBOBEG0, ILBOCMM0 and ILBOMSG0, as follows:

```
INCLUDE SYSLIB(ILBOSRV0,ILBOBEG0,ILBOCMM0,ILBOMSG0)
ALIAS ILBOSR,ILBOSRV0,ILBOSRV1,ILBOST,ILBOSTP0,ILBOSTP1
NAME ILBOSRV(R)
```

2. Then concatenate that special SRV library ahead of the regular COBOL library in the SYSLIB data sets for the linkedit of the COBOL subsystem.

3. Additional ALIAS names may be used for ILBOSR3, ILBOSRST, ILBOBEG, ILBOCMM, ILBOCMM1, ILBOMSG, and ILBOCOM depending on unresolved references in the COBOL subsystem linkedit.

4. The ENDJOB compiler option should be used to prevent 80A, 804 and 906 abends if the subroutine library is used.

NOTE: ANS Version 4 or CAPEX Optimizer routines might be on a library other than SYS1.COBLIB. Research this point for proper compile and linkedit SYSLIB JCL when using Intercomm procedures, and execution time STEPLIB JCL.

```
REENTSB1 CSECT
*
*  NEGATIVE OFFSETS ARE USED BY SPECIFYING AN OFFSET ENDING IN B'11',
*  WHICH IS INCREMENTED BY 1 AND COMPLEMENTED TO OBTAIN TRUE OFFSET
*  BY COBREENT AND PMIPL1.
*
         SUBMODS   NAME=MAPFREE    OFFSET -92,CODED AS 91
         SUBMODS   NAME=FECMRLSE   OFFSET -88,CODED AS 87
         SUBMODS   NAME=FESEND     OFFSET -84,CODED AS 83
         SUBMODS   NAME=FESENDC    OFFSET -80,CODED AS 79
         SUBMODS   NAME=ALLOCATE   OFFSET -76,CODED AS 75
         SUBMODS   NAME=ACCESS     OFFSET -72,CODED AS 71
         SUBMODS   NAME=MAPURGE    OFFSET -68,CODED AS 67
         SUBMODS   NAME=MAPCLR     OFFSET -64,CODED AS 63
         SUBMODS   NAME=MAPEND     OFFSET -60,CODED AS 59
         SUBMODS   NAME=MAPOUT     OFFSET -56,CODED AS 55
         SUBMODS   NAME=MAPIN      OFFSET -52,CODED AS 51
         SUBMODS   NAME=INTUNSTO   OFFSET -48,CODED AS 47
         SUBMODS   NAME=INTSTORE   OFFSET -44,CODED AS 43
         SUBMODS   NAME=INTFETCH   OFFSET -40,CODED AS 39
         SUBMODS   NAME=FECMFDBK   OFFSET -36,CODED AS 35
         SUBMODS   NAME=FECMDDQ    OFFSET -32,CODED AS 31
         SUBMODS   NAME=QWRITEX    OFFSET -28,CODED AS 27
         SUBMODS   NAME=QREADX     OFFSET -24,CODED AS 23
         SUBMODS   NAME=QWRITE     OFFSET -20,CODED AS 19
         SUBMODS   NAME=QREAD      OFFSET -16,CODED AS 15
         SUBMODS   NAME=QCLOSE     OFFSET -12,CODED AS 11
         SUBMODS   NAME=QOPEN      OFFSET -8,CODED AS 7
         SUBMODS   NAME=QBUILD     OFFSET -4,CODED AS 3
         ENTRY REENTSBS
REENTSBS DS    0A                 ALLOW FOR NEGATIVE OFFSETS
         DC    A(REENTEND-REENTSBS-4)       REQUIRED
         SUBMODS   NAME=SELECT     CODE 4-  FILE SELECT
         SUBMODS   NAME=RELEASE    CODE 8-  FILE RELEASE
         SUBMODS   NAME=READ       CODE 12- FILE READ
         SUBMODS   NAME=WRITE      CODE 16- FILE WRITE
         SUBMODS   NAME=GET        CODE 20- FILE GET
         SUBMODS   NAME=PUT        CODE 24- FILE PUT
         SUBMODS   NAME=RELEX      CODE 28- RELEASE EXCL. CONTROL
         SUBMODS   NAME=FEOV       CODE 32- FILE FEOV
         SUBMODS   NAME=DISEL      CODE 36- DISAM SELECT
         SUBMODS   NAME=DIREL      CODE 40- DISAM RELEASE
         SUBMODS   NAME=DIREAD     CODE 44- DISAM READ
         SUBMODS   NAME=DIWRITE    CODE 48- DISAM WRITE
```

Figure 3-9. REENTSBS Release Version (Page 1 of 2)

```
         SUBMODS  NAME=DIGET       CODE 52- DISAM GET
         SUBMODS  NAME=DIPUT       CODE 56- DISAM PUT
         SUBMODS  NAME=DIDEL       CODE 60- DISAM DELETE
         SUBMODS  NAME=DIRELEX     CODE 64- DISAM RELEX
         SUBMODS  NAME=COBPUT      CODE 68- COBOL MESSAGE SWITCHING
         SUBMODS  NAME=MSGCOL      CODE 72- MESSAGE COLLECTION
         SUBMODS  NAME=COBSTORF    CODE 76- COBOL STORFREE
         SUBMODS  NAME=CONVERSE    CODE 80- CONVERSE
         SUBMODS  NAME=DBINT       CODE 84- DATA BASE REQUEST
         SUBMODS  NAME=LOGPUT      CODE 88- LOGPUT
         SUBMODS  NAME=PAGE        CODE 92- PAGE ROUTINE
         SUBMODS  NAME=GETV        CODE 96- VSAM GET
         SUBMODS  NAME=PUTV        CODE 100-VSAM PUT
*************************************************************************
***      INSERT   USER    SUBMODS    MACROS                            ***
*************************************************************************
         COPY  USRSUBS
REENTEND EQU   *                   REQUIRED AFTER LAST SUBMODS
         ENTRY REENTEND
REENTSB1 CSECT
         END
```

Figure 3-9. REENTSBS Release Version (Page 2 of 2)

3.10.3 PL/1 Subsystem Interfaces

In the Intercomm environment, a PL/1 subsystem requires special consideration for each allowable option. Specifications of the options chosen are indicated for the subsystem in the PL1 and PL1LNK parameters of the SYCTTBL macro. These options are as follows:

1. The PL/1-F compiler, specified via PL1=F, the default on the SYCTTBL macro or the PL/1 optimizing compiler, specified via PL1=OPT on the SYCTTBL macro.

2. The linkage conventions used by Intercomm to construct the parameter list may be either nonbased (character string) or based (dummy arithmetic scalar) format for the first three parameters in the list, as specified by the PL1LNK parameter of the SYCTTBL macro.

An Intercomm module is required as the interface between Intercomm and the PL/1 compiler in use, either PREPL1 or PREPLI, as shown in Figure 3-10. Figure 3-11 illustrates the interface when the subsystem is dynamically loaded.

PREPL1 is the interface for PL/1-F subsystems. Each thread of a PL/1 subsystem is a separate instance of the PL/1 environment. For the F compiler, PREPL1 issues SPIE and STAE to override PL/1's SPIE and STAE during execution of the thread.

All storage allocation is performed in the usual PL/1 fashion--abnormal termination, which does not raise the ERROR condition, such as program checks, may leave storage allocated after the thread terminates. Storage obtained by PL/1, that is, automatic variable, is not monitored by the Intercomm Resource Management facility.

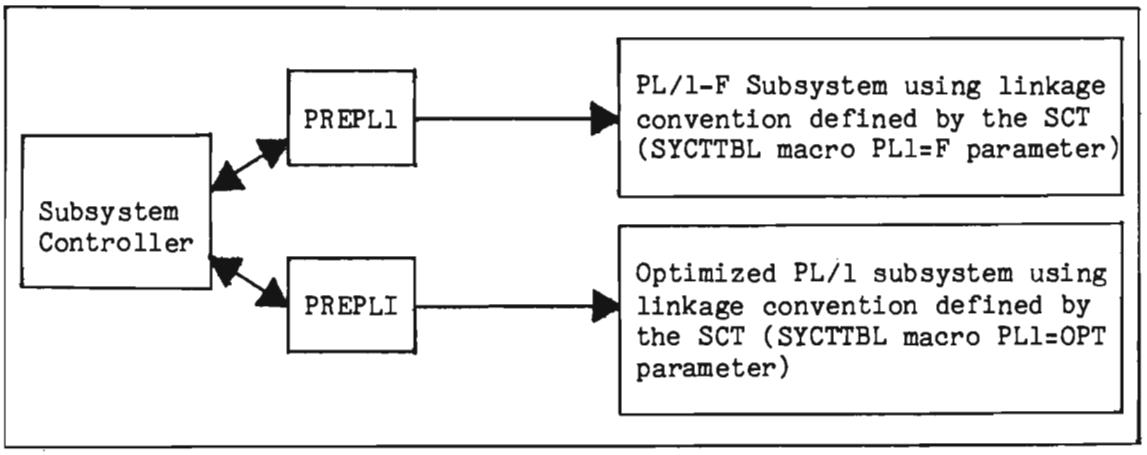


Figure 3-10. PL/1 Subsystem Interface Options

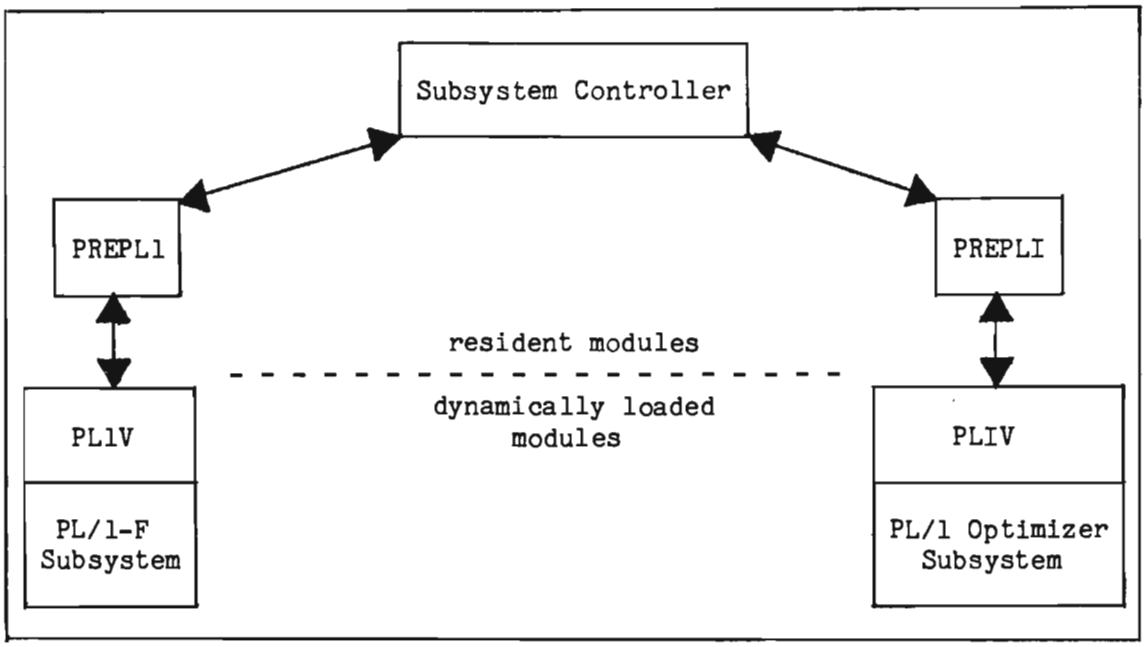


Figure 3-11. Dynamically Loaded PL/1 Subsystems

PREPLI is the interface module for PL/1 optimized subsystems. As released, PREPLI specifies no options. PL/1 invocation options STAE, SPIE and REPORT should be disabled for production. However, they may be specified by changing the PREPLI macro coded within the member PREPLI, then reassembling PREPLI. The Intercomm System Manager may provide an alternate PREPLI module for testing, specifying some or all of the above options. As with subsystems compiled by the F-compiler, each thread is a separate instance of the PL/1 environment. Since PL/1 STAE and SPIE can be suppressed by invocation options, Intercomm STAE and SPIE will remain effective.

Another option available to optimizer users is preallocated ISA, which allows PREPLI to allocate the ISA from Intercomm storage, based on the specified size on the SPAC parameter of the SYCTTBL macro, and to pass it to the subsystem. This makes clean abnormal thread termination possible where the ERROR condition is not raised.

The subroutine interface program PMIPL1 must be used. When calling non-PL/1 subroutines, it will reformat the parameter list to pass data addresses. Subroutines are referenced by specifying the offset into the REENTSBS table as the first parameter. The offsets are defined for PL/1 in the copy member PENTRY. If a subroutine not currently represented in REENTSBS is called, both tables must be updated. When coding user entries in REENTSBS, PMIPL1 assumes all parameters are passed in character format (with the exception of MSGCOL, PAGE and CONVERSE). This method can be bypassed when using the optimizer.

For optimizing compiler users, PMIPL1 functions can be achieved for Assembler Language subroutines by copying the member PLIENTRY into the subsystem, or by declaring the subroutine, for example, COBPUT, as

```
DCL  COBPUT ENTRY OPTIONS  (ASM INTER)
```

and calling, in the usual PL/1 fashion:

```
CALL COBPUT (message, return-code)
```

Dynamically loaded PL/1 subsystems must be linkedited so that the load module, specified by the SYCTTBL macro LOADNAM parameter, contains the address table PL1V (for F subsystems) or PLIV (for Optimizer subsystems). PL1V or PLIV must be specified as the load module entry point via a linkage editor ENTRY statement.

Additional compiler-dependent linkedit considerations are:

- PL/1(F)

  The address table PL1V must be first in the module so that the start of all pseudo-register vectors have the same format. Also, PL1V must be included during the resident linkedit before all PL/1 modules, so that the order of pseudo-register vectors match those of the dynamically loaded modules. IHESAPA and IHESIZE must be included in every dynamically loaded subsystem.

- PL/1 (Optimizer)

  There are no special ordering requirements for either dynamically loaded subsystems or the resident linkedit. This simplification is possible because the library does not use pseudo-registers, as does the F implementation.

In the PL/1 subsystem, the procedure given control by Intercomm must specify OPTIONS(MAIN,REENTRANT), or OPTIONS(MAIN), if nonreentrant. OPTIONS(MAIN) is used to get the true subsystem entry point in Csect IHEMAIN(F) or PLIMAIN(OPT). Since resident or overlay subsystems use the SBSP parameter on the SYCTTBL macro for this purpose, for them OPTIONS(MAIN) is not needed but will be accepted.

The subsystem should avoid unnecessary data conversion to keep PL/1 library routines called by the subsystem to a minimum. If Dynamic Linkedit is used, some or all of the PL/1 library subroutines may be included in the resident portion of Intercomm, eliminating their duplication in each dynamically loaded subsystem that references them.

PL/1 library subroutines eligible for residency are those normally included via automatic library call (control section name, preceded by an asterisk in the link map listing). Either specify the NCAL linkage editor option to remove all control sections, or prevent automatic call of selected control sections (see below) via linkage editor LIBRARY statements. Use of LIBRARY statements to exclude a standard set of commonly used routines allows the automatic library call to include infrequently used modules when referenced, eliminating special programmer effort once a set of resident routines have been selected by examining typical linkedits.

3.10.4 PL/1 Subsystem Linkedit Considerations

PL/1 subsystems necessitate inclusion in the Intercomm linkedit of the Intercomm Abend Intercept Routines SPIEEXIT and STAEEXIT, as well as the PL/1 interface routines PMIPL1, COBPUT, PREPL1 and PREPLI, as required. Additionally, the common subroutines IHEMAIN, IHESAPA, IHELTTA, IHESADA, and IHESAFA should be resident. Coding PL1=F or OPT on the ICOMLINK macro automatically generates the necessary include statements for the above (except COBPUT).

When using the PL/1 optimizing compiler, the transient library modules are loaded into dynamic storage as required. With a relatively high message volume for Pl/1 subsystems, a high overhead can be encountered while loading and deleting the transient library modules. To ease this problem, load some of the most used modules at startup time (via USRSTRT1), such as IBMBPGRA, IBMBPIIA and IBMBPITA, or make them resident in the Intercomm linkedit.

The Optimizer uses three transient modules which are loaded and deleted for each thread. They are IBMBPII, initialization; IBMBPIT, termination; and either IBMBPGR, transient library storage management, or IBMBPIR, resident library storage management with REPORT. To keep them resident, thereby greatly improving response time, the USRSTRT1 user exit routine could also load them at startup and a USRCLSE1 user exit routine could be written to delete them at closedown.

3.10.5 FORTRAN Subsystems

Application subsystems coded in the FORTRAN language are executed under Intercomm in the same manner as nonreentrant COBOL subsystems. They are single-threaded. Their SYCTTBL macros should specify LANG=FORT and MNCL=1. They must be linkedited with compiler-dependent subroutines; see the description of the FORTLINK procedure in Chapter 2.

3.11 SUBROUTINE INTERFACES AND LINKEDIT CONSIDERATIONS

The following subsection describes the use of user-coded subroutines with user-coded subsystems and their residency and linkedit considerations. For further details, see the applicable Programmers Guide.

3.11.1 Resident Subroutines

Resident and Overlay A Assembler Language subsystems may call resident Assembler subroutines using standard linkage conventions. Dynamically loaded Assembler Language subsystems must either be dynamically linkedited with the resident subroutines, or use the MODCNTRL macro to access user subroutines previously defined via the SUBMODS macro in REENTSBS; Intercomm routines may be accessed via VCONs in the SPALIST.

Resident, Overlay A and dynamically loaded COBOL and PL/1 subsystems must use Intercomm interfaces to all noncompiler subroutines. The interface routines are COBREENT and PMIPL1, respectively. The user subroutines are defined to Intercomm via the SUBMODS macro in the REENTSBS table. Copy code tables to define subroutine codes to match entries in REENTSBS are ICOMSBS (COBOL) and PENTRY (PL/1). PL/1-Optimizer subsystems may optionally call resident Assembler Language subroutines directly by adding the name to the PLIENTRY table included in the program; however, this option cannot be used for dynamically loaded subsystems unless dynamically linkedited.

A maximum of 350 user SUBMODS entries using the NAME parameter (resident), or LNAME and RES=LINKEDIT or RES=BOTH (default) parameters, may be defined (due to an Assembler restriction on ESD entries). An additional 49 are reserved for Intercomm service routine definitions. However, additional entries may be defined using the LNAME and RES=LOADMOD parameters of the SUBMODS macro. See also the PERMRES parameter, as described in Basic System Macros.

Note the following language interface considerations:

- Reentrant COBOL subsystems must use the Intercomm interface COBREENT to call subroutines, and may only call reentrant or reusable COBOL and Assembler Language subroutines.
- Reentrant COBOL subroutines may be called only by reentrant COBOL subroutines and subsystems which use the COBREENT interface.
- PL/1 subroutines may not be called by Assembler or COBOL subroutines or subsystems due to language differences in parameter list construction.
- Reentrant PL/1 subsystems must use the Intercomm interface PMIPL1 to call PL/1 subroutines; COBOL subroutines may not be called. See the discussion of Resident Subroutines (above) for Assembler subroutine interface considerations.
- Nonreentrant COBOL and PL/1 subsystems may call only nonreentrant or reusable subroutines and reentrant Assembler subroutines. Nonreentrant Assembler subsystems and subroutines may call reentrant Assembler subroutines if standard linkage conventions are used.

3.11.2 Subroutines Linked with Dynamically Loaded Subsystems

Use of this convention is not recommended under Intercomm as it impacts reentrancy and multithreading, in addition to adding to the size of the load module.

3.11.3 Dynamically Loaded Subroutines

Intercomm subsystems have the ability to link to dynamically loaded subroutines. For all languages, these subroutines must be defined in REENTSBS using the SUBMODS macro. The loaded subroutines will be dynamically linkedited at startup time to resolve any unresolved VCONs and then loaded as required when accessed by a subsystem. A BLDL list for each subroutine may optionally be maintained for efficiency. Loaded subroutines will be automatically deleted from storage after a user-specified period of inactivity. Optionally, a subroutine can be loaded at startup and then made resident for the duration of the Intercomm execution (see PERMRES parameter of the SUBMODS macro).

Subroutines may be dynamically loaded during testing and then later be made resident or defined for the subroutine overlay region with no changes to the application. New versions of dynamically loaded subroutines can be obtained during Intercomm execution by use of the LOAD system control command (except if made resident at startup).

Intercomm imposes no size restriction for these subroutines. Dynamic subroutine loading is dependent upon storage availability. Loading is overlapped through the use of subtasking. Subroutines which issue INTENQ/DEQ or process file I/O, which might cause a time-out, should not be dynamically loadable, unless made resident at startup.

3.11.3.1 Application Programming Conventions

Language-dependent considerations for application subsystem coding are as follows:

- Reentrant COBOL subsystems use COBREENT and REENTSBS in the standard manner; dynamic load is transparent to the application program. COBOL subroutines must be coded and defined to Intercomm as reentrant.

- PL/1 subsystems must call PMIPL1 in the standard manner (the ENTRY option of the Optimizer is not allowed for dynamic-loaded subroutine reference); dynamic load is transparent to the application program. Dynamically loaded subroutines written in PL/1 require special linkedit considerations. In order to maintain the PL/1 environment constructed for the calling subsystem, the PL/1 initialization routines generated by the compiler must be removed, and the subroutine entry point must be explicitly specified. This can be accomplished by the following linkedit control cards for the subroutine (with the name SUBROUT):

```
REPLACE   PLIMAIN
REPLACE   PLISTART
INCLUDE   SYSLIB(SUBROUT)
ENTRY     SUBROUT
NAME      SUBROUT(R)
```

- Assembler subsystems must issue a MODCNTRL macro to invoke dynamic subroutine load.

Nonreentrant COBOL and FORTRAN subsystems may not use the Dynamic Load facility directly. The user may provide a reentrant interface routine in Assembler Language for those subsystems.

3.11.3.2 Implementation

The macro SUBMODS is coded in REENTSBS and defines the name and characteristics of the subroutine (deletion time, residency, etc.) and may specify a BLDL list (see Basic System Macros). A separate Csect, DYNLSUBS, is generated to contain control data for dynamically loaded subroutines. The modules PMIDLOAD, DYNLLOAD, and REENTSBS must be included in the Intercomm linkedit. Coding DYNLOAD=YES and DYNLINK=YES on the ICOMLINK macro will generate the necessary INCLUDE statements. See the description of dynamically loaded subsystems and the Dynamic Linkedit facility for further installation details.

3.11.4 Transient Subroutine Overlay Region (TRAN)

The Intercomm Transient Subroutine Overlay Region allows rarely used Intercomm and Assembler Language application subroutines (which may give up control) to be linkedited as separate overlay segments in an overlay region reserved for this purpose. This can significantly reduce the resident storage requirements of such Intercomm and application subroutines.

To be eligible for the transient area, a subroutine and its callers must follow several rules:

- All callers of the subroutine in the transient area must call the transient area using the CALLOVLY macro.
- The subroutine in the transient area must, in all cases, return eventually to the calling program. It cannot branch away forever into some other module. It must return.
- Usage of the transient area cannot be nested; that is, no subroutine to be used in the transient area can CALLOVLY another subroutine which is also in the transient area. It can, however, CALL resident subroutines. (See Figure 3-12.)
- The subroutine in the transient area must be serially reuseable or reentrant, and must follow standard linkage conventions.
- The caller must be an Assembler Language program. If the user wishes to use a high-level language and call a transient subroutine, he must do the following:

  1. Write a reentrant Assembler Language interface, using standard linkage conventions, to issue the CALLOVLY for the high-level program, and define it in REENTSBS.
  2. Parameters to be passed to the subroutine in the transient area must initially be passed to the Assembler Language interface by the high-level language. (See Figure 3-13.)

3. The high-level language caller of the Assembler Language interface must be defined as reentrant, that is, provide save area chaining.

- The subroutine in the transient area must invariably complete its processing within five minutes. The time-out interval is fixed by the Intercomm transient subroutine handler. After this time, it will be subject to being overlaid by other subroutines.

	Allowed		Not Allowed
ASUB	CSECT . CALLOVLY BSUB . END	ASUB	CSECT . CALLOVLY BSUB . END
BSUB	CSECT . CALL DSUB . END	BSUB	CSECT . CALLOVLY CSUB . END

Figure 3-12. Illustration of Nested CALLOVLY Coding Conventions

```
     .
     CALL 'COBREENT' USING CSUBI-code, Parameter-A, Parameter-B
      .
-----------------------------------------------------------------------
   * REGISTER ONE CONTAINS THE COBOL PARAMETER LIST ADDRESS
             CSUBI   CSECT
                     USING   *,12
                     STM     14,12,12(13)
                     LR      12,15
                     LR      2,1
                     STORAGE ADDR=8(13),LEN=72,RENT=NO
                     L       3,8(13)
                     ST      13,4(3)
                     LR      13,3
                     LR      1,2
                     CALLOVLY CSUB,(1)
                     LR      1,13
                     L       13,4(13)
                     STORFREE LEN=72,ADDR=(1)
                     LM      14,12,12(13)
                     BR      14
                     END
```

Figure 3-13. Using CALLOVLY in an Assembler Language Interface for a High-Level Language Program

The set of Linkage Editor control statements illustrated below would result in a root section containing the resident subsystems PGM1 and PGM2, and in the Intercomm transient area, the subroutines SUB1, SUB2, SUB3 and SUB4. The transient subroutine OVERLAY and INSERT statements must be placed in the Intercomm linkedit after the Intercomm OVERLAY TRAN(REGION) statement.

```
 INCLUDE  SYSLIB(PGM1,PGM2)
INCLUDE  SYSLIB(SUB1,SUB2,SUB3,SUB4)
                    .
                    .
OVERLAY TRAN(REGION)
                  . Intercomm transient subroutines
OVERLAY TRAN
          INSERT SUB1
OVERLAY TRAN
          INSERT SUB2
OVERLAY TRAN
          INSERT SUB3
OVERLAY TRAN
          INSERT SUB4
```

PMIOVLAY and LOADOVLY must be included in the Intercomm linkedit. The appropriate control cards for these modules and applicable Intercomm routines in the Transient Subroutine Overlay Region may be generated via the ICOMLINK macro specifying TRANS=YES and requires coding of OVLYSTR=YES.

Since the OS linkage editor cannot create more than four overlay regions, the use of one of them as a transient area will restrict the application subsystems to the use of Intercomm Overlay Areas A, B and C.

Since the transient area is a serially reuseable resource, care must be taken not to use it for subroutines that, due to frequency of usage or duration of processing, will create a decrease in message throughput or delay system control functions.

3.11.5 Subroutine Overlay Region (SUB)

Intercomm provides an overlay region dedicated to rarely used Assembler Language subroutines which follow normal linkage conventions and never relinquish control to the Dispatcher (no I/O, no time delays, etc.). Some Intercomm routines are defined for this overlay region and thus accomplish a saving of 6-9K. ICOMLINK parameters are the same as for Overlay Region TRAN.

OVERLAY and INSERT statements, for user subroutines eligible for this area, must be placed in the Intercomm linkedit after the Intercomm OVERLAY SUB(REGION) statement, and INCLUDE statements must be added as described above for the TRAN area. Use of this area in addition to the TRAN area will restrict application subsystems to Overlay A and B only.

3.12 GENERALIZED SUBTASKING

The concept of using OS subtasks to perform operations containing inherent WAITs, (for example, GET, OPEN, CLOSE, etc.) has been generalized. At startup time the generalized subtasking facility will create a pool of general purpose subtasks which can thereafter be used to perform functions of this type. This facility, which is used by Intercomm system routines, is also available for use by Assembler-coded subsystems or subroutines. A SUBTASK macro is coded to specify a subroutine which is to receive control under a general subtask. The subroutine executes under the subtask, then returns control to the original routine at the next sequential instruction after the SUBTASK macro. The linkage between the issuer of the SUBTASK macro and the subroutine is similar to a CALL; all registers must be preserved and restored as they would be during a CALL.

The code executed as the subtask cannot relinquish control to any Intercomm service routines such as the Dispatcher, File Handler, etc. nor issue an OS WAIT macro. Execution of the subtask logic is synchronous with respect to the thread issuing the SUBTASK macro. The calling routine may be resident or dynamically loadable, but may not execute in an overlay area. The TCTV for the originating subsystem must be generous to prevent unnecessary time-outs.

3.12.1 Special Subtasks

Special subtasks are subtasks from the general pool which are reserved by Intercomm with a unique identification number. Special subtasks are defined to allow exclusive use of a subroutine. This is useful for subtasking subroutines which may only be executed serially, that is, nonreentrant code.

The first issuance of a SUBTASK macro with an ID number specified via the TASKNUM parameter causes Intercomm to fetch a subtask from the general pool, assign the ID number to it and place its address in the special subtask table. Control is then passed to the subroutine to execute under that subtask. For every subsequent SUBTASK macro with the same ID specified, Intercomm retrieves the source subtask and determines whether it is active. If it is active, an INTWAIT is performed until the subtask is free. When it is free, or if it was inactive, control is passed to the subroutine to execute under the subtask, and that subtask is marked active. The ID assigned to the subtask is unique and remains in effect until closedown.

The difference between a general subtask and a special subtask is that when a general subtask is requested (no ID is provided), an inactive subtask is chosen at random from the general pool and control is passed to the subroutine to execute under that subtask. If a special subtask is requested (an ID is provided with the SUBTASK macro), the subtask to which the ID is assigned is located, and control is passed to the subroutine only if the subtask is inactive, even though there may be other inactive general or special subtasks. This method forces serial reusability for the special subtasks.

If a subroutine is requested under a general subtask while it is executing under a special subtask, control will be passed to the subroutine and it will execute concurrently under both the general and special subtasks. In addition, if a subroutine is executing under one special subtask and that subroutine is requested for execution under a different special subtask (different ID number), control will be passed only if the second subtask is inactive. Intercomm can only determine whether a special subtask is active or free; it cannot determine whether the subroutine is active, nor can it associate special subtasks with subroutines. Thus, to prevent concurrent use of the subroutine by multiple requests, a subroutine should always be executed under the same special subtask ID.

As with general subtasks, special subtasks should not relinquish control to Intercomm, and they may not issue a WAIT or cause a program check. Intercomm does not use special subtasks.

3.12.2 Implementation

The number of general and special subtasks in the system is specified to Intercomm via the TASKNUM parameter of the SPALIST macro. If the number of special subtasks in TASKNUM is zero, special subtasks will not be allowed. The module ICOMTASK must be included in the linkedit if general and/or special subtasks are in use.

To execute a subroutine under a general subtask, code the SUBTASK macro in-line and omit the TASKNUM parameter. To execute a subroutine under a special subtask, code the SUBTASK macro in-line, and code the TASKNUM parameter with a valid subtask ID number (within the range specified for the SPALIST TASKNUM parameter).

The subroutine must be coded in Assembler and must be resident. Refer to Basic System Macros for coding specifications of the SPALIST TASKNUM parameter and the SUBTASK macro.

3.13 TIME CONTROLLED MESSAGE PROCESSING

The Subsystem Controller automatically generates messages based on the time of day, as dictated by the user's Time Zone Table. The user specifies through the parameters supplied in the table what Verb/Message Identifier is to be defined by the Subsystem Controller as part of the constructed message header. The message is sent, through Message Collection, to the specified subsystem. The message is processed at the time of day specified by the user. The format of the message produced by Intercomm is as follows:

- Byte 1-42: Standard Intercomm message header with:
  - MSGHSSCH set to binary zero, MSGHSSC to C'M'
  - MSGHRSC and MSGHRSCH fields set to the values supplied by the user
  - MSGHVMI field set to the value specified by the user
- Byte 43: Item Code=1
- Byte 44: Length=1
- Byte 45: Time Zone Code Value (supplied by user)
- Byte 46: Item Code=2
- Byte 47: Length=2
- Byte 48-49: Time to allow for processing of this message (specified by the user)

The Time Zone Table is constructed by coding one TMZONE macro for each message the user wishes to be automatically started by Intercomm based on the time of day. The TMZONE macros must be coded in a Csect named PMITIMTB. The end of the table must be delineated by the PMISTOP macro, which indicates the end-of-table condition at execution time. The receiving subsystem can further trigger later iterations of the same message via the Dispatcher. Such a subsystem might be used to:

- queue System Control Command messages
- start a remote input terminal or line
- generate a FECMDDQ for printer output

The module TRIGGER must be included as a resident program in addition to the resident Time Zone Table.

Following is a sample Time Zone Table:

```
PMITIMTB CSECT
* MESSAGE TO SUBSYSTEM AA AT NOON:
         TMZONE  SCHT=1200,PGID=A,PGIH=A,PVMI=N,TMZC=Z
*
* MESSAGE TO SUBSYSTEM XY AT 4:00 PM:
         TMZONE  SCHT=1600,PGID=Y,PGIH=X,PVMI=X,TMZC=Y
*
* END OF TABLE
         PMISTOP
         END
```

Chapter 4

TASK MANAGEMENT

4.1 DISPATCHER AND RELATED SERVICE ROUTINES

The Intercomm multitasking Dispatcher (IJKDSP01) controls all scheduling of task execution in the Intercomm environment, replacing the Operating System multitasking facility. All system programs (Front End, Subsystem Controller, File Handler, etc.) effect overlap of operation, interprogram communication and scheduling via the Dispatcher.

4.2 DISPATCHER QUEUES

The Dispatcher controls operation via task queues of three different types:

- Execution Queues

  Tasks which are executable, based upon their order of readiness within order of priority

- Event Queues

  Tasks which will become executable upon completion of an event, indicated via the posting of an Event Control Block; whether by the operating system (WAIT queue) or an internal posting (IPOST queue--see DISPATCH macro)

- Time Queue

  Tasks which will become executable at a particular time of day, or on completion of a timed wait.

Tasks are created via the DISPATCH or INTWAIT macros, described in Basic System Macros, and the Assembler Language Programmer's Guide.

4.2.1 Defining the Number of Task Queue Elements

The Dispatcher contains assembled space for task queue elements allowing up to 120 concurrent tasks (executable, event or time-dependent). Task queue elements not in use are members of a free queue element pool. Except in cases of very high message volume, this number of queue elements is satisfactory. The number of queue elements is a global specification:

&NUMWQES within INTGLOBE and SETGLOBE

To increase the number of queue elements, update the global setting in SETGLOBE and reassemble and link IJKDSP01. If the free queue is empty when a new task element is to be created, Intercomm abends with a user code 901 (see IJKTRACE description, below). To estimate the number of WQEs necessary for a high-volume system, add the number of SYCTTBLs generated for Front End processing to the number of BLINE macros and/or VTAM I/Os (RCVNO and RCVRSP on VCT macro), and the total MNCL across all subsystem SYCTTBLs, plus 50 for Intercomm processing.

4.2.2 IJKPRINT-Output to SYSPRINT

This Dispatcher-related service routine calls the PUT entry point in the File Handler to output a print line image whose address was passed to IJKPRINT in register 1. Print line images must be IBM standard format V (variable-length) records, with an ASA printer spacing control character as the first text byte. (Maximum logical record length is that defined in the JCL for SYSPRINT.)

A count is maintained of the number of lines printed on the text page; when the count exceeds sixty lines, the next line output will specify a skip to head of form (ASA control character '1'), and the line count will be reset.

Output is directed to the file with ddname SYSPRINT. If the file is undefined or incorrectly defined, no output is produced and no diagnostic indication is given.

The DD statement for SYSPRINT must define a DCB with DSORG=PS, RECFM=VA, or VBA, LRECL=137 and BLKSIZE=141 or a multiple of 137 plus 4.

Any program may, if desired, call upon this routine to perform routing of similarly formatted records to SYSPRINT. Control is not released to the Dispatcher during IJKPRINT processing.

4.2.3 IJKTRACE-List Dispatcher Queues

This service routine constructs print line images producing a formatted display of all Dispatcher task queues. It is called automatically whenever the program check handler (SPIESNAP) is entered for a snap 126, and by RMPURGE when purging a subsystem thread with outstanding resources not released by that thread. It is also called by the Subsystem Controller (SYCT400) when a subsystem times out (snap 118 produced), by STAXEXIT (for snaps 121 and 122), and by VTERRMOD for VTAM error recovery (snap 63). It may also be called for diagnostic purposes by any other program. The maximum line length is 120 characters, giving a maximum LRECL of 125. Successful execution of this program also requires inclusion of IJKCESD and IJKWHOIT in the Intercomm linkedit (see sections 4.2.4 and 4.2.5). IJKTRACE calls IJKPRINT to output the print line images to SYSPRINT (see above). For efficiency, the SYSPRINT data set should be blocked.

Each print line image is passed to the IJKPRINT routine for output to SYSPRINT. Fields are printed in hexadecimal format, unless otherwise noted. The following are detailed explanations of the elements of the listing:

- Heading Line 1--General information giving:

  -- The Julian date and time (decimal) at entry to the routine, as obtained from the operating system clock:

     IJKTRACE ENTERED DATE yy.ddd TIME hh.mm.ss.

  -- The byte specifying the priority and overlay group of the last program path given control by the Dispatcher:

     PRI/OVLY xx

  -- The byte specified by the last executed SETOVLY macro instruction (00 if no overlay structure is used):

     SETOVLY xx

  -- The caller (Csect name and displacement) of IJKTRACE:

     CALLED BY name+displacement

- Heading Line 2--Defines the list type, locations and activity:

  -- The Dispatcher list name whose task elements, if any, are printed below:

     aaaaa LIST

     In place of 'aaaaa' will appear the list type: FREE, WAIT, IPOST, TIME or EXEC.

  - The FREE list contains task elements that are unused or that represent program paths already either given control or cancelled, in the order in which these events occurred. The oldest (first) entry in the FREE list is reused when required for a new program path. The newest (last) entry is for the most recently dispatched task. Only the last 200 entries are printed. To print more or less, modify the local global &FQENUM in IJKTRACE.

  - The WAIT list contains task elements for program paths awaiting the posting of an Event Control Block (ECB) by Intercomm or the operating system. Task elements appear in the order in which the requests were made.

  - The IPOST list contains elements for program paths awaiting the posting of an ECB by Intercomm via an internal INTPOST request. Task elements in this list are in the order in which the requests were made.

- The TIME list contains task elements for program paths to be resumed at a given real time; the list is maintained in ascending real time sequence, with first-in first-out sequence for equal real time values.

- One EXEC (execution) list for each priority level (maximum=4) in the system contains task elements representing program paths ready to be given control, in the order that readiness was determined.

-- The addresses of the list table entry, the first task element, and the last task element in the list are displayed. Where a list is empty (zero count), all three addresses are equal:

WQT xxxxxx FIRST xxxxxx LAST xxxxx

-- The decimal count of task elements currently in the list, and the highest or lowest count value that has occurred since processing began (highest count for all but the FREE list):

COUNT aaaa HI/LO aaaa

NOTE: if the free queue LO value is below 10, the total number of task queue elements should be increased (see Section 4.2.1).

- Heading Line 3--Provides task element column descriptors if the list contains any task elements (nonzero count). This is followed by task element fields, one task element per line. The column headings are:

-- WQE--Address of the task element.

-- FLAGS--Letters corresponding to flag bits in the task element, as follows:

D--Program path has been given control (dispatched).

C--Program path has been cancelled.

E--Task element has been placed on execute list.

P--Task element is (has been) in the IPOST list.

I--WAIT list element is internal ECB.

T--Task element is (has been) in the TIME list.

W--Task element is (has been) in the WAIT list.

NOTE: where an invalid combination of flags has been detected, an asterisk (*) precedes the flags field.

-- PRI/OVL--The priority and overlay-group portions of the priority/overlay byte specified in the DISPATCH or INTWAIT macro instruction; the sum of these values is the value of the PRI/OVL field.

-- ECB/T--The ECB address or real time, where applicable. A real time is a 24-bit value with the least significant bit representing 1/37.5 of one second in this display. A description of converting timer units is provided in the chapter on "General Debugging Techniques" in Messages and Codes.

   This field is not printed for task elements that have been in neither a WAIT, nor the TIME list, and thus may be missing where an ECB was posted (internally or externally) prior to the issuance of wait request.

-- ENTRY PT--Address for transfer of control to resume the program path; the high-order (leftmost) byte contains the thread number in hexadecimal (if nonzero, subsystem processing created the task element).

-- PARAMETER--Value to be passed to program in register 1.

-- (ECB)--Value in ECB if the FLAGS field contains a W; it is the value before posting if the task element is in an event list (WAIT, IPOST).

-- TIME--for task element that is (was) in time list if the FLAGS field contains a T, the time it was (will be) dispatched or, if it was cancelled, the time it would have timed out.

-- CSECT--the Csect (+ displacement if any), that was (will be) given control of this task element (see ENTRY PT above). If the Csect name is not easily recognizable, refer to the Csect/Module name correspondence table for Intercomm system modules in Figure 4-1.

-- ENTRY--the entry point within the Csect at which this Csect was (will be) entered, if known (defined by an ENTRY statement within the module).

-- SUBSYS--if CSECT is SYCTRL (no ENTRY), or the thread is not zero (and the task element not on the Free Q); the subsystem code of the subsystem processing under this task element.

-- SUB NAME--if CSECT is SYCTRL (no ENTRY), or the thread is not zero (and the task element not on the Free Q); the name of the subsystem processing under task element.

An example of IJKTRACE output is shown in Figure 4-2.

CSECT Name	Module Name	Function
SYCTRL	SYCT400	Subsystem Controller
IXFMON00	IXFHND00	File Handler Initialization
IXFMON01	IXFHND01	File Handler Processing
IXFABWTO	IXFHND01	File Handler ISK
IXFSUBS	IXFHND01	File Handler save area processing
IXFMON09	IXFHND00	File Handler Closedown
PMISTUP	STARTUP3	Intercomm Startup
STUOVLY	STARTUP3	Intercomm Startup
RSMGMNT	MANAGER	Resource Management
RM....	MANAGER	Resource Management subfunction
MSGCOL	BLMSGCOL	Message queuing
PMISUBL2	--	SUBLINK macro subroutine
PMILINK2	--	LINKAGE macro subroutine

Figure 4-1. IJKTRACE - Csect/Module Name Correspondence Table

```
IJKTRACL ENTERED   DATE 83.076   TIME 18.35.14   PRIOVLY 00   SFTOVLY 0A   CALLED BY SYCTRL+130E

 FREE LIST    WQE 142490   FIRST 1429F0   LAST 142A40   COUNT 0054   HI/LO 0043

 WQE      FLAGS    PRI OVL  ECB/T    ENTRY PT  PARAMETER  (ECB)      TIME        CSECT              ENTRY     SUBSYS   SUB NAME
1429F0    CT       00 00    266C67   051A5D2C  153RR000              18:39:09.84  PMINODE0+46C
1426R0    CT       00 00    266C5F   051A5D2C  153F3000              18:39:09.63  PMINODE0+46C
1426F0    CT       00 00    26RF7D   00147336  0015FC40              18:43:09.31  SYCTRL+1246       PURGE
1425F0    DFW      00 00    152938   041AC5CF  001DF0R0                           IXFMON01+2756     OVRLAPIO
142960    DEW      00 00    152938   041AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142R20    DEW      00 00    15293R   041AC5CF  0015E6R8                           IXFMON01+2756     OVRLAPIO
1425A0    CT       00 00    264F1E   00147336  001601AD              18:35:49.92  SYCTRL+1246       PURGE
1427D0    DE       00 00             00144868  001RR4C0                           LOGPUT+8FR
1426C0    DFW      00 00    1524RR   001AC5CE  0015F6R8                           IXFMON01+2756     OVRLAPIO
142590    DET      R0 00    264948   0012C438  00127DE0              18:35:10.08  BTAMSIM+A50
142860    DET      80 00    264949   0012C3R0  001699D8              18:35:10.11  UTAMSIM+998
142AE0    DEW      R0 00    127DE0   0013152C  00156RE0                           BLHOT+E7C
142R40    DEW      80 00    152988   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142AR0    DET      80 00    26494D   0012C91E  0016RFF8              18:35:10.21  BTAMSIM+F36
142C40    DEW      R0 00    1140C0   001AC5CF  0015E6R8                           IXFMON01+2756     OVRLAPIO
142970    DFW      R0 00    1140C0   001AC5CF  0015F6R8                           IXFMON01+2756     OVRLAPIO
1429B0    DEW      R0 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142740    DEW      R0 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142560    DEW      80 00    1140C0   001AC5CE  0015F6R8                           IXFMON01+2756     OVRLAPIO
142B70    DEW      R0 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
1425R0    DEW      80 00    1140C0   001AC5CF  0015F6R8                           IXFMON01+2756     OVRLAPIO
142B10    DEW      80 00    1140C0   001AC5CE  0015F6R8                           IXFMON01+2756     OVRLAPIO
142900    DEW      R0 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142BE0    DFW      80 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142A60    DEW      R0 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142870    DEW      80 00    1140C0   001AC5CE  0015F6R8                           IXFMON01+2756     OVRLAPIO
142R40    DEW      80 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142500    DEW      R0 00    1140C0   001AC5CF  0015E6R8                           IXFMON01+2756     OVRLAPIO
142600    DEW      80 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
1425R0    DEW      80 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142AD0    DEW      80 00    1140C0   001AC5CE  0015E6R8                           IXFMON01+2756     OVRLAPIO
142B90    DET      R0 00    26494E   0012C438  00127D50              18:35:10.24  BTAMSIM+A50
142650    DEW      80 00    127R50   0012FE62  001561ER                           BLHIN+18E2
142C20    DET      R0 00    264952   00117C76  00156CCR              18:35:10.35  GFDRIVER+E6
142AA0    DET      R0 00    264955   0012C438  0012R0C0              18:35:10.43  BTAMSIM+A50
142910    DEW      80 00    128DC0   0013A25E  00156AFR                           BSCLEASE+8AE
142A90    DET      R0 00    26495A   0012C438  00127DE0              18:35:10.56  BTAMSIM+A50
142570    DEW      R0 00    127DE0   0013152C  00156RE0                           BLHOT+E7C
142R00    DE       R0 00             0012F604  00127DE0                           BLHIN+84          BLMIN000
142C50    DET      R0 00    26497C   0012C380  00169B18              18:35:11.47  BTAMSIM+998
142B50    DET      R0 00    2649R0   0012C3R0  0016A55R              18:35:11.57  BTAMSIM+998
1427C0    DET      R0 00    264980   0012C3R0  0016AF9R              18:35:11.57  BTAMSIM+998
142A70    DET      R0 00    2649R7   0012C3R0  0016B4B8              18:35:11.76  BTAMSIM+998
142B30    DLT      R0 00    26498B   0012C91F  0016C41R              18:35:11.87  BTAMSIM+F36
1429C0    DET      R0 00    264997   0012C438  00127BA0              18:35:12.19  BTAMSIM+A50
142510    DEW      R0 00    1278A0   0013152C  00156018                           BLHOT+E7C
142C30    DEIW     00 00    1383AC   00132D30  00154770                           BMH000+B50
1425E0    DET      R0 00    2649AR   0012C3R0  0016C938              18:35:12.72  BTAMSIM+998
1428C0    DET      R0 00    2649AE   0012C3R0  0016CE58              18:35:12.80  BTAMSIM+998
142920    DEW      R0 00    1528EB   001AC5CF  0015E6R8                           IXFMON01+2756     OVRLAPIO
142B10    DET      00 00    2649RA   0012C43R  00127ACR              18:35:13.12  BTAMSIM+A50
142BE0    DEW      R0 00    127ACR   0012FE62  00156DR0                           BLHIN+18E2
142640    DE       00 00             00131FF4  00152C60                           BMH000+4          BMHIN000
142A40    DET      00 00    2649CF   00147336  0016004R              18:35:13.65  SYCTRL+1246       PURGE
```

Figure 4-2. Sample IJKTRACE Listing (Page 1 of 3)

```
WAIT LIST    WQT 1424BC   FIRST 142BA0   LAST 1427A0   COUNT 000A   HI/LO 001A

 WQE      FLAGS    PRI OVL  ECB/T    ENTRY PT   PARAMETER  (ECB)      TIME        CSECT              ENTRY     SUBSYS   SUB NAME
142BA0    W         80 00   12E5C4   00132430   00142C90   005DC06A                DMH000+450         DMHDEQAB
142BA0    W         80 00   12C05C   0012C9F6   4E12C9E0   005DC06A                DTAMSIM+100E
142BD0    W         80 00   12B194   0013AACC   0013AC10   005DC06A                CNTDIMOD+2DC       ECBWAIT
142620    IW        00 00   18CA20   0017B930   00000000   00000000                PMISTUP+318
142540    IW        00 00   13B46C   00132B30   00154DB0   03000000                DMH000+B50
142930    IW        00 00   13B26C   00132B30   00154E58   3F127CF8                DMH000+B50
142C00    IW        00 00   13B12C   00132B30   00155108   001277E0                DMH000+B50
1427A0    IW        00 00   1383AC   00132B30   00154770   03127928                DMH000+B50

TIME LIST    WQT 1424A8   FIRST 142B60   LAST 142AF0   COUNT 0031   HI/LO 0037

 WQE      FLAGS    PRI OVL  ECB/T    ENTRY PT   PARAMETER  (ECB)      TIME        CSECT              ENTRY     SUBSYS   SUB NAME
142B60    T         80 00   264A0C   0012E604   00127D50                18:35:15.31 BLHIN+A4           BLHIN000
142BF0    T         80 00   264A10   0012E604   001274F0                18:35:15.41 BLHIN+A4           BLHIN000
142B30    T         80 00   264A13   0012C3B0   0016BEF8                18:35:15.49 BTAMSIM+998
142B90    T         80 00   264A1B   0012C3B0   0016B9D8                18:35:15.63 BTAMSIM+998
142A00    T         80 00   264A3B   0012C3B0   00169B1A                18:35:16.56 BTAMSIM+998
142530    T         80 00   264A3E   0012C3B0   0016A55A                18:35:16.64 BTAMSIM+998
142BD0    T         80 00   264A3E   0012C380   0016AF98                18:35:16.64 BTAMSIM+998
142BB0    T         80 00   264A45   0012C3B0   0016B4BA                18:35:16.83 BTAMSIM+998
1427E0    T         80 00   264A59   0012C91E   0016C418                18:35:17.36 BTAMSIM+F36
1426B0    T         80 00   264A69   0012C3B0   0016C938                18:35:17.79 BTAMSIM+998
1427F0    T         80 00   264A79   0012C3B0   0016CE58                18:35:18.21 BTAMSIM+998
1425C0    T         80 00   264A84   0012C91E   001695FA                18:35:18.51 BTAMSIM+F36
142AC0    T         00 00   264AA0   0017A002   00177438                18:35:19.25 DYNLLOAD+32A
1427B0    T         80 00   264ACA   0012C91E   0016AA78                18:35:20.37 BTAMSIM+F36
142B00    T         80 00   264ACD   0012CA0F   00127CF8                18:35:20.45 BTAMSIM+1026
142C60    T         80 00   264B35   00131AB2   07127B50                18:35:23.23 BLHOT+1402         HARDBACK
142AB0    T         00 00   264C1A   0014C9B2   0014D7B4                18:35:29.33 RSMGMNT+362
142660    T         00 0A   264E6B   02118ED4   00153D90                18:35:45.15 TEST3270+10C                3/00F3   TEST3270
142BF0    T         00 00   265B49   001B24F0   00157098                18:37:12.99 CONVERSE+C2B
142A50    T         00 00   2660CF   001B1F20   00000000                18:37:50.67 CHECKPT
142B00    T         00 0A   266BB7   00147336   0015FD98                18:39:05.15 SYCTRL+1246        PURGE
142500    T         00 00   26719D   0014C6F6   0018C6A0                18:39:45.41 RSMGMNT+A6
1429B0    T         80 00   2683CC   001391A2   00127FB0                18:41:49.55 BSCDIAL+96A        BSCDLOUT
1429A0    T         80 00   2683CC   00139182   00128008                18:41:49.55 BSCDIAL+96A        BSCDLOUT
142B50    T         80 00   268D1C   00139182   00127FB0                18:42:53.12 BSCDIAL+96A        BSCDLOUT
142A30    T         40 00   26E4F0   0014B1D0   0017B9C0                18:52:52.69 TRAFFIC
142BC0    T         00 00   26ED29   00141570   00000000                18:53:48.83 INTSTS
142950    T         00 00   26F036   00148410   0018EF4E                18:54:09.65 PMIAUTOF
142610    T         00 00   26F654   00148410   0018EF8A                18:54:51.41 PMIAUTOF
142B20    T         00 00   279D26   0018A8A0   0002BF20                19:13:50.24 IXFRPT01
142AF0    T         00 00   2E8637   03120C3A   001537E0                22:35:07.07 WAGEMNGR+70A                JP/D1D7  WAGEMNGR

EXEC LIST    WQT 1424B4   FIRST 142880   LAST 142880   COUNT 0001   HI/LO 0026

 WQE      FLAGS    PRI OVL  ECB/T    ENTRY PT   PARAMETER  (ECB)      TIME        CSECT              ENTRY     SUBSYS   SUB NAME
142880    E         00 00            00141FA2   0083076F                            IJKDSP01+53A       IJKRETX

EXEC LIST    WQT 1424C0   FIRST 1424C0   LAST 1424C0   COUNT 0000   HI/LO 0002

EXEC LIST    WQT 1424CC   FIRST 1424CC   LAST 1424CC   COUNT 0000   HI/LO 0016
```

Figure 4-2. Sample IJKTRACE Listing (Page 2 of 3)

```
EXEC LIST     WQT 1424D8    FIRST 1424DP   LAST 1424D8   COUNT 0000   HI/LO 0001

IPOST LIST    WQT           FIRST          LAST          COUNT 0026   HI/LO 0030

 WQE      FLAGS    PRI OVL  ECB/T    ENTRY PT  PARAMETER  (ECB)      TIME    CSECT          ENTRY    SUBSYS  SUB NAME
142400    PW        40 00   18C914   00178CAC  0017924C   001424F0           DELOAD+44
142520    PW        00 00   18D6E4   001460F0  0018D684   00142520           SYCTRL         SCNMAIN  PG/D7C7 MMUTESTM
142550    PW        00 00   18CEB0   001460F0  0018CE50   00142550           SYCTRL         SCNMAIN   N/00D5 PMIOUTPT
142630    PW        00 00   18CDE8   001460F0  0018CD88   00142630           SYCTRL         SCNMAIN   U/00E4 PMIOUTPT
142670    PW        00 00   18CE4C   001460F0  0018CDEC   00142670           SYCTRL         SCNMAIN   V/00E5 PMIOUTPT
142690    PW        00 00   18CF78   001460F0  0018CF18   00142690           SYCTRL         SCNMAIN  LL/D3D3 LOADSCT
1426A0    PW        00 00   18D040   001460F0  0018CFE0   001426A0           SYCTRL         SCNMAIN  GP/C7D7 GPSS
1426D0    PW        00 00   18D108   001460F0  0018D0A8   001426D0           SYCTRL         SCNMAIN   P/00D7 PAGEMSG
1426E0    PW        00 00   18D16C   001460F0  0018D10C   001426E0           SYCTRL         SCNMAIN   W/00E6 SENDBACK
142700    PW        00 00   18D234   001460F0  0018D1D4   00142700           SYCTRL         SCNMAIN   H/00C8 CHANGE
142710    PW        00 00   18D298   001460F0  0018D238   00142710           SYCTRL         SCNMAIN  HH/C8C8 **NONE**
142720    PW        00 00   18D2FC   001460F0  0018D29C   00142720           SYCTRL         SCNMAIN  CC/C3C3 **NONE**
142730    PW        00 00   18D360   001460F0  0018D300   00142730           SYCTRL         SCNMAIN  LM/D3D4 MISSING
142750    PW        00 00   18D428   001460F0  0018D3C8   00142750           SYCTRL         SCNMAIN  L1/D3F1 FNTESTL
142760    PW        00 00   18D48C   001460F0  0018D42C   00142760           SYCTRL         SCNMAIN  L2/D3F2 SNBKL1
142770    PW        00 00   18D4F0   001460F0  0018D490   00142770           SYCTRL         SCNMAIN  L3/D3F3 SNBKL2
142780    PW        00 00   18D554   001460F0  0018D4F4   00142780           SYCTRL         SCNMAIN  L4/D3F4 CONV1L
142790    PW        00 00   18D5B8   001460F0  0018D558   00142790           SYCTRL         SCNMAIN  MD/D4C4 TESTRLSE
1428D0    PW        00 02   18D7AC   001460F0  0018D74C   001428D0           SYCTRL         SCNMAIN  C1/C3F1 MONOVLY
142940    PW        00 00   18CFDC   001460F0  0018CF7C   00142940           SYCTRL         SCNMAIN  MM/D4D4 MMUCOMM
142990    PW        00 00   18D1D8   001460F0  0018D170   00142990           SYCTRL         SCNMAIN   R/00D9 IXFRPTIO
1429E0    PW        C0 00   18CF14   001460F0  0018CEB4   001429E0           SYCTRL         SCNMAIN   J/00D1 PMICLDWN
142A10    PW        00 00   18D3C4   001460F0  0018D364   00142A10           SYCTRL         SCNMAIN  RQ/D9D8 SQCOBOLA
142A20    PW        00 00   18D680   001460F0  0018D620   00142A20           SYCTRL         SCNMAIN   Q/00D8 CHCKPTSS
142B80    PW        40 01   18D748   001460F0  0018D6E8   00142B80           SYCTRL         SCNMAIN   M/00D4 MONOVLY
142C10    PW        00 00   18D0A4   001460F0  0018D044   00142C10           SYCTRL         SCNMAIN   B/00C2 SWITCH

IJKTRACE ENDED.
```

Figure 4-2. Sample IJKTRACE Listing (Page 3 of 3)

When all Dispatcher lists have been scanned and formatted, the following line is generated:

IJKTRACE ENDED.

Control is retained in the current program path for the duration of processing by this module; the Dispatcher is not entered, and no other system work is performed.

4.2.4 IJKCESD--Initialize Csect/Entry Tables

IJKCESD is called once during system startup to scan the main Intercomm load module and to scan LPSPA (if the Intercomm Link Pack facility is used) in order to create the internal tables used to provide the Csect and Entry names for the IJKTRACE report and various Intercomm debugging messages, snap printouts, and the Resource Management Thread Dump. IJKCESD may be resident or in the startup overlay (conditionally called by the STUOVLY Csect). It is automatically included if the ICOMLINK macro is used to generate the Intercomm linkedit.

If an LPSPA linkedit is used (placing selected Intercomm load modules in the Link Pack Area as described in Chapter 7), then a DD statement for the load library containing the LPSPA load module must be added to the Intercomm execution JCL after the //PMISTOP DD DUMMY statement (library not processed via the File Handler), as follows:

```
//LPSPALIB  DD  DISP=SHR,DSN=LPSPA-load-module-library
```

4.2.5 IJKWHOIT--Find Csect/Entry and Subsystem Names

IJKWHOIT is called by several Intercomm system modules to determine the Csect name, and displacement within that Csect, of an address passed as a parameter. It may also be called to find out the name of the subsystem for which the SCT entry address is passed as a parameter. Note that the SCT entry address is the third parameter passed to all subsystems on transfer of control from Intercomm. IJKWHOIT must be included in the Intercomm linkedit as resident (automatic if the ICOMLINK macro is used to generate the Intercomm linkedit).

To find the name of (and displacement within) a Csect in which an address in the Intercomm (or LPSPA, if there) or dynamically loaded load module resides, call IJKWHOIT as follows:

```
CALL IJKWHOIT,({addr},{sct},wherecsect,{wherentry},{wheresub}),
               { 0 }                    {   0   } {   0   }
     VL(,MF=(E,list))
```

where:

addr — is a pointer to the field containing the address whose Csect name is to be found (if only SCT desired, code 0--see below);

sct — is a pointer to the SCT (SYCTTBL) entry for a subsystem (if not desired/available, code 0);

wherecsect — is a pointer to the area to which the caller wants the Csect name moved (a print line, for example): minimum area length must be 13 bytes for the Csect name plus displacement, if any (if the Csect name cannot be found, the value UNKNOWN ADDR is placed in the area) (if only SCT passed, code 0);

wherentry — is a pointer to the area to which the caller wants the entry point name (if available within the Csect) moved: minimum area length must be 8 bytes (if not desired, code 0);

wheresub — is a pointer to the area to which the caller wants the name of the subsystem (if sct pointer coded) moved: minimum area length is 8 bytes. If a subsystem defined as resident or overlay is not included in the linkedit, the value **NONE** is placed in the area. (If sct is not coded, code 0).

To obtain only a subsystem name, use the following form of the call:

```
CALL IJKWHOIT,(0,{sct},0,0,wheresub),VL[,MF=(E,list)]
               {(r)}
```

where (r) is a register pointer to the SCT entry.

Return Codes: 0 - address(es) converted and required information moved to user area(s);
4 - either address not found, or IJKCESD was not in the Intercomm linkedit, or an error encountered at startup - no CESD table entries were formatted.

4.2.6 IJKDELAY--Request Time Delay

This module may be called, instead of using the DISPATCH or INTWAIT macros for a timed wait, to introduce a timed delay averaging 100 milliseconds into a program path. The Dispatcher is given control to perform other processing and returns at the expiration of the delay interval. No parameters are passed. Standard linkage conventions are used. The current thread will resume processing, after expiration of the interval, with the same execution priority. There is no REENTSBS code; a SUBMODS must be added for the routine if it is not called by an Assembler Language program.

The facility may be utilized to give a time-slicing effect within a routine that would otherwise monopolize CPU time. It can also force the buildup of parallel program paths for reentrant testing purposes in an environment where actual parallel execution otherwise might not ensue, or it may be invoked to await the passing of a temporary condition that is to be resolved by another previously scheduled program.

4.2.7 IJKTLOOP--Trace Program Loop

This module assists in detecting closed program loops. If it is included in the Intercomm linkedit, it will be activated automatically at system startup. IJKTLOOP functions as an Intercomm subtask. When IJKTLOOP is called at startup, a subtask is ATTACHed, followed by a CHAP (change priority request) in the Intercomm main task giving the subtask the highest priority in the Intercomm region. The subtask:

- Initializes flags in the Intercomm Dispatcher
- Issues a STIMER to schedule an exit routine, then
- WAITs on an ECB to be posted by that exit routine.

After 30 seconds (real time), the exit routine receives control and posts the ECB placing the subtask in the ready state. When the subtask receives control, it checks flags in the Dispatcher to determine whether various conditions have occurred and to take the appropriate actions as follows:

- If closed loop detection has been deactivated via a call to IJKTSTOP (see below), the closed loop subtask is DETACHed by the Intercomm main task and closed loop processing is no longer operative.
- If the Intercomm main task is in the WAIT state, then the STIMER is reissued to schedule the exit routine and the subtask WAITs again without taking further action.
- If the Dispatcher has been entered, indicating that a task has been scheduled in the intervening 30 seconds (that is, the task that was executing at the start of the 30-second interval has returned control to the Dispatcher and thus was not in a long duration closed loop), then the Dispatcher-entered flag is cleared (flag will be reset by the Intercomm Dispatcher in the main task). The exit routine is then rescheduled and a WAIT is performed as before.

- If none of the above conditions are true, the subtask returns to the main task, which issues the message numbered MP020I and abends with a user code of 909, accompanied by a snap with ID=121, an IJKTRACE printout and a thread dump. The abend 909 will be recovered by STAEEXIT (if included in the Intercomm linkedit), which cancels the looping thread, issues message MP003I, and then transfers control to the retry routine, STAERTRY, if it is included in the Intercomm linkedit. The retry routine will call IJKTLOOP to reactivate the closed loop detector and then restore the Intercomm environment (via transfer of control to SPIESNAP at entry ABNDCANC).

Closed loop detection may be deactivated at any time via a call to IJKTSTOP, an entry in IJKTLOOP. No parameters are required; standard linkage conventions are followed. This may be useful if, for example, a program thread requires control, or calls an Intercomm routine (for example, the File Handler) that requires control, for a longer than average duration before returning to the Dispatcher. Once the closed loop detector, IJKTLOOP, is deactivated via IJKTSTOP, it must be reactivated to reinstate closed loop detection. Intercomm will not reinstate it automatically unless a 909 abend occurs.

Closed loop detection is reactivated via a call to IJKTLOOP. No parameters are required; standard linkage conventions are followed. If IJKTLOOP is called and closed loop detection is already active, a return code of X'04' is returned in register 15 to the caller without any further action taken.

NOTE: The hard-coded interval for the scheduling of the exit is 30 seconds real time, not task time. This means that the time is decremented continuously whether Intercomm has control of the CPU or not. This should be taken into account if Intercomm runs on the system with other higher priority jobs.

To summarize, IJKTLOOP processing requires inclusion in the Intercomm linkedit of IJKTLOOP, STAEEXIT, STAERTRY, SPIEEXIT and SPIESNAP, in addition to IJKTRACE, IJKCESD, IJKWHOIT, IJKPRINT and TDUMP (and the DD statements for SYSPRINT, SMLOG, SNAPDD and optionally LPSPALIB). When generating the Intercomm linkedit via the ICOMLINK macro, code LOOPTIM=YES. Also see Chapter 8 for further details on snap processing and the description of snap 121 in Messages and Codes.

Chapter 5

RESOURCE MANAGEMENT

5.1 INTRODUCTION

Intercomm Resource Management has three major options:

1. Resource Auditing and Purging
2. User-defined pools of core storage
3. Accumulation of core-use statistics

All or any combination of these three options can be selected by the user, according to installation requirements. If only the pools option (recommended) is selected, Resource Management still provides the system with an extremely efficient version of storage management. Macros and their parameters referenced in this section are described in Basic System Macros.

5.2 RESOURCE AUDITING AND PURGING

Resource Auditing refers to the maintenance of a chain of resource control blocks (RCBs) defining user-accessed resources for every active thread. There are five audited resource types:

1. CORE--acquisition of storage by invoking the STORAGE macro
2. FILE--use of a data set indicated by a call to SELECT
3. DDQ--access to a dynamic data queue indicated by a call to QBUILD or QOPEN
4. DYNL--loading of a dynamically loaded subroutine via invoking the MODCNTRL macro by the user, COBREENT, PMIPL1 or LOADSCT
5. NQ--activating an enqueue upon a resource by issuing the INTENQ macro

Each time a thread acquires a resource, a control block is created containing information about the resource and is attached to a chain of similar blocks. When the thread releases control of the resource, the corresponding control block is detached from the chain. The on-line TDUMP utility (see Section 5.9) is provided to print out the control block chains. This output shows which thread was in control, what resources each thread owned, which module acquired each resource, and the order of acquisition.

Resource Purging means that when a thread completes, normally or abnormally, its chain of resource control blocks is checked; in the case of a non-empty chain, the used control blocks are released after freeing blocks of storage, releasing files, etc.

All levels of Resource Management will purge Dispatcher queue entries for failed message processing threads. With Resource Auditing, storage, files, DDQs, loaded subroutines and enqueued resources are also purged. Additionally, a "must complete" disable/enable facility ensures that threads are not purged during critical operations; that is, if a subsystem times out while an I/O event is outstanding, a timed wait for the I/O event to complete is effected before attempting the purge.

5.3 USER-DEFINED STORAGE POOLS

User-defined storage pools are generated by the Intercomm ICOMPOOL macro and may be dynamically loaded at startup or linkedited into the Intercomm load module. A pool is a set of storage blocks of a given size; there is no limit to the number of blocks in a pool. The ICOMPOOL macro also generates an index that permits the storage management routine to quickly determine whether or not a storage request can be filled out of the pools. Freeing an area of pool storage is usually just as fast. Furthermore, the code is loop-free, so that these time values are constant, and system degradation due to storage fragmentation does not occur. The increase in efficiency provided by judiciously tailored Intercomm pools more than offsets any overhead increment from core-use statistics gathering. Creation of the user-defined Intercomm pools (via ICOMPOOL macro) is described later in this chapter. Acquiring and releasing core under Intercomm is accomplished via the STORAGE and STORFREE macros described in Basic System Macros.

5.4 CORE-USE STATISTICS

Three sets of core-use statistics can be accumulated via the RMTRACE routine. Statistics are computed and printed at intervals defined in SPALIST macro parameters.

1. Global statistics--the number of STORAGE and STORFREE macros issued, the average storage request length, the number of requests filled from the pools, etc.

2. Breakdown of STORAGE requests into detailed user-defined core block size ranges. For each range, the number of requests falling into that range is given, plus "concurrency" statistics: at any given moment, the concurrency of a range is the number of blocks that have been obtained, but not freed. In addition to the instantaneous concurrency, high, low and average concurrencies are computed. These figures are particularly useful in working out pool sizes; the most

value from a pool is obtained if the block size falls in a range with a large number of requests, and the average concurrency of the range indicates how many blocks are needed in the pool. However, if the size is small, the high concurrency may be used to get maximum efficiency, at a relatively low cost in storage.

3. Pool-use detail statistics measure the effects of different choices of pools, providing such information as the number of requests that could not be filled from the user-defined pool (because all the blocks were in use), the average number of free blocks, etc.

5.5 STORAGE CUSHION

Every version of Resource Management includes the Storage Cushion feature. At startup, a block of storage is obtained and held until a request arrives that cannot be satisfied out of the Intercomm pools or dynamic storage (OS subpool area). The storage cushion is then released and no new threads started until the cushion is available again. Thus, a temporary shortage of storage is not likely to bring the system down. The user specifies the size of the cushion in the SPALIST macro CUSHION parameter; a zero size is acceptable. A WTO informs the user whenever release and acquisition of the cushion occurs. (Front End input operations are also temporarily halted if the module SSPOLL is included--see Chapter 7.)

5.6 RESOURCE MANAGEMENT MODULES AND GLOBALS

Seven modules automatically included in the Intercomm linkedit are used to support Resource Management. Their member names are MANAGER (Csects: RSMGMNT, RMPC and RMFNQ), RMPURGE, RMTRACE, TDUMP, POOLDUMP, RMNADISA and the core pools definition module.

- MANAGER is the main Resource Management module. It contains entry points for STORAGE and STORFREE macro processing (STORAGEM and STORFRED), routines that switch control of blocks of storage between threads (RMPASS and RMCATCH), and those that handle resource control blocks for files (RMFON/OFF), enqueued resources (RMNQON/OFF), etc.

- RMPURGE is the Resource Purging routine. It is called by the Subsystem Controller when a nonzero thread completes to free any resources not previously freed by the thread.

- RMTRACE computes and prints out core-use statistics. (See Figure 5-2 for explanation and sample output.)

- TDUMP prints out RCB chains. (See Figure 5-3.)

- POOLDUMP prints out the current status of the user pools. (See Figure 5-4.)
- RMNADISA is the Intercomm disable/enable routine, and is also used for resource purging.
- NEWPOOLS (or user-defined name) contains ICOMPOOL macros defining storage pools.

Four independent options apply to Resource Management, and are defined by binary set symbols in INTGLOBE and set in SETGLOBE, controlling assembly of the MANAGER module. These options are as follows:

1. &RM

   If set to 1, Resource Audit and Purge are obtained; it is necessary to include RMPURGE amd RMNADISA if this option is chosen. Also, TDUMP should be included.

2. &RMPOOLS

   If set to 1 (required), pool support is obtained; an ICOMPOOL module must be defined. POOLDUMP may be included.

3. &RMSTATS

   If set to 1, global core-use statistics are provided. RMTRACE must be included.

4. &RMACCT

   If set to 1, detail core block size and pool-use statistics are provided. RMTRACE must be included.

5.6.1 Obtaining a Save Area with Resource Management

The STORAGE macro has Resource Management parameters. Instead of a LINKAGE macro, STORAGE can be issued without supplying a save area or a parameter list by the coding of RENT=NO. (See Figure 5-1). The macro will generate code to build the list in MANAGER, and MANAGER will save registers in its own in-line save area. In fact, with Intercomm, the in-line save area is first used, shifting only to the user's save area when a storage request fails and a retry is necessary. Thus, coding RENT=NO means only one attempt is made to obtain user storage; however, the retry feature is not as likely to be invoked with the Storage Cushion facility in use, and less likely to succeed when it is invoked because it competes for storage with the routine that tries to reacquire the cushion. If a STORAGE request fails, an error routine may be given control as specified by the ERRADDR parameter. VS users can optionally specify page boundary alignment in the STORAGE macro.

The code in Figure 5-1 illustrates a save area obtained via a STORAGE macro.

```
*Register 15 is used by the STORAGE macro, as are 14, 0 and 1.  Thus,
*the user must establish a base register other than 15.

          ENTRY   SUB
          USING   SUB,Rz
SUB       STM     14,12,12(13)
          LR      Rz,R15

*Next, establish addressability to the SPA Csect.

          L       Rx,=V(SPA)

*Issue STORAGE macro to obtain storage for save area and set forward
*chain in current save area.

          STORAGE LEN=len,ADDR=8(13),SPA=(Rx),RENT=NO

*Test for valid return (ensure storage was obtained)

          LTR     15,15
          BNZ     error-routine

*Restore registers used by STORAGE (optional)

          LM      14,1,12(13)

*Initialize new save area

          L       Ry,8(13)          Get save area address
          ST      13,4(Ry)          Back chain
          LR      13,Ry             Point to new save area
```

NOTE: Rx, Ry and Rz refer to three general registers (2 to 12). They have the following uses:

- Rx points to the System Parameter Area (SPA).
- Ry temporarily holds the address of the storage obtained.
- Rz is the base register.

Figure 5-1. Obtaining a Save Area via the STORAGE Macro

The RTNLINK macro, SPA=(r) parameter, is used by Resource Management. RTNLINK generates a call to the PMIRTLR Csect, which in turn calls STORFRED to release the save area. If PMIRTLR finds its STORFRED VCON unresolved, it expects the SPA address in register 2. If a register has been specified as the SPALIST base in the preceding LINKAGE macro, RTNLINK will generate a LR of the base into register 2. In cases where a LINKAGE macro was not issued or the SPALIST base is no longer valid upon a return, the SPA address must be loaded into a register (r) and the SPA=(r) parameter must be coded on the RTNLINK macro.

5.7 INSTALLING RESOURCE MANAGEMENT WITH CORE-USE MONITORING AND POOLS

5.7.1 SETGLOBE Settings

The following globals must be defined in SETGLOBE:

```
&RMPOOLS    SETB    1   use Intercomm pools (required)
&RMSTATS    SETB    1   generate global core-use statistics
&RMACCT     SETB    1   generate detail usage statistics
```

and MANAGER must be reassembled.

An additional option implemented via the conditional assembly of MANAGER with the global &RMINTEG in SETGLOBE SETBd to 1, causes validation of the integrity of the storage pools on each entry to MANAGER. If the storage pool area is not intact, an error message (RM022A) is generated. This facility assists in detecting problems in destruction of storage, often difficult to find due to their random nature. This facility is controlled by the STRT/STOP system commands, and is set off at startup.

NOTE: This facility should be used in the test environment only, due to CPU overhead. See also the description of the TRAP debugging module in Messages and Codes.

5.7.2 SPALIST Parameters

Associated parameters in the SPALIST macro are described below. Other SPALIST parameters, not used at this level of Resource Management, are discussed in conjunction with Resource Auditing.

Choose appropriate values for these parameters and, if necessary, reassemble INTSPA (SPA and SPAEXT Csects).

CUSHION
is the size in bytes of a block of storage (specify in 2K (MFT/VS1) or 4K (MVT/MVS) increments) that will be acquired by a GETMAIN at startup and released when a request for main storage cannot be satisfied. When the cushion is released, the SPAHOLD switch is set so that no new threads are started, and a routine issuing a GETMAIN is dispatched on a time interval to get the cushion back. If unsuccessful, it leaves SPAHOLD set and redispatches itself. The default is 2048.

CUSHTM
is the interval in seconds between tries at getting the cushion back. The default is 1.

COREACC
is coded YES if computation of core block size statistics, broken down by ranges with pool "concurrencies", and pool-use detail statistics are desired. (See Figure 5-2.) The default is YES.

RMSTIM
is the time interval, in seconds, between successive invocations of the detailed pool usage statistics program (RMTRACE). The maximum value is 32,767 (9 hours, 6 minutes and 7 seconds). The default is 5 seconds.

TRACETM
is the interval, in seconds, between printouts of global (and detailed) core-use statistics by RMTRACE. The default is 120.

5.7.3 Defining the Intercomm pools (ICOMPOOL)

The ICOMPOOL macro is coded by the user to define each user pool area and has the following operands:

LEN
is the size of a pool block up to a maximum of 256K less 8 bytes.

NUMBER
is the number of blocks of that size.

LOWLIM
optionally specifies the minimum request size to be filled out of this pool.

For example, to define a pool of 20 16-byte blocks, code:

```
ICOMPOOL LEN=16,NUMBER=20
```

To define a second pool of 10 256-byte blocks, and to ensure that only requests for greater than 200 bytes (but less than or equal to 256) will be allocated from the pool, code:

```
ICOMPOOL LEN=256,NUMBER=10,LOWLIM=200
```

The number of bytes allocated from a pool block will always be greater than the block size of the preceding pool. LOWLIM is coded only when the difference in block sizes between successive pools is large and user intent is to reduce wastage. If LOWLIM were not coded in the above example, an infrequent 30-byte request could tie up an entire 256-byte block.

ICOMPOOL macros must be arranged by increasing block size; that is, the values of the LEN parameters have to be in ascending order. A maximum of 255 ICOMPOOL macros may be coded.

The following JCL can be used to create the pools member:

```
// EXEC   LIBE,Q=LIB
./ ADD NAME=member-name
./ NUMBER NEW1=1000,INCR=1000
ICOMINX CSECT
        ICOMPOOL macro 1
        .
        .
        .
        .
        .
        ICOMPOOL macro n
        END
```

Assemble the new member. One set of pools, member name NEWPOOLS, is included on the release tape. These pools are roughly sized to handle the storage requirements of the Intercomm beta test, and may be used as a starter set before core-use statistics have been collected.

The member may be linkedited with the Intercomm load module, or it may be chosen dynamically at startup if the dynamic core pool facility is in use. (If the latter, the pools may not be linkedited with the load module.) If the pool load module is to be selected dynamically, the member name must be ICPOOLxx where xx is a two-digit number 00-99. When dynamic pools are in use, a number of different sets of pool load modules can be created and the proper one will be loaded at startup, as described below.

5.7.3.1 Dynamically Loaded Core Pools

At startup time, the user may dynamically choose a set of storage pools for the system to use. That is, instead of choosing a set of storage pools at linkedit time, a set of pools may be chosen at execution time. The set of pools chosen is brought into core via a LOAD macro and, for every Intercomm execution, a new set or the same set of pools may be chosen. This option may prove advantageous if it is desired to experiment with different sets of core pools to find the most efficient, or if it is known that at certain times variations in system activity make a different set of pools more efficient than they would be normally. Also, in some operating systems, the size of load modules is restricted, making the use of Intercomm administered storage pools difficult. With dynamic core pools, because they are a separate load module, the need for relinks of the system for every tuning of the pools, and/or the problem of size restriction, can be alleviated.

To use dynamic core pools, the following must be done:

- Include the module POOLSTRT in the Intercomm linkedit
- Exclude NEWPOOLS or whatever member name currently contains the ICOMPOOL macros to define the user pool areas. (The ICOMLINK macro will generate the proper INCLUDE statements if DYNPOOL=YES is coded. If DYNPOOL=NO, an INCLUDE for NEWPOOLS is generated but not for POOLSTRT.)
- Assemble and link the set(s) of pools (created via ICOMPOOL macros) onto a library which will be part of the //STEPLIB concatenation for Intercomm execution. The member names for the pool load modules must be ICPOOLxx where xx is a two-digit decimal number 00-99.
- If the module POOLSTRT is present in the Intercomm load module, it will be called at startup time and it takes the following actions:

  1. Checks if the pools were linkedited in with the system. If so, no further action is taken and the linkedited pools will be the ones used in the run.

  2. If not 1), a WTOR is issued requesting a reply in the form of a two-digit number which is the suffix of the name of the desired pool load module (the xx in ICPOOLxx).

  3. A LOAD is attempted for ICPOOLxx. If found, the module is loaded and execution of startup is continued. If not found, or if the reply is invalid (not numeric), another WTOR is issued, giving the operator the choice of:

a) retrying (the first WTOR is reissued and the operator may reply with a different two-digit suffix)

b) continuing without pools (all storage for the run will be GETMAINed)

c) cancelling the run - a return to OS is effected with a step return code of 16. No dump is taken.

- Under a VS (MVS) operating system, do not ORDER the pool Csects (ICOMINX, ICOMCHN, POOLACCT, ICOMPOOL, POOLEND) if they are dynamically loaded.

- If the pools are defined in the Page Fixing table (FIXTABLE); they will be page fixed at startup, even if they are dynamically loaded.

If the pools are subsequently to be linked into the Intercomm load module, add an INCLUDE for the desired pools module (ICPOOLxx) to the linkedit deck before the system linkedit is executed. The INCLUDE for POOLSTRT does not have to be removed.

5.7.4 Specifying Core Block Detail Statistics

Core block detail statistics are specified by coding the COREACCT macro and determining the desired ranges, as described below. Additionally, the MANAGER module must be reassembled.

Initially, core block usage is broken down by ranges: the "number of requests" column of the printout (see Figure 5-2) is used to decide the pool block sizes; the "average concurrency" is used to decide the number of blocks per pool. The ranges are defined by a COREACCT macro in the MANAGER module. The COREACCT parameters are:

(range1,....,rangen)	positional
,FROM=start,TO=end,BY=increment	optional

There are two ways to define the ranges:

1. Explicitly, by naming the range limits in the first positional argument.

2. Implicitly, by giving a starting limit, an ending limit and an increment.

For example, to break down storage core requests into those under a thousand and those over, code:

```
COREACCT (1000)
```

The macro will generate a table with two entries, one for requests between 0 and 1000, and the second for requests greater than 1000. On the other hand, to obtain a breakdown by 256-byte ranges up to 4128 bytes, the following code would be used:

```
COREACCT ,FROM=256,TO=4128,BY=256
```

These approaches may be mixed; the explicit limits have to be written in ascending order, but they do not have to be less than the implicit limits. The user may thus code:

```
COREACCT (504),FROM=32,TO=1600,BY=32
```

and the range from 480 to 512 would be broken down further into ranges 480-504 and 505-512.

In the MANAGER module as released, the macro is written:

```
COREACCT ,FROM=64,TO=4096,BY=64
```

The COREACCT macro also generates a routine FINDBUCK for finding the table entry corresponding to a storage request size. The code is optimal; for the second example with the 256-byte ranges, the routine would merely consist of a shift to divide the request size by 256, another shift to multiply the result by the table entry length, and an add to the table address. For this reason, explicit ranges should be excluded and the increment should be a power of two, if statistics are to be collected during a production run where efficiency is important.

5.7.5 Linkedit

The following modules must be included in the Intercomm Linkedit:

- MANAGER--storage management routine (reassemble after SETGLOBE updated)
- RMTRACE--statistics-gathering routine
- NEWPOOLS or a user-defined ICOMPOOLs member--user pools (unless dynamically loaded at startup)
- INTSPA--reassembled SPA and SPA Extension
- POOLSTRT--if pools are to be dynamically loaded

5.7.6 Execution

In the execution step, include the following DD statement for the data set that will receive the statistics:

```
//SMLOG    DD   SYSOUT=A,
//  DCB=(DSORG=PS,LRECL=120,BLKSIZE=120,RECFM=FBA)
```

For efficiency, BLKSIZE may be increased to a multiple of 120.

To eliminate core-use monitoring, change SETGLOBE so that &RMACCT and &RMSTATS are 0, reassemble MANAGER, and take RMTRACE out of the linkedit.

To keep the global statistics, reassemble the SPA with COREACC=NO and/or change SETGLOBE so that &RMACCT is 0 and reassemble MANAGER.

5.7.7 Sample Output

Figure 5-2 provides a sample output of core-use statistics. The following should be noted:

- CORE USE STATISTICS

  Except for TOTAL POOL STORAGE, POOL STORAGE AVAILABLE and BYTES OUTSTANDING, the figures are cumulative global statistics, accounting for all Storage Management activity from the beginning of the run.

- TOTAL ICOMPOOL WASTAGE

  Wastage is the difference between the length of the pool block and the length of the requested area allocated from the block; available blocks are not wastage. PERCENT WASTAGE is important; a low figure is desirable. Wastage is controlled by the LOWLIM parameter in the ICOMPOOL macro. Wastage is broken down by pool in the Pool Use Detail Statistics.

- ICOMPOOL FAILURES

  A count of the number of times a request failed from one of the pools because all the blocks in the pool were in use. A high figure means that at least one of the pools should have more blocks. Failures are broken down by pool in the Pool Use Detail Statistics.

- QUICK FREES

  This applies only to areas allocated from the pools: "quick" means no search was made to find the block containing the area to be freed; that is, the address passed pointed to the beginning of a pool block, and 8 is subtracted to get the pool block header. Most of Resource Management's overhead is in STORFRED's search loops, so a higher quick frees value is better.

- AVERAGE SEARCH LENGTH

  For Resource Auditing, this gives the average number of RCBs that STORFRED used to find the one corresponding to the area being freed, when it could not do a quick free. Without Resource Auditing, this is the average number of pool blocks STORFRED checked to find the one containing the area being freed.

- RCB TABLE RELOCATIONS

  When the RCB table is full, and an attempt is made to allocate an additional RCB, space is obtained to contain the current RCB table plus the number of RCBs to add as specified by the SPALIST macro parameter RCBSADD. (See Section 5.8.2.) This statistic shows the number of times this occurred. More than one relocation is undesirable.

- POOL USE DETAIL STATISTICS--AVG FREE BLOCKS

  This is the average number of blocks available for allocation. If this figure is low, relative to the number of blocks in the pool, then failures are usually high, and vice versa.

NOTE: Headings denoting DOUBLEWORDS indicate that the calculation is in doublewords: multiply by eight to get the corresponding value in bytes. All storage requests are rounded up to the next highest doubleword.

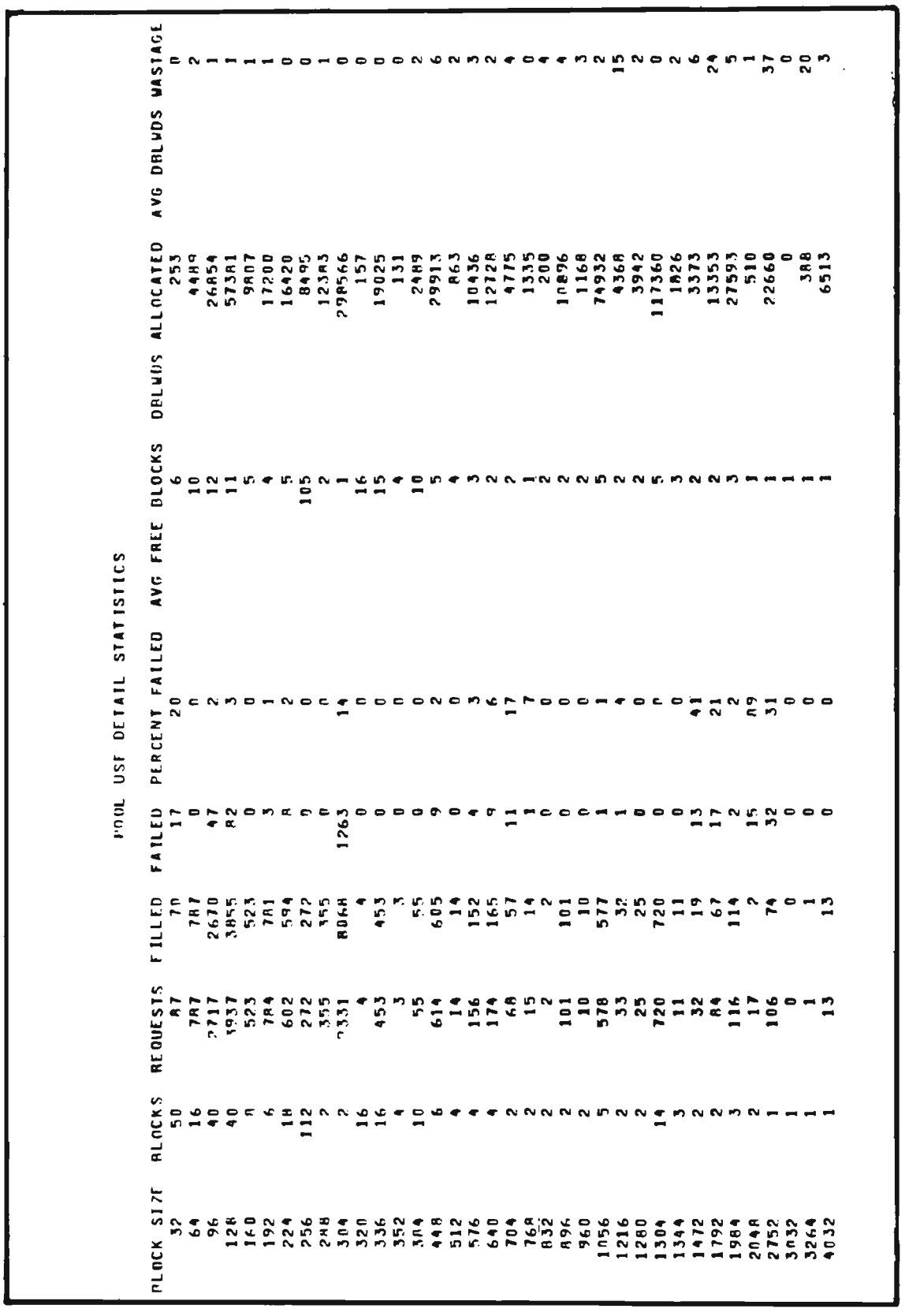
CORE USE STATISTICS TIME 10.39.45 83.076

STORAGES ISSUED	22802				
DOUBLE WORDS REQUESTED	896970	AVERAGE REQUEST LENGTH	40		
DOUBLE WORDS GRANTED	896970	HIGH THIS PERIOD	12495	HIGH THIS RUN	12495
TOTAL POOL STORAGE	143784	POOL STORAGE AVAILABLE	96680	PERCENT AVAILABLE	68
REQUESTS FILLED FM ICOMPOOL	21265	PERCENTAGE FM ICOMPOOL	94		
DOUBLE WDS GRANTED FM ICOMPOOL	822782	PERCENTAGE FM ICOMPOOL	92	AVERAGE LENGTH FM ICOMPOOL	39
DOUBLE WDS WASTED IN ICOMPOOL	35316	AVERAGE DOUBLE WDS WASTED	2	PERCENT WASTAGE	5
ICOMPOOL FAILURES	1535	PERCENT FAILURES	7		
STORFREES ISSUED	22751				
POOL BLOCKS FREED	21223	QUICK FREES	17988	PERCENTAGE QUICK	85
DOUBLE WORDS FREED	887997	DOUBLE WORDS OUTSTANDING	8973		
REQUESTS NOT FILLED	0	PERCENT NOT FILLED	0		
AVERAGE SEARCH LENGTH	290				
RCB TABLE RELOCATIONS	0				

DISTRIBUTION OF CORE BLOCK SIZES

RANGE	NUMBER OF REQUESTS	CONCURRENCY-- NOW	HIGH	LOW	AVERAGE
1- 64	874	61	68	10	53
65- 128	6654	57	89	4	58
129- 192	1307	5	13	2	6
193- 256	874	21	27	0	20
257- 320	9690	9	29	1	8
321- 384	511	3	7	0	2
385- 448	614	2	5	0	2
449- 512	14	0	2	0	1
513- 576	156	2	7	1	2
577- 640	174	2	7	1	3
641- 704	68	2	3	0	1
705- 768	15	2	3	0	2
769- 832	2	0	1	0	0
833- 896	101	0	2	0	0
897- 960	10	0	1	0	0
961- 1024	29	0	1	0	1
1025- 1088	572	0	6	0	1
1089- 1152	3	0	1	0	0
1153- 1216	7	0	3	0	0
1217- 1280	25	0	2	0	0
1281- 1344	731	12	12	0	10
1345- 1408	7	0	1	0	0
1409- 1472	25	0	4	0	0
1473- 1536	43	0	3	0	0
1537- 1600	1	0	1	0	0
1601- 1664	2	0	1	0	0
1665- 1728	2	0	1	0	0
1729- 1792	36	0	4	0	1
1793- 1856	7	0	5	0	0
1857- 1920	5	1	2	1	1
1921- 1984	105	0	1	0	0
1985- 2048	17	3	5	3	3
2049- 2112	5	0	1	0	0
2113- 2176	1	0	1	0	0

Figure 5-2. Example of Core-Use Statistics (Page 1 of 2)



POOL USE DETAIL STATISTICS

BLOCK SIZE	BLOCKS	REQUESTS	FILLED	FAILED	PERCENT FAILED	AVG FREE BLOCKS	DBLWDS ALLOCATED	AVG DBLWDS WASTAGE
32	50	87	70	17	20	6	253	0
64	16	787	787	0	0	10	4489	2
96	40	2717	2670	47	2	12	26454	1
128	40	1937	1855	82	3	11	57381	1
160	8	523	523	0	0	5	9807	1
192	6	784	781	3	1	4	17200	1
224	18	602	594	8	2	5	16420	0
256	112	272	272	0	0	105	8495	0
288	2	355	355	0	0	2	12383	1
304	2	7331	8068	1263	14	1	298566	0
320	16	4	4	0	0	16	157	0
336	16	453	453	0	0	15	19025	0
352	4	3	3	0	0	4	131	0
384	10	55	55	0	0	10	2489	2
448	6	614	605	9	2	5	29913	6
512	4	14	14	0	0	4	863	2
576	4	156	152	4	3	3	10436	3
640	4	174	165	9	6	2	12728	2
704	2	68	57	11	17	2	4775	4
768	2	15	14	1	7	1	1335	0
832	2	2	2	0	0	2	200	4
896	2	101	101	0	0	2	10896	4
960	2	10	10	0	0	2	1168	3
1056	5	578	577	1	1	5	74932	2
1216	2	33	32	1	4	2	4368	15
1280	2	25	25	0	0	2	3942	2
1344	14	720	720	0	0	5	117360	0
1472	3	11	11	0	0	3	1826	2
1536	2	32	19	13	41	2	3373	6
1792	2	84	67	17	21	2	13353	24
1984	3	116	114	2	2	3	27593	5
2048	2	17	2	15	89	1	510	1
2752	1	106	74	32	31	1	22660	37
3032	1	0	0	0	0	1	0	0
3264	1	1	1	0	0	1	388	20
4032	1	13	13	0	0	1	6513	3

Figure 5-2. Example of Core-Use Statistics (Page 2 of 2)

5.8 INSTALLING RESOURCE MANAGEMENT WITH RESOURCE AUDIT AND PURGE

Concurrency figures (see Figure 5-2) may be more accurate when using Resource Auditing. The difference is the way in which partial STORFREEs are recorded. For example, of 256 bytes, 16 bytes are freed; without Resource Auditing, there is no indication that the area being freed is part of a larger one. Thus, if the concurrency for the 16-byte range is decremented, then the concurrency for 16 is one too low and the concurrency for 256 is one too high. A subsequent STORFREE for the remaining area will make the concurrency for 240 inaccurate as well. With Resource Auditing, the RCB is available to indicate that the area is part of a 256-byte block; the concurrency is decremented for the 256-byte range and a flag set in the RCB. This causes accounting for STORFREEs on this block to be skipped so the eventual freeing of the other 240 bytes will not affect the concurrencies. A few partial frees will not make a significant difference in the average concurrencies, the most important figures. The number of partial frees in the ranges corresponding to the pools can be estimated by looking at the percentage of "quick frees" in the global statistics; a partial free will cause at least one search. Other advantages and restrictions are described below.

5.8.1 SETGLOBE Settings

The following global must be set in SETGLOBE for Resource Audit and Purge:

```
&RM  SETB  1
```

5.8.2 SPALIST Parameters

In addition to the previously discussed parameters in Section 5.7, there are two SPALIST macro parameters applicable to Resource Audit and Purge.

- RCBSINT

  The initial number of RCBs. Although the RCBs are chained together, they occupy a single area of storage called the RCB table. This permits an efficient sequential scan of all the RCBs, minimizes storage fragmentation, and reduces the risk of useless page faults under VS. Space for the RCB table is obtained the first time STORAGEM is called; this parameter indicates how many entries should be created in the table. The default is 75.

NOTE: The RCB table also contains a pointer to the free-RCB chain and the 256-entry thread table, making its total length:

4 + (8*256) + 20*(number of RCBs)

- RCBSADD

  The number of fresh RCBs to add when space is depleted in the RCB table. When the available RCBs are exhausted, space is obtained for a new table sufficient to hold this many new RCBs, plus all the RCBs in the old table. The contents of the old table are moved and the storage it occupied is freed. The default is 5.

The area for the expanded RCB table is acquired via a GETMAIN for storage from the subpool area. If space for a new RCB table cannot be obtained, Intercomm will abend with a code of 1111. This can be avoided by making RCBSINT large enough so relocation of the RCB table is not necessary. One of the global statistics is the number of relocations (see Figure 5-2); use the figure from the last statistics printout to compute the right size for RCBSINT.

5.8.3 Macro Specifications

Installation of Resource Auditing mandates the following two rules for Assembler Language programs:

1. To pair STORAGE and STORFREE macros, and LINKAGE and RTNLINK macros. If a block of storage is obtained with a STORAGE and freed with a FREEMAIN, an abend will occur with an A0A if storage was obtained from OS dynamic storage, or a 30A if obtained from pools (under MVT) where pool storage is in subpool 255. If a pool block under MFT is FREEMAINed, RMPURGE will first mark the block free, then later reallocate it. But the new owner will destroy the OS free-queue chain field and thus eventually cause an abend with a 50A. Conversely, if a block is obtained with a GETMAIN and freed with a STORFREE, Resource Management will issue a RM013A message and program check. LINKAGE and RTNLINK both use Resource Management to get and free work areas, so the same remarks apply to a LINKAGE followed by a FREEMAIN or a GETMAIN followed by a RTNLINK. Of course, a LINKAGE can be followed by a STORFREE, etc. In other words do not use GETMAIN and FREEMAIN macros.

   An A0A may occur in STORFRED. This almost always means that a thread has issued a FREEMAIN for a block of storage obtained with a STORAGE or LINKAGE. The thread completes and there is still an RCB pointing to the freed area; RMPURGE

calls STORFRED to free it and an AOA results. The address of the block is in register 9. RMPURGE will issue a thread dump: look for an RCB belonging to the thread being purged, that is, SMLOG's owner, whose resource address matches register 9. The ACQUIRED BY field for that RCB will locate the module that obtained the storage. (See Figure 5-3.)

2. Care must be taken not to leave blocks of storage unfreed. In one sense, this rule is relaxed, since acquired storage will be freed automatically upon return to the Subsystem Controller. On the other hand, an area cannot be left to be picked up, used, and freed by another thread--passing areas between threads must be done explicitly. This forces shielding of the area from the purge routine by attaching its RCB to Intercomm's chain, then moving the RCB to the receiving thread's chain (performed automatically for message queuing) so it will be freed if the receiving thread completed abnormally.

   There are two ways to handle this: the RCB can be put on Intercomm's chain at the time the area is obtained, by coding SYS=YES in the STORAGE macro; however, if there is a chance of a program check or time-out before the receiving thread is informed where the area is, the area should be obtained in the normal way, and later its RCB should be switched onto the system chain. The PASS macro is used to do the switching:

   ```
   PASS LEN=length,ADDR=address and optionally ,SPAEXT=(r)
   ```

   Code the length and address exactly as for STORFREE. Programs not linkedited with MANAGER must set up a base register for the SPA Extension. In particular, Message Collection passes the area containing the message; this means that while it is usually safe to do a GETMAIN as long as it is paired with a FREEMAIN, storage always has to be obtained for a message with a STORAGE. If this is not done, a RM009A message and a program check will result because RMPASS will not find an RCB for the area.

   The receiving subsystem claims the area with the CATCH macro coded just like PASS:

   ```
   CATCH LEN=length,ADDR=address and optionally ,SPAEXT=(r)
   ```

   NOTE: if the SYS=YES parameter is coded on the STORAGE macro and the user wishes to free the block while its RCB is still attached to the system chain, SYS=YES should also be coded on the STORFREE macro. Otherwise, Resource Management will search for the RCB sequentially through all the RCBs in the table, which is inefficient.

5.8.4 Linkedit

MANAGER must be reassembled after SETGLOBE is updated. The SPA and SPA Extension must be reassembled if the RCB table size parameters are changed. The Intercomm linkedit must include MANAGER, RMNADISA, TDUMP and RMPURGE, plus whatever modules are needed to support any other Resource Management options chosen (see Section 5.6).

The MANAGER module supports full Resource Management. If pool statistics accounting is not required, reassemble with the appropriate SETGLOBE globals set to 0.

5.8.5 Enqueue-Dequeue Facility

In a multitasking on-line system it is sometimes necessary to serialize the use of a particular resource (main storage, data set, etc.) by allowing only one task at a time to "own" the resource.

It is also sometimes desirable to limit the number of concurrent users of a resource to some predetermined maximum. Both these facilities are provided by the Intercomm Enqueue-Dequeue routine (Csect name PMINQDEQ) through the use of the macros INTENQ and INTDEQ. All control is effected by a resource name of from one to fourty-four characters; hence all programs utilizing a particular resource must include enqueue/dequeue logic referencing the identical resource name and providing the identical length of that name (default=16). A time-out control prevents "runaway" exclusive control. The inclusion of PMINQDEQ in the linkedit is automatic, as it is a required Intercomm system routine. Resource Audit and Purge monitors the Enqueue-Dequeue facility.

The following example requests and subsequently releases exclusive control of the resource whose ID-address is RESOR, within the issuer's region only. The default time-out value from the SPALIST (NQTIM parameter) will be used.

```
INTENQ   RESOR
.
.
INTDEQ   RESOR
```

The example below requests that all other Intercomm regions be prevented from using the resource whose ID-address is in register 1. Also, up to five tasks within the issuer's region may share use of the resource. There will be no time-out protection. The SHARE parameter is not defined for the release request.

```
INTENQ (1),SHARE=5,SYSTEM=YES
.
.
INTDEQ (1),SYSTEM=YES
```

5.8.6 Thread Hung User Exit--IOEXIT

If a nonzero thread program checks or times out (TCTV or Enqueue time expires), and the thread is disabled, resource purging is suspended. A thread may be disabled because:

- the last action was a file I/O, Store/Fetch flush, message queuing, or message logging request which did not complete before the time-out.
- a dynamically loaded subroutine program checked or timed out, and this thread originally caused the load of the subroutine (that is, issued the first call/link).
- a Data Base access interface module disabled the thread before starting processing of the data base request.
- the thread is executing under the general or special subtask facility.

A thread is disabled from resource purging via an internal DISABLE macro, and subsequently enabled for purging via an internal ENABLE macro. At nonzero thread purge time, if an outstanding DISABLE exists, purge processing is halted for the TCTV time of the originating subsystem, or until all required ENABLEs are issued (whichever occurs first). If the TCTV wait time expires without all necessary ENABLEs, a user exit IOEXIT is called (by RMNADISA) if coded and included as resident in the Intercomm linkedit. Subsequently, a subsystem disabled message (RM016I) is issued, a thread dump is produced, and only enqueue waits and outstanding WQEs are purged.

At entry to IOEXIT, standard linkage conventions are used, with register 1 pointing to the SYCTTBL entry for the thread being purged. The user exit could be used to issue a PMIWTO to alert the operator at the control terminal that one of the above disable reasons could degrade Intercomm execution time, such as tying up access to the hung resource. Optionally, the subsytems accessing the hung resource could be delayed from new executions via the system control command DELY. Repeated occurrences of this situation could be cause to close down Intercomm until the problem is resolved via dump/program analysis. Particularly, check for Enqueue lockouts, excessive Store/Fetch flushing, excessive disk queuing (NUMCL too low for terminals/subsystems), VSAM exclusive control waits (Control Interval Lockouts), Data Base interregion access waits, etc.

The TALY,DA command (see System Control Commands) can be used to display information about currently active and hung threads.

5.9 DEBUGGING AIDS--THREAD RESOURCE AND POOL DUMPS

5.9.1 The Thread Resource Dump

This consists of a listing of all outstanding Resource Control Blocks (RCBs), broken down by thread. The dump is written by a routine called TDUMP onto a SYSOUT data set called SMLOG. Thread dumps are taken when a program check occurs and when a thread completes without freeing all its resources. One call to TDUMP is from SPIESNAP (accompanying a 126 snap); another is in RMPURGE, the routine called to purge "leftover" resources. If the thread dump is followed in the printout by a pool dump, it was taken by SPIESNAP; if not, it was taken by RMPURGE. TDUMP is also called by STAEEXIT to accompany snaps 121 (long-term loop control) and 122 (user/system abend), and by PMINQDEQ to accompany a snap 114 (enqueue time-out). An RCB for SMLOG will always appear at the top of the list of resources of one of the threads. (If it does not, storage destruction has occurred in the Thread Status Table, entry point TSTATAB, in SYCT400.) The thread owning SMLOG therefore had control when the dump was taken.

TDUMP is called with register 1 pointing to the address of a fullword argument. To dump one thread's resources, the argument is the thread number, that is, three bytes of zeros and the thread number in the low-order byte. Thread number can be obtained from IJKTHRED (an entry in the Dispatcher) which is the label of a fullword field containing the currently executing thread number in the low-order byte. For example:

```
LA    R1,=V(IJKTHRED)      POINT TO IJKTHRED ADDRESS
CALL TDUMP
```

To dump all the threads, CALL TDUMP with R1 pointing to the address of an argument of -1.

The RCBs are stacked, that is, a thread's most recently acquired resource is located at the top of its list and the oldest is at the bottom. This is useful in determining what a subsystem was doing just before the dump was taken. The contents of the in-line save area (INTSAVE) used by STORAGEM and STORFRED provide useful information in case of a snap, and is one of the areas snapped in an indicative dump (see Chapter 8). In a full snap, use the linkedit to find the MANAGER module (Csect RSMGMNT), and then look for the literal 'RMSAVE REGS 14 to 12' in the EBCDIC printing on the right side of the dump. The register contents (14-12) begin after the literal; there is no space for save area chaining. Register 15 can be checked to see if the module was entered at STORAGEM or STORFRED.

Successful execution of TDUMP requires including IJKCESD and IJKWHOIT in the Intercomm linkedit (see Chapter 4), and a DD statement for SMLOG (see Section 5.7.6).

The following explains the thread resource dump in Figure 5-3:

- THREAD/SUBCODE

  The three-digit thread number, 000-255, in decimal, followed by the two-byte subsystem code in hexadecimal. For thread 000 (the system resource thread), the subcode is meaningless.

- RESOURCE TYPE

  There are five resource types: CORE, FILE, DDQ, DYNL and NQ. For an enqueue resource, the entry will either be

  -- NQ(OWNER)--thread has control of the resource

  -- NQ(WAIT)--thread is waiting for control

  -- NQ(POST)--the ECB for the enqueue has been posted and the thread will get control after the Dispatcher transfers the corresponding WQE to the execute list

- ACQUIRED BY

  The Csect name (+ displacement), or the address, of the location immediately following a branch-and-link. If the resource is an area of storage, it may locate a call to STORAGEM, a call to PMILINK2, a PASS macro or a CATCH macro. If the resource is a file, it locates the call to SELECT. If the resource is an enqueue, it locates the call to PMINQDEQ generated by an INTENQ macro. If the resource is a DDQ, it locates a call to QBUILD or QOPEN. If the resource is a dynamically loaded subroutine, it locates the issuer of a MODCNTRL macro which requested access to the subroutine.

- SUBPOOL NUMBER

  Either nnn or ICOM. ICOM means the storage was acquired by MANAGER from the Intercomm pools, not dynamic (subpool nnn) storage.

- RESOURCE ADDRESS

  If storage, this is the start of the block. If a file, this is the address of the external DSCT; there is usually another RCB for an area of storage containing the external DSCT. If it is an enqueued resource, this is the address of the 72-byte resource-ID block obtained by PMINQDEQ. There will always be a storage RCB in the thread 000 list containing the ID block. Immediately after an NQ(WAIT) or NQ(POST), the RCB will be an RCB for a 128-byte work area which is chained to the ID block. If the resource is a DDQ, this is the address of the internal Queue Locate Block (QLB). If the resource is a subroutine defined in REENTSBS via a SUBMODS macro with the

LNAME parameter, this is the address of that macro's expansion in the DYNLSUBS Csect generated within REENTSBS.

- RESOURCE LENGTH

  The length of a storage resource (in decimal). Note that this value may be less than LOWLIM if a partial free was done from an ICOMPOOL block (flagged by an asterisk after the length value).

- ICOMPOOL HEADER

  The address of the doubleword control block prefixed to the pool block from which the storage resource was allocated. Generally, eight less than the resource address, unless part of the area has been freed or passed to another thread.

- ICOMPOOL BLOCKSIZE

  The size of the pool block from which the area was allocated.

- FILE NAME

  The file ddname. The owner of SMLOG caused the thread dump.

- DDQ NAME

  The 16-byte DDQ identifier.

- SUB NAME

  The eight-byte (dynamically loaded) Subroutine (DYNL) identifier (defined via a SUBMODS macro--see Chapter 3).

- NQ/DQ NAME

  The 16 to 44 characters of the identifier passed to PMINQDEQ via an INTENQ macro.

- RCB ADDRESS

  The location of the 20-byte RCB. There are a few things the RCB indicates that do not appear in the thread dump. A file RCB contains the address of the internal DSCT in a field labelled RCBIDSCT. An enqueue RCB has a flag set if it is an enqueue on an OS resource, and the address of the enqueue ECB is in RCBNQCHN. A storage RCB has a flag set if it is left over from a larger area. See the RCB Dsect in any of the Resource Management modules for flag settings and offsets.

Note: the phase IS ACTIVE BUT OWNS NO RESOURCES usually indicates the thread is in a CONVERSE wait.

THREAD RESOURCE DUMP — CALLED BY RMPURGE+24E — TIME=18.36.10, DATE=83.076

THREAD/ SUBCODE	RESOURCE TYPE	ACQUIRED BY	S/P NO.	RESOURCE ADDRESS	RESOURCE LENGTH	ICOMPOOL HEADER	ICOMPOOL BLOCKSIZE	FILE/DDQ/SUB/NQ/DQ NAME	RCB ADDRESS
000/0008	CORE	RMPURGE+11B	ICOM	153CC0	88	153CB8	96		19C0E8
000/0000	CORE	FDITCB+F6	ICOM	152F30	40	152F28	64		19C388
000/0000	CORE	SYCTRL+16	ICOM	15FEF0	336	15FFL8	336		19C680
000/0000	CORE	BTAMSIM+A22	ICOM	167L30	1040	167E28	1056		19C4F8
000/0000	CORE	BTAMSIM+63F	ICOM	16C418	1304	16C410	1304		19C624
000/0000	CORE	BLHOT+12B4	ICOM	152F78	48	152F70	64		19C548
000/0000	CORE	BLHOT+12B4	ICOM	161760	360	161758	384		19C660
000/0000	CORE	BTAMSIM+63E	ICOM	168DD8	1304	168DD0	1304		19C124
000/0000	CORE	BLHOT+12B4	ICOM	152DB0	48	152D78	64		19C264
000/0000	CORE	BLHOT+12B4	ICOM	153298	96	153290	96		19BEE0
000/0000	CORE	BLHIN+1862	ICOM	156588	224	156580	224		19BAA8
000/0000	CORE	BTAMSIM+63E	ICOM	16DEF8	1304	16DEF0	1304		19B8C8
000/0000	CORE	PMIRETRY+8A6	ICOM	1577C0	192*	157788	256		19C180
000/0008	CORE	BTAMSIM+63E	ICOM	169B18	1304	169B10	1304		19C250
000/8000	CORE	BSCDIAL+DD2	ICOM	1575B0	256	1575A8	256		19C4BC
000/0000	CORE	PMIRETRY+8A6	ICOM	1571E8	152*	1571A8	256		19C304
000/0000	CORE	MSGCOL+462	ICOM	155CD8	184	155CD0	192		19BB48
000/0000	CORE	BLHOT+C12	ICOM	1563B8	224	1563B0	224		19B92C
000/0000	CORE	PMISUBL2+F2	ICOM	154F68	120	154F60	128		19C778
000/0000	CORE	PMIRETRY+8A6	ICOM	157298	88*	157290	256		19C6C4
000/0000	CORE	MSGCOL+F80	ICOM	155A80	184	155A78	192		19C4D0
000/0000	CORE	BLHOT+C12	ICOM	1561E8	224	1561E0	224		19C390
000/0000	CORE	PMISUBL2+F2	ICOM	154EE0	120	154ED8	128		19B968
000/0000	CORE	MSGCOL+462	ICOM	166600	728*	1665F8	960		19C64C
000/0000	CORE	MSGCOL+462	ICOM	168BE8	1032*	168BE0	1280		19C5FC
000/0000	CORE	BTAMSIM+63E	ICOM	1695F8	1304	1695F0	1304		19C1EC
000/0000	CORE	BLHIN+1BFC	000	1D0560	2464				19BA08
000/0000	CORE	BLHIN+1862	ICOM	156CC8	224	156CC0	224		19BEB8
000/0000	CORE	BTAMSIM+63E	ICOM	16CE58	1304	16CE50	1304		19C4B0
000/0000	CORE	BSCLEASE+AD8	ICOM	1611C0	344	161188	352		19C278
000/0000	CORE	IXFMON01+3844	ICOM	153570	88	153568	96		19BE54
000/0000	FILE	BTAMSIM+C72		1750E0				CPU03	19B580
000/0000	CORE	BTAMSIM+63E	ICOM	16AA78	1304	16AA78	1304		19C430
000/0000	CORE	BLHIN+1862	ICOM	156758	224	156750	224		19C0AC
000/0000	CORE	MSGCOL+462	000	1D46B0	304				19C14C
000/0000	CORE	MSGCOL+462	000	1D47B0	304				19C458
000/0000	CORE	MSGCOL+462	000	1D48E0	304				19C73C
000/0000	CORE	MSGCOL+462	000	1D4DA0	304				19C714
000/0000	CORE	MSGCOL+462	000	1D4B40	304				19B558
000/0000	CORE	MSGCOL+462	000	1D4C70	304				19C764
000/0000	CORE	MSGCOL+462	000	1D3610	304				19C6EC
000/0000	CORE	MSGCOL+462	000	1D3150	304				19C570
000/0000	CORE	MSGCOL+462	000	1D3280	304				19B8C0
000/0000	CORE	MSGCOL+462	000	1D33B0	304				19C0FC
000/0000	CORE	MSGCOL+462	000	1D34E0	304				19C50C
000/0000	CORE	MSGCOL+462	000	1D3740	304				19C4E4
000/0000	CORE	MSGCOL+462	000	1BF0D0	304				19BE7C
000/0000	CORE	MSGCOL+462	000	1D3870	304				19BF6C
000/0000	CORE	MSGCOL+462	000	1D39A0	304				19C63B
000/0000	CORE	MSGCOL+462	000	1D3AD0	304				19C188
000/0000	CORE	PMISUBL2+F2	ICOM	155298	120	155290	129		19C318

Figure 5-3. Sample Thread Resource Dump (Page 1 of 3)

000/0000	CORE	IXFMON01+38A4	ICOM	153438	96	153430	96		19B878
000/0000	CORE	PMISUBL2+B4	ICOM	155F30	208	155F28	224		19B698
000/0000	CORE	IXFMON01+38A4	ICOM	1531C8	96	1531C0	96		19B6D4
000/0000	CORE	MMUSTART+330	000	1B9880	1920				19B6B4
000/0000	CORE	PMISUBL2+B4	ICOM	1544C8	112	1544C0	128		19B65C
000/0000	CORE	PMISUBL2+B4	ICOM	155B48	184	155B40	192		19B4A4
000/0000	FILE	GFDRIVER+76		117E84				GFFIN	19B670
000/0000	FILE	BTAMSIM+3C6		174AB0				PAUL1	19B634
000/0000	FILE	BTAMSIM+3C6		174798				NWK01	19B60C
000/0000	FILE	BTAMSIM+3C6		174A20				SFC02	19B5F8
000/0000	FILE	BTAMSIM+3C6		174948				OGU01	19B5E4
000/0000	FILE	BTAMSIM+3C6		174C18				CHI02	19B5D0
000/0000	FILE	BTAMSIM+3C6		174630				CNT01	19B5BC
000/0000	FILE	BTAMSIM+3C6		174750				IND05	19B5A8
000/0000	FILE	BTAMSIM+3C6		174C60				LON01	19B594
000/0000	FILE	BTAMSIM+3C6		175098				CPU02	19B56C
000/0000	FILE	BTAMSIM+3C6		1740B8				CIN01	19B544
000/0000	FILE	BTAMSIM+3C6		1740D0				CHI01	19B530
000/0000	FILE	BTAMSIM+3C6		174678				NYC02	19B51C
000/0000	FILE	BTAMSIM+3C6		1749D8				LAX01	19B508
000/0000	FILE	BTAMSIM+3C6		174CF0				PAR01	19B4F4
000/0000	FILE	BTAMSIM+3C6		174DC8				ROM01	19B4E0
000/0000	FILE	BTAMSIM+3C6		1746C0				PHL03	19B4CC
000/0000	FILE	BTAMSIM+3C6		1748B8				BOS01	19B4B8
000/0000	CORE	BTAMSIM+A0	ICOM	174540	3104	174538	3264		19B454
000/0000	CORE	PMISUBL2+B4	ICOM	154440	128	154438	128		19B490
000/0000	CORE	PMISUBL2+B4	ICOM	1626B0	392	1626A8	448		19B47C
000/0000	CORE	IXFMON01+38F2	ICOM	152488	32	152480	32		19B378
000/0000	CORE	IXFMON01+38A4	ICOM	1530F8	88	1530F0	96		19B260
000/0000	CORE	IXFMON01+38F2	ICOM	152460	32	152458	32		19B3F0
000/0000	CORE	IXFMON01+38A4	ICOM	1543B8	104	1543B0	128		19B3DC
000/0000	CORE	IXFMON01+38F2	ICOM	152438	32	152430	32		19B3C8
000/0000	CORE	IXFMON01+38A4	ICOM	154330	104	154328	128		19B3B4
000/0000	CORE	DDQTRANS+504	ICOM	153028	96	153020	96		19B364
000/0000	CORE	DDQTRANS+504	ICOM	164528	600	164520	640		19B288
000/0000	CORE	DDQTRANS+504	ICOM	163980	576	163978	576		19B274
000/0000	CORE	IXFMON01+38F2	ICOM	152410	32	152408	32		19B350
000/0000	CORE	IXFMON01+38A4	ICOM	1542A8	104	1542A0	128		19B33C
000/0000	CORE	IXFMON01+38F2	ICOM	1523E8	32	1523E0	32		19B328
000/0000	CORE	IXFMON01+38A4	ICOM	154220	104	154218	128		19B314
000/0000	CORE	IXFMON01+38F2	ICOM	1523C0	32	1523B8	32		19B300
000/0000	CORE	IXFMON01+38A4	ICOM	154198	104	154190	128		19B2EC
000/0000	CORE	IXFMON01+38F2	ICOM	152398	32	152390	32		19B2D8
000/0000	CORE	IXFMON01+38A4	ICOM	154088	104	154080	128		19B2C4
000/0000	CORE	IXFMON01+38F2	ICOM	152370	32	152368	32		19B2B0
000/0000	CORE	IXFMON01+38A4	ICOM	154110	104	154108	128		19B29C
000/0000	FILE	STUOVLY+412		1BEFCC				INTERLOG	19B24C
000/0000	FILE	STUOVLY+412		1BEADC				INTERLOG	19B238
000/0000	FILE	STUOVLY+412		1BE5EC				INTERLOG	19B224
000/0000	FILE	STUOVLY+412		1BF0FC				INTERLOG	19B210
000/0000	FILE	STUOVLY+412		1BDC0C				INTERLOG	19B1FC
000/0000	FILE	STUOVLY+412		1BD71C				INTERLOG	19B1E8
000/0000	FILE	STUOVLY+412		1BD22C				INTERLOG	19B1D4
000/0000	FILE	STUOVLY+412		1BCD3C				INTERLOG	19B1C0
000/0000	FILE	STUOVLY+412		1BC84C				INTERLOG	19B1AC

Figure 5-3. Sample Thread Resource Dump (Page 2 of 3)

```
000/0000    FILE    STUDVLY+412             1BC35C                          INTERLOG                    198198
000/0000    FILE    STUDVLY+412             1BBF6C                          INTERLOG                    198184
000/0000    FILE    STUDVLY+412             1BB97C                          INTERLOG                    198170
000/0000    CORE    STUDVLY+1222    000     1BB4C0    1516R                                             19B0L4
000/0000    CORE    IXFMON01+38A4   ICOM    153090      96    153088      96                            19815C
000/0000    CORE    TASKSTRT+64     ICOM    15E550     304    15E548     304                            198134
000/0000    CORE    IXFFAR+1188     ICOM    152B40      40    152B38      64                            198120
000/0000    CORE    IXFMON01+38A4   ICOM    152FC0      96    152FB8      96                            19810C
000/0000    CORE    IXFMON01+38A4   ICOM    154000     104    153FF8     128                            1980F8
000/0000    CORE    STOSTART+7E     ICOM    155540     160    15553B     160                            1980D0

                                                        0226 RESOURCES OWNED BY THIS THREAD.
                                                        0067856 BYTES OF MAIN STORAGE.
                                                        031 FILES.

001/00C3 IS ACTIVE BUT OWNS NO RESOURCES.

002/00C9    FILE    TDUMP+5A                150AB4                          SMLOG                       198940
002/00C9    CORE    SYCTRL+1212     ICOM    153F98      B0    153F90      96                            1988F0

                                                        0002 RESOURCES OWNED BY THIS THREAD.
                                                        0000080 BYTES OF MAIN STORAGE.
                                                        001 FILES.

004/C9C3    CORE    ICMTST+16       ICOM    153D90      96    153D88      96                            1988A0

                                                        0001 RESOURCES OWNED BY THIS THREAD.
                                                        0000096 BYTES OF MAIN STORAGE.

        CURRENT NUMBER OF UNUSED RCB'S = 0171          (* INDICATES THAT A PARTIAL FREE HAS BEEN DONE ON THIS BLOCK)
```

Figure 5-3. Sample Thread Resource Dump (Page 3 of 3)

5.9.2 Status of Intercomm Administered Storage (Pool Dump)

This is produced by a call to POOLDUMP. There are no parameters for the call, as with the thread dump. POOLDUMP is written onto SMLOG. Currently, the only time a pool dump is taken is after a program check; the call is in SPIESNAP following the call to TDUMP. Figure 5-4 illustrates part of the output from POOLDUMP.

The pool dump consists mainly of a block-by-block listing of the status of the Intercomm pools. It also includes the status of the storage cushion (if it is released, the SPAHOLD switch is set and no new threads are started) and the address of the RCB table. The latter information may be useful in examining the free-RCB chain. The location of the top RCB in the free chain is the first fullword in the RCB table. It is given as a halfword offset (divided by 4) from the start of the table. The length of the freed resource is a fullword 8 bytes into the RCB, followed by the resource's address and the address of the call that acquired it. However, RCBs are taken from the top of the free chain as well as returned there, so the one at the top of the chain may have been freed for some time.

If any of the addresses appear strange (such as 404040 or BBBBBB), that is a good indication that storage destruction has occurred (possibly by the owner of the preceding pool block). RMINTEG processing (see Section 5.7.1) or the TRAP module (see Messages and Codes) may be used to find the culprit.

5.9.3 Finding the Dynamically Loaded Pools

Under Release 9 of Intercomm, pointers to all pool VCONs (address of ICOMPOOL CSECT, etc.) are located in the SPAEXT. (Under earlier versions, they were located in MANAGER). Thus, if the addresses of these items are required in debugging a snap, the fullwords located in the SPAEXT which are listed below contain the addresses of the entry points listed at the right:

SPAEXT Label	ICPOOLxx Csect
SEXICMPL	ICOMPOOL
SEXPOOLN	POOLEND
SEXICMCH	ICOMCHN
SEXICMNX	ICOMINX
SEXPOOLA	POOLACCT

Note: When ordering resident pool Csects, the above order may be used; POOLEND must be ordered immediately after ICOMPOOL.

```
                    STATUS OF INTERCOMM ADMINISTERED STORAGE

02048-BYTE STORAGE CUSHION NOT RELEASED.
CUSHION ADDRESS = 1147A8
RCB TABLE AT 19A8B8, LENGTH = 10052 BYTES.
  400 TOTAL RCBS.
  170 FREE RCBS.
***USER POOL. BLOCKLENGTH = 00384

BLOCK IN USE. HEADER LOCATION = 161758
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 161760, LENGTH =   368, RCB OFFSET = 0015C4, GOT BY 131964

   10 TOTAL BLOCKS.
    9 FREE BLOCKS.

***USER POOL. BLOCKLENGTH = 00448

BLOCK IN USE. HEADER LOCATION = 1626A8
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 1626B0, LENGTH =   392, RCB OFFSET = 000BC4, GOT BY 1BE59C

BLOCK IN USE. HEADER LOCATION = 162CC0
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 162C08, LENGTH =   440, RCB OFFSET = 000CB4, GOT BY 1AE5DE

    6 TOTAL BLOCKS.
    4 FREE BLOCKS.

***USER POOL. BLOCKLENGTH = 00512

    4 TOTAL BLOCKS.
    4 FREE BLOCKS.

***USER POOL. BLOCKLENGTH = 00576

BLOCK IN USE. HEADER LOCATION = 163978
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 163980, LENGTH =   576, RCB OFFSET = 0009BC, GOT BY 1A9834

BLOCK IN USE. HEADER LOCATION = 163E08
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 163E10, LENGTH =   528, RCB OFFSET = 001128, GOT BY 1AD71C

    4 TOTAL BLOCKS.
    2 FREE BLOCKS.

***USER POOL. BLOCKLENGTH = 00640

BLOCK IN USE. HEADER LOCATION = 164520
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 164528, LENGTH =   600, RCB OFFSET = 0009D0, GOT BY 1A9834

BLOCK IN USE. HEADER LOCATION = 1647A8
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 1647B0, LENGTH =   600, RCB OFFSET = 001AD8, GOT BY 1B3B9E

BLOCK IN USE. HEADER LOCATION = 164A30
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 164A38, LENGTH =   600, RCB OFFSET = 0017A4, GOT BY 1A757A

    4 TOTAL BLOCKS.
    1 FREE BLOCKS.

***USER POOL. BLOCKLENGTH = 00704

    2 TOTAL BLOCKS.
    2 FREE BLOCKS.

***USER POOL. BLOCKLENGTH = 00768

BLOCK IN USE. HEADER LOCATION = 165248
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 165250, LENGTH =   766, RCB OFFSET = 001470, GOT BY 1A757A

BLOCK IN USE. HEADER LOCATION = 165550
THREAD/SS =   0/0000, SUBBLOCK ADDRESS = 165558, LENGTH =   768, RCB OFFSET = 000D54, GOT BY 131964

    2 TOTAL BLOCKS.
    0 FREE BLOCKS.
```

Figure 5-4. Sample Pool Dump

Chapter 6

FILE HANDLER SPECIFICATIONS

6.1 INTRODUCTION

The Intercomm File Handler provides data management facilities of the operating system to all user processing programs. Only external data management planning (data set organization and processing techniques) is required by the user. Internals are handled entirely by the File Handler.

The general purposes of the File Handler are to eliminate all the required input/output programming within those application programs functioning in the on-line system, and to coordinate all concurrent requests for input or output operations from the on-line programs. An I/O operation is requested by simply calling a File Handler service routine.

When a request for an input or output operation is received by the File Handler, the appropriate control blocks are generated, the operation is started and other programs in concurrent execution are allowed to continue operation. The File Handler provides overlap of I/O operations via the Intercomm Dispatcher (Event Queue). It is the interaction of the File Handler and the Dispatcher that provide Intercomm's multithreading facility within application programs and/or Intercomm programs during data set I/O operations.

In general, the functions performed by the File Handler provide:

- All I/O operations against on-line system data sets under monitor control
- Total overlap of all I/O operations with on-line application program processing
- I/O error analysis and simplified reporting of errors to the application programs
- Detection of errors which would otherwise cause abnormal task termination
- Elimination of opening and closing of data sets at each execution of an on-line processing module
- Exclusive record (or file) control preventing simultaneous record updating

- Simplified linkage and removal of all data management considerations, control blocks and exit routines from processing programs

6.2 ACCESS METHODS

All supported data set organizations (sequential, direct, keyed) and processing techniques (by logical record, by physical block, indexed sequential, random) are available to programs written in any language which may execute under Intercomm. Any data base structure may be accessed either directly by the user, or through the Intercomm File Handler. There is, therefore, no restriction imposed upon interfacing with any supported or unsupported (EXCP or BPAM) access methods.

The following access methods are supported by the File Handler:

- VSAM--Virtual Storage Access Method (KSDS, ESDS, RRDS)
- BDAM--fixed and variable length (keyed/non-keyed)
- BSAM--fixed, variable and undefined length
- QSAM--fixed, variable and undefined length
- BISAM--fixed and variable length
- QISAM--optionally treated by File Handler as BISAM
- CFMS--IBM's Chained File Management System (see Section 6.13)
- IAM--Innovation Access Method
- DISAM--Intercomm-developed and supplied access method
- AMIGOS--Comress-developed access method

While partitioned data sets are not supported, an individual member may be processed as a sequential data set.

6.2.1 QISAM via BISAM

ISAM support may optionally function for QISAM via BISAM, which is essentially transparent to the application programmer. QISAM requests may be coded (GET, PUT), but the File Handler will operate as if they were made for BISAM, eliminating multiple DCBs and reducing the range of exclusive control. A large core reduction is realized upon elimination of QISAM routines and the associated channel program and I/O areas. See Sections 6.2.8 and 6.4.5 for further details.

6.2.2 VSAM and VSAM/ISAM Compatibility

The VSAM access method of VS is fully supported through the File Handler in a manner which allows most existing ISAM files accessed by subsystems through the File Handler to be converted to VSAM files with no modification to the subsystems. New VSAM functions which did not exist in ISAM are also supported by the File Handler for use by newly developed application subsystems.

6.2.3 IAM

An interface with IAM, a transparent replacement access method for ISAM (using BDAM files), developed by Innovation Data Processing, Inc., is provided within the File Handler. AMIGOS and DISAM may not be used in the same region, but ISAM can be used. To implement Intercomm support, set &IAM in SETGLOBE to 1 and reassemble IXFHND00, IXFHND01, IXFQISAM, IXFFAR, ICOMCESD and IJKCESD.

6.2.4 DISAM

The File Handler provides support for DISAM, an Intercomm-supplied access technique consisting of ISAM index files and one or more BDAM data files, both of which are fixed or variable length. In addition, the data files can be referenced by either relative block number (RBN) or by relative track (TTR). (This support is described in Section 6.12.) DISAM and IAM should not be used in the same Intercomm region.

6.2.5 AMIGOS

An interface with AMIGOS, a replacement access method for ISAM, developed by Comress, Inc., is provided within the File Handler as an unsupported feature. This facility is described in the Intercomm AMIGOS Users Guide. AMIGOS may not be used in the same Intercomm region with ISAM, IAM or DISAM.

6.2.6 Exclusive Control

Exclusive control for update of a record is provided at the data set level (QISAM), physical record level (BISAM), and by record block (BDAM) to prevent simultaneous file updates from destroying one another's updates. If Resource Auditing is in use, the purge function automatically releases exclusive control in the event that a subsystem omits the release function (due to logic error, program check or time-out). If Resource Auditing is not in use, a File Handler

exclusive control time-out routine will, after an internally specified period of time, automatically release those records still held by the processing thread. To activate the File Handler exclusive control time-out routine, the File Handler must be reassembled with &RM set to zero in SETGLOBE. The File Handler, as released, relies on Resource Auditing to perform this function, not the time-out routine.

The following facilities of exclusive control are applicable to particular access methods:

- QISAM

  Optional QISAM via BISAM provides exclusive control at the physical record level.

- BISAM/BDAM

  The manner in which the File Handler provides exclusive control for ISAM files minimizes overhead. Each exclusive read request requires only one READ. No OS enqueues are issued. This results in minimal I/O activity and CPU usage.

See Section 6.6, "File Attribute Records," XCTL parameter for ISAM files, and ICOMBDAMXCTRL parameter for BDAM files.

For VSAM files, the access method provides exclusive control for update if Shareoption 1 or 2 is specified at file creation time. The File Handler provides exclusive control for a Shareoption 4 file if requested by a FAR parameter described later in this chapter. Also, shared or exclusive control for VSAM files (created with Shareoption 2 or 4) across multiple Intercomm regions (batch or on-line) in the same CPU is provided if requested by the user.

6.2.7 Dynamic Buffering

The File Handler will allow dynamic buffering for all data sets accessed via BISAM. The BUFNO parameter in the DD statement is ignored during execution and buffers appropriate for the block size are obtained at the first access following a SELECT, and freed by the subsequent RELEASE. This feature should reduce the dynamic subpool requirements for users with many ISAM files accessed on-line. There is no upper limit placed on the number of concurrent threads accessing a file (hence, exclusive control cannot be forced by BUFNO=1). The buffers will be obtained through Intercomm Resource Management; low core conditions in the dynamic pools are handled by time delays between retries for buffer space. Additionally, this feature eliminates the need to open all data sets at startup time to ensure adequate buffer pool space.

For VSAM files, buffer pools via IBM's Local Shared Resources option is supported as described later in this chapter.

6.2.8 Overlapped GET And READ/WRITE Processing

The following facilities of sequential file support are applicable to particular access methods:

- QSAM/QISAM

  If the QISAM via BISAM option is not utilized, the File Handler may support both QSAM and QISAM in an overlapped manner. All GETs to QSAM and QISAM files are totally overlapped with other processing in the Intercomm environment. Without modifying OS, Intercomm will overlap GETs to QSAM and QISAM files with other Intercomm productive work, without placing the entire task issuing a GET into a WAIT state. The SETL and ESETL macros, which are used by the File Handler when applications use GET/PUT logic on ISAM files, are also overlapped with other Intercomm activity. This facility is implemented automatically via the Generalized Subtasking facility (see the SPALIST macro, TASKNUM parameter).

  In addition, the File Handler will always leave one QISAM DCB open throughout the day for users of each QISAM file. If concurrent use of a QISAM file by two or more applications is required, a second (or third, etc.) DCB will be opened, as required. For the majority of messages, the extra QISAM DCBs will not have to be opened. This applies only to QISAM, since under Intercomm other access methods require only one DCB for the file. However, for shareable sequential input files on disk, multiple DCBs are also opened (see Section 6.5.4).

- BSAM/BISAM

  During overlap of I/O, the total of all READ and WRITE requests against a DCB may exceed the NCP subparameter. The excess threads may time out (Snap 118) trying to access the same DCB.

  The amount of overlapping against the same DCB can be controlled by the MNCL parameter of the SYCTTBL macro. Where many subsystems use the same ddname for BSAM or BISAM operations, they may be placed in competing Overlay A or VS execution groups. (See the OVLY and EXGRP parameters of the SYCTTBL macro.) The total MNCL value for all such subsystems within a group may not exceed the NCP value. Alternatively, the number of concurrent users of a resource (ddname) may be limited by the RESOURCE macro used in conjunction with associated SYCTTBL macros (see the RESOURC parameter).

For a few users, it may be necessary to single-thread certain I/O functions; that is, bypass the overlap of I/O. Among the File Handler options which can be specified for the Data Set Control Table are single-thread physical sequential BSAM reads and do not overlap BISAM reads. Either or both requests are easy to implement on a systemwide basis via the OPTIONS parameter of the IXFDSCTA macro used to generate the file control table, but can severely degrade system performance.

The amount of overlapping may optionally be controlled by application logic. Intercomm's INTENQ macro (with SHARE specified as less than or equal to the NCP value) and a subsequent INTDEQ macro may be used to control concurrent use of a specific resource (file).

See also the FAR option NCPWAIT described in section 6.6.1.

6.2.9 Creating and Defining ISAM Files

Because Intercomm uses the more efficient IBM BISAM access method against ISAM files, where possible, certain restrictions apply concerning the creation and definition of ISAM files for use under Intercomm:

- Do not define separate Area Names (PRIME, INDEX, OVERFLOW) when creating the file. Let the access method allocate these areas from the primary allocation defined for the file and from the CYLOFL DCB parameter on the DD statement. It is better to use the IBM Utility IEBISAM (or an Assembler Language program using BISAM) to create the file than to create it with a COBOL or PL/1 program. Do not define the file as blocked.

- Use only one DD statement on the execution JCL; do not define separate Area Names. The only DD parameters necessary are DISP=OLD or SHR, the data set name, the unit and volser if not catalogued, and the DCB parameter DSORG=IS. Optionally, OPTCD may also be specified for the DCB parameter.

- If an existing file to be used on-line under Intercomm does not meet the above criteria, use the FAR parameter OPEN=QUEUED to force only QISAM (GET/PUT) access to the file (see Section 6.6).

6.2.10 Undefined Record Support

Undefined record support applies to QSAM/BSAM only. Full GET/PUT, READ/WRITE support for undefined records on sequential data sets is provided by the File Handler. The application program must supply the record length as a parameter for File Handler calls.

6.2.11 Variable Length Sequential File Support

The application program must be aware that each block starts with a BDW (halfword of block length plus 4, followed by a halfword of binary zeros), and each record with an RDW. When READ and WRITE are used, blocking and deblocking of blocked files must be performed by the application program. If GET and PUT are used, the access method will block and deblock the file (if RECFM=VB). Whatever form, the record always starts with an RDW (halfword of record length plus 4, followed by a halfword of binary zeros). For output, the application program must initialize the RDW before calling the File Handler. When WRITE is called for a blocked file, both the BDW and the RDWs (for each record in a block) must be initialized. The type of access to the file must be specified by a FAR OPEN option; BASIC if READ/WRITE is used, QUEUED if GET/PUT is used. DCB=DSORG=PS must be specified on the DD statement. Also specify BLKSIZE (add 4 bytes for BDW), LRECL (including RDW) if a blocked file, and NCP=n and OPTCD=C (see Overlapped Processing above). See also the FAR NCPWAIT and WRITEOVER parameters.

6.2.12 Sequential Output Disk File Flip-Flop Facility

This facility invokes automatic protection of Intercomm from an x37 abend resulting from running out of space on a BSAM (sequential output) disk file or the Intercomm Log (when logging to disk).

A companion disk file must be defined to effect this protection. The ddname of the companion file is constructed by right-"padding" the ddname of the original file, up to the maximum of eight characters, with the character 'C'; one character of the ddname is replaced, if necessary. The following illustrates construction of the alternate ddname:

Original	No. Chars.	Alternate	Comment
INTERLOG	8	INTERLOC	Last character replaced by 'C'
DISKX	5	DISKXCCC	Padded with 'CCC'
XYZ	3	XYZCCCCC	Padded by 'CCCCC'

The two data sets are used alternately. When one gets full, the resulting x37 abend is intercepted, the full data set is closed, and output is written to the companion data set. The message FR080R is issued to instruct the operator to copy the full data set off-line, effectively "emptying" it so that it may then be reused. When both data sets become full, the message FR081I is issued, and Intercomm enters the wait state until the operator replies to FR080R.

To implement this facility, the module IXFB37 must be included in either the Intercomm linkedit or the Intercomm Link Pack Module, and the original disk file (for example, INTERLOG) must have the B37 FAR option specified. x37 abend protection may not be specified for any original file whose ddname is eight characters ending with the letter C. The DD statement for the alternate disk file (ddname ending in C) must be specified after the //PMISTOP DD DUMMY statement in the Intercomm JCL to prevent an internal DSCT from being created. No on-line access to the alternate file by non-system (Intercomm) programs is allowed.

Both the original and the companion data sets must reside on a DASD device, must be defined as physical sequential (DSORG=PS), may only be accessed using WRITE, and must have a disposition of SHR to allow off-line accessing after an x37 abend has occurred. Neither data set may be DUMMY nor have a dsname of NULLFILE. If they do not meet these criteria, then the original data set will not be marked as eligible for abend recovery. The data sets must be preallocated in another job, not in a previous step of the same job. The NCP count (DCB subparameter) must be exactly the same for both data sets, if chained scheduling is used. If recovery of the file after a system crash is desired, see the description of ICOMFEOF in Chapter 12. Abend protection is automatically forced for INTERLOG if INTERLOC is correctly defined. For other files, the user may also wish to specify the FAR options OPEN=BASIC and NCPWAIT. WRITEOVER (DISP=NEW) is forced for all files with x37 abend protection.

6.2.13 File Recovery

As a special feature, Intercomm provides for off-line restoration of updated files, on-line recovery of updated files during message restart, and for dynamic backout (Backout-on-the-Fly) after an application program check or time-out. These facilities are described in the File Recovery Users Guide.

6.2.14 Dynamic File Allocation Facility

This special feature provides for dynamic allocation (creation) and access of sequential files not defined in the Intercomm execution JCL. This feature is described in Dynamic File Allocation.

6.2.15 On-line File Control Commands

Two Intercomm system commands (see System Control Commands) are provided to perform on-line control of file access and to display statistics on file access. These are:

- FILE command--open or close a file; prohibit or allow access to a file; change the processing status (read only or allow writes); and under MVS to dynamically allocate and deallocate files defined in the Intercomm execution JCL (see Section 6.2.15).
- FHST command--display statistics on file selection and access for one or all files and/or VSAM LSR buffer pool usage; see also the description of File Handler Statistics Reports in Section 6.10.

6.2.16 Dynamic Deallocation and Reallocation via FILE Command

Two FILE command parameters are available to dynamically deallocate and reallocate on-line files. The parameters ALLOC and DEALL make use of MVS Dynamic Allocation services via the DYNALLOC macro (SVC 99). The syntax of, and response messages pertaining to, these parameters are fully described in System Control Commands. The following discussion deals with restrictions and operational considerations for these parameters.

The main purpose is to allow a file which is accessed thru the Intercomm File Handler and originally allocated to Intercomm via JCL to be deallocated and thus made available for processing by batch jobs. Once the batch jobs are completed, the file may then be reallocated to Intercomm and thus again become available for on-line subsystems. The commands cannot be used to allocate a file to Intercomm which was not originally allocated via the Intercomm execution JCL.

When MVS deallocates the file, all traces of it (JFCB, etc.) are disconnected from the job doing the deallocation. After deallocation, no reference to the file exists in the system control blocks belonging to Intercomm; it is as though the file was never allocated to Intercomm in the first place. In order to successfully reallocate the file later, information about the current allocation must be saved before the file is deallocated. That information is obtained out of various system control blocks such as the JFCB, TIOT and UCB, and saved in a storage area which is pointed to by the internal DSCT for the file. If it is known that the file will not need to be reallocated to Intercomm later in the run, the NOREALC option of the DEALL parameter can be used. This option causes the obtaining and saving of the reallocation information to be bypassed, thus saving some processing time and storage. Under MVS, do not code FREE=CLOSE for any data set.

In order to keep the amount of information that must be obtained and saved about a file to a minimum, and because certain information is unobtainable, the following restrictions on the reallocation of a file must be considered:

- Temporary data sets (&&dsname) may be deallocated but not reallocated.

- A data set whose DISP status was NEW in the beginning of the run (as coded on the JCL DD statement) will have a status of MOD when reallocated.

- If the ddname in the FILE command describes a concatenated data set, only one of the members of the concatenation will be reallocated. The member of the original concatenation that will be reallocated is unpredictable:

```
if //DD1  DD DSN=FILEA
   //     DD DSN=FILEB
   //     DD DSN=FILEC
```

  and DD1 is deallocated, upon reallocation, DD1 will point to either FILEA, FILEB or FILEC but not the original concatenation sequence.

- When a data set is deallocated, any subsequent reallocation will be attempted using DD statement parameters assigned via the original JCL. Any parameters <u>not</u> provided will not be supplied and the IBM defaults for them will be taken, as necessary:

DSN	-	as coded on DD statement
member-name	-	as coded on DD statement for a PDS.
Generation Data Group	-	if coded on DD statement
LABEL number	-	as coded on DD statement
LABEL type	-	as coded on DD statement
SYSOUT class	-	will be A upon reallocation. SYSOUT class may be overridden with the DEALL command CLASS option.
UNIT	-	Direct access types: 2305-1, 2305-2, 2314, 3330, 3330-11, 3340, 3350. Tape units: 2400, 3400. If the unit type is not one of the above, SYSDA will be used for reallocation.
VOL=SER	-	the first 5 volumes coded on the DD statement. Only one unit will be requested for a tape multivolume data set. For a DA multivolume data set, as many units as there are volumes will be requested for PARALLEL MOUNT.

DISP - as coded on DD statement with exception of NEW which is changed to MOD upon reallocation.

- Catalogued data sets are an exception to the above. For a catalogued data set, UNIT type and VOL=SER information is not checked. All other information, including LABEL data, is verified.

- If a data set is named by more than one ddname, each ddname must be named by the operator on a separate FILE command (for example, VSAM base and alternate index paths).

The following DCB subparameters will also be preserved for the specified data types; all other parameters will be taken from the internal DCB or DSCB:

BFTEK - Buffering techniques (BDAM, QSAM, BSAM)
BLKSIZE - Block size (BSAM, QSAM, BDAM)
DSORG - Data set organization (BSAM, QSAM, BDAM)
EROPT - DCB error options (QSAM)
LRECL - Logical record length (QSAM, BSAM)
NCP - Number of channel programs before CHECK (BSAM, BISAM)
OPTCD - Operational services (QSAM, BSAM, BDAM)
RECFM - Record format (QSAM, BSAM, BDAM)
DEN - Tape density (QSAM, BSAM)
KEYLEN - Key length (Keyed BDAM or ISAM)
LIMCT - Search limit (Keyed BDAM)
BUFNO - Number of buffers (all)

NOTES: DSORG=PO data sets are not supported by the File Handler and may not be deallocated. Sequential output disk data sets defined for x37 abend protection may not be deallocated. VSAM data sets may be reallocated but JCL overrides (AMP=AMORG) of VSAM parameters will not be preserved. That is, upon reallocation, VSAM will take all necessary parameters from its control blocks. IBM currently does not support the provision of VSAM AMP parameters via dynamic allocation.

6.2.16.1 Retry of ALLOC or DEALL After Error

Upon completion of the DYNALLOC macro, a return code in register 15 indicates whether or not the request completed successfully. If it did not, the error reason code field in the dynamic allocation request block is checked. The error reason codes are divided by IBM into classes as documented in the IBM OS/VS2 SYSTEM PROGRAM LIBRARY - JOB MANAGEMENT manual. An error code whose two-byte hex value is X'02nn' is represented to be significant of a failure due to insufficient

system resources. As such, Intercomm will consider such errors temporary and preserve the internal control blocks necessary for a retry. An error code whose value is other than X'02nn' is a permanent error, due to an invalid parameter list, system routine error or environment error. When these occur, the internal control block necessary for the function is freed and the request cannot be retried by entering a subsequent ALLOC or DEALL. In either case, after a failing ALLOC or DEALL, the status of the file remains the same as it was before the failing command. In the case of a permanent error, a snap (ID=34) is taken of the SVC 99 request block and the parameter list used to attempt the request. The snap is not taken for a temporary error. An error message is issued to the requesting terminal for both temporary and permanent error conditions.

6.2.16.2 Subtasking of DYNALLOC Macro

When a request for allocation is accepted by the operating system, a certain lag time for volumes to be mounted, off-line units to be varied, etc., may occur before the allocation request can complete. In order to avoid forcing all Intercomm activity to wait while these events take place, the system will attempt to issue the DYNALLOC under a general subtask. To take advantage of this, the user should implement the Intercomm Generalized Subtasking facility in his system (see Chapter 3). As many general subtasks should be created as there are expected to be concurrent DEALL or ALLOC commands entered, plus the number required for other system and user functions. This is important because, if a general subtask is not available, ICOMTASK performs the subtasked code (the DYNALLOC macro) under the main task, which may cause a significant deterioration of system performance.

6.2.16.3 Status of Files While Deallocated

Intercomm closes and marks a file as locked in the internal DSCT before deallocating it. This means that any subsystem selecting the file through the File Handler will receive a return code of C'9' in the status field, and no I/O can be done. If the deallocation request fails, the file will remain locked but may be unlocked by a FILE$UNLOCK command. Thus, the operator may free the file for subsystems to use until the deallocation request is retried. If the deallocation is to be retried immediately, however, it is recommended that the file not be unlocked so as to avoid the time lag involved in quiescing the file a second time.

Once a file has been deallocated, it remains locked until a subsequent successful reallocation (FILE$ALLOC) request occurs. An unlock command cannot unlock a deallocated file. Upon successful reallocation, the file is immediately marked unlocked, regardless of whether or not it was locked prior to deallocation by a FILE$LOCK command.

6.3 VSAM FILE SUPPORT

All three VSAM file types (key-sequenced, entry-sequenced and relative-record) are supported under Intercomm. Access may be either sequential or direct via key, relative byte address (RBA) or relative record number (RRN), where applicable. Generic keyed access may also be performed. Additionally, alternate index (path) and base cluster processing may be performed against KSDS files. Details on access parameters and restrictions are provided in the Intercomm Programmers Guides.

Several additional restrictions and processing considerations apply to using VSAM files as follows:

- Do not define a JOBCAT DD statement for the Intercomm execution JCL stream.

- If user catalogs are used, define the STEPCAT DD statement(s) after the //PMISTOP DD DUMMY statement (see Section 6.5) in the Intercomm execution step in order to prevent File Handler access to the catalog at startup. DISP=SHR must be coded.

- If an ESDS file is loaded on-line, at least one (dummy) record must preexist on the file, otherwise, if a read (GETV call) is made against the file before the first write (PUTV call) is issued on-line, an unpredictable error return code will result if there are no existing records in the file. Also, the file should by loaded by only one subsystem which should be single-threaded (MNCL=1). When allocating the file via IDCAMS, specify RECOVERY (not SPEED) on the DEFINE statement.

- STAEEXIT must be included in the Intercomm linkedit to ensure closing of VSAM files after an abend occurs (see the description of STAEEXIT usage in Chapter 8 and of its processing in Messages and Codes). Note that Intercomm file closing is not performed after a system cancel (x22 abend) under MVS, or if a second abend condition occurs during STAEEXIT processing (VS1 and MVS). The VS Operating System does not perform VSAM file closing if STAEEXIT does not successfully complete, nor, of course, if a system crash (requiring reIPL) occurs. Therefore, it may be necessary to add steps to the Intercomm execution JCL stream to run IDCAMS on critical (updated) files before starting/restarting Intercomm. While a VERIFY operation will make an inquiry-only file accessible (but does not update the catalogue), it is recommended to use a REPRO (unload/load) operation against a file updated (added to) on-line in the previously unsuccessful execution.

- When using an alternate index to access a base cluster, the base file should be opened at startup; use the FAR parameter OPEN=VSAM (see Section 6.6).

6.3.1 Using a VSAM Local Shared Resources Pool

Local Shared Resources is a VSAM facility which allows selected VSAM data sets to share a common set of buffers rather than having a pool created for each data set at ACB open time. This facility implements a more efficient utilization of VSAM buffers and of dynamic storage since buffers will be acquired for a data set only when an I/O operation is started and are returned to the pool when the I/O completes. The buffer pool is acquired by VSAM when the BLDVRP macro is issued at startup, ensuring that the buffer pool will reside in a contiguous storage area and thus reducing storage fragmentation. Since the Intercomm File Handler overlaps I/O requests for VSAM data sets, use of Local Shared Resources can cut down on paging requests for I/O buffers; if a page containing a buffer is fixed for one I/O operation, no subsequent paging need be done for other I/O operations which require buffers residing on the same page. Additionally, the user has the option to page fix the whole resource pool in real storage which may prove advantageous if use of the pool is expected to be heavy. For further information on Local Shared Resources, see the IBM VSAM Options for Advanced Applications manual.

To install Local Shared Resources under Intercomm, first code the applicable BLDVRP parameters on the SPALIST macro. The parameters on the SPALIST are coded exactly the same as they would be coded on the VSAM BLDVRP macro (omitting the TYPE parameter). Coding these parameters causes a list form of the BLDVRP macro (a BLDVRP parameter list) to be built in a Csect named VRPLIST. The BLDVRP parameter list is variable in length, the length dependent upon the number of buffer pools there are. (Each VSAM buffer size coded causes a pool to be built; for example, if 512 and 1024 are specified, a pool of 512-byte buffers and one of 1024-byte buffers are built.)

One and only one local shared resource pool may be built per Intercomm region. For each region in a Multiregion Intercomm, code BLDVRP parameters on each region's SPALIST.

The pool is built at startup when an execute form of the BLDVRP macro is issued naming the list form BLDVRP in the VRPLIST Csect. Once this is accomplished, the resource pool characteristics cannot be changed until Intercomm is brought down and back up again with a revised version of the SPALIST coding in the linkedit. Furthermore, the pool will not be built if Intercomm does not find at least one VSAM data set to be connected to it, as discussed below. Once BLDVRP completes, a message is issued giving status information on the pool. If unsuccessful, the return code is displayed. Certain parameters can be checked at assembly time (such as invalid buffer size) but others, such as failing GETMAIN, are contingent on circumstances. If the BLDVRP fails, it is not retried and VSAM buffers will be built per data set as usual.

If it is desired to fix the buffer pools and/or I/O control blocks in real storage, the Intercomm Interregion SVC (IGCICOM) must be installed in the system (see Chapter 7). For further information on the FIX option of BLDVRP, see VSAM Options for Advanced Applications.

6.3.1.1 Connecting Data Sets to the LSR Pool

The local shared resource pool will be built only if the user specifies data sets to be connected to it. This is done by a FAR option, LSR, coded on a FAR statement for each data set that is to use the shared resources. When LSR is coded, the File Handler will alter the ACB for the data set to connect it to the resource pool and test special OPEN return codes for it. Also, resource-pool-oriented usage statistics may be accumulated for the buffer pools. These statistics are discussed in Section 6.10; File Handler Statistics.

Specifying LSR for a VSAM file also causes its ACB to be opened at startup, provided a VSAM resource pool exists. (That is, BLDVRP was successful.) If a VSAM resource pool is not created, the data set is not opened at startup unless OPEN=VSAM is also specified on the FAR card.

Even though a local shared resource pool may be created successfully, a data set may be unable to connect to it. This latter fact is discovered when its ACB is opened and VSAM returns a special return code indicating the error. (These error conditions and return codes are fully discussed in VSAM Options for Advanced Applications.) When an attempt to connect a data set to the resource pool fails, the File Handler will issue a message to call attention to this error and then retry the OPEN, this time using the normal nonshared buffers. That is, the retry of the OPEN will not specify connection to the shared resource pool. When an attempt to connect a data set to a resource pool fails, it is usually due to a conflict between the data set control interval size and resource pool specifications, or because the data set is empty. The return code in the error message can be used to determine the necessary action to be taken.

During execution, any VSAM request failing due to a lack of resources (for example, STRNO exceeded or no buffers available) will be retried on a 1/3-second basis. Statistics about these failures may be kept and reported so that the resource pool configuration may be adjusted accordingly, as described in Section 6.10.1.

6.3.2 Sharing VSAM Files Under Intercomm

When a VSAM Shareoption 2 or 4 file is shared by multiple Intercomm on-line or batch regions in the same CPU, the VSAMCRS FAR option can be used to augment VSAM shared file protection. For Shareoption 1 data sets, VSAM provides total READ/WRITE integrity. For Shareoption 3 files, VSAM provides no integrity; integrity for such files is not provided by Intercomm either.

For Shareoption 2 files, VSAM provides complete WRITE integrity in the update region; that is, it will allow only one GET-update/ PUT-update or PUT-insert at any time. VSAM does not provide READ integrity in this instance; a record just read by one region may be updated or deleted by another before the first region is finished

processing it. The VSAMCRS FAR option augments VSAM processing by providing READ integrity for Shareoption 2 files. Under this option, Intercomm will issue an OS ENQ for shared control of the file on the first GET by a thread, and retain that ENQ until the last user in the same region releases the file. This will allow any region sharing the file to read from the VSAM file, but no user may update that file until all regions have released shared control. Conversely, no region may read from the file while one region holds an exclusive control ENQ on the file for the purpose of updating. Thus, Intercomm ensures that a user program always has the latest copy of a VSAM record. The VSAM file in the read-only region must also have the READONLY FAR option specified for it.

For Shareoption 4 files, VSAM provides minimal aid toward READ/WRITE integrity. The VSAMCRS option will ensure file integrity in this case again by ENQing on the file for shared control before GETs, and for exclusive control before GET-update/PUT-update or PUT-insert. In addition, an exclusive control ENQ within the region is issued before processing any sequential request (for update or not) so as to preserve VSAM positioning for the file. A DEQ and an ENDREQ are issued at subsystem release time to release this positioning as well as to cause VSAM to write out any updated buffers.

To conclude, the VSAMCRS FAR option should be coded when:

- READ integrity is desired for a Shareoption 2 VSAM file which will be updated by another sharing Intercomm region.
- A Shareoption 4 file will be shared across two or more Intercomm regions.

If any batch regions will be sharing the file while Intercomm is executing, the batch access should be performed via the File Handler. If this is not done, the user program should issue an OS ENQ before any VSAM access, and DEQ afterwards (see the description of the VSAMCRS FAR option for enqueue names). Further information on sharing of VSAM files may be found in the IBM VSAM Programmers Guide.

6.3.2.1 Implementation for Sharing VSAM Files Across Regions

The VSAMCRS option must be coded on a FAR card for a Shareoption 2 or 4 VSAM file in every Intercomm region which will share that file. In addition, the module IXFVSCRS must be linked with the File Handler, IXFHND01. IXFVSCRS is Link Pack eligible so it must be linked with IXFHND01 when the File Handler is Link Pack resident. The File Handler will check for the VSAMCRS option when SELECT is called and ensure that the IXFVSCRS module has been linkedited with it. If IXFVSCRS is not present, SELECT will shut off the option, mark the file locked, and return a code of 9. The VSAM file may be used but only if the operator unlocks the file via the FILE command (see System Control Commands).

If VSAMCRS is coded for a Shareoption 3 file, the option is ignored and the file is locked. In this case, the file may be used if the operator unlocks the file via the FILE command. However, if the operator unlocks a file which was locked because of either of the above reasons, unpredictable errors may occur.

If VSAMCRS is coded for a Shareoption 1 file, it is ignored but the file is not locked. However, VSAM may not allow the region to open the file because Shareoption 1 restricts processing of a file to a single region.

The Intercomm Interregion SVC (IGCICOM) must be installed.

Note: TCTV time-out values of subsystems using VSAMCRS files may have to be increased substantially, depending upon volume of activity against the files used. The ENQ issued by IXFVSCRS is done with time-out suppressed, so that the limiting value is the subsystem time-out value. However, if an OS ENQ request for exclusive control never completes during thread purge processing because the thread is disabled (see Chapter 5), then further access to the file may be prevented because an update request never completed. The TALY,DA system control command may be used to determine thread status.

6.3.3 ISAM/VSAM COMPATIBILITY UNDER INTERCOMM

Subsystems accessing ISAM files can function with little or no modification when their files are converted to VSAM. Intercomm's ISAM/VSAM interface does not use IBM's VSAM/ISAM interface modules. ISAM/VSAM support is provided as an option which is specified by setting the global &VSISAM to 1 in SETGLOBE before assembly of IXFHND01.

The File Handler, when processing a converted VSAM data set, uses QISAM-compatible access for a GET or PUT call and BISAM-compatible access for a READ or WRITE call. An ISAM retrieval is converted to a VSAM GET for update. If a key is provided, it is, of course, treated as a full key. For GET, with a key, positioning and a search for a greater or equal key is performed. For READ, a search is made for an equal key. The FHCW is initialized internally for this operation.

ISAM delete code processing continues to function as usual via the OPTCD subparameter of AMP on the DD statement. The new OPTCD parameters (I,IL) which specify supplementary delete code processing are also supported.

The appropriate Intercomm Programmers Guide should be consulted for specifics on coding techniques and return codes.

6.4 FILE HANDLER COMPONENTS

The File Handler is organized into eight control sections:

Member	CSECT	Function
IXFDSCTn	IXFDSCTA	Data Set Control Table
IXFHND00	IXFMON00 IXFMON09	File Handler Initialization File Handler Termination
IXFHND01	IXFMON01	File Handler Processing
IXFQISAM	IXFQISAM	QISAM Scan Mode via BISAM
IXFFAR	IXFFAR	File Attribute Record Processing
IXFB37	IXFB37	File Flip/Flop Processing
IXFVSCRS	IXFVSCRS	VSAM Cross-region Control Processing

The functions of each control section are detailed below, and diagrammed in Figure 6-1. If any new version of any supported access method (particularly VSAM) is installed, all File Handler components must be reassembled and relinked.

6.4.1 Data Set Control Table (IXFDSCTA)

The Data Set Control Table (DSCT) contains, during execution, an entry for each file (data set) that may be processed by the File Handler. Each entry contains the data set name (corresponding to the name of the Job Control DD statement defining the file); the addresses of any Data Control Blocks or Access Control Block constructed to process the file; buffer addresses; flags defining file characteristics (data set organization, device type, disposition, and access method); flags identifying the current processing status of the file; I/O error flags; and a pointer to an associated File Attribute Block (FAB), if any, created at initialization time via IXFFAR.

Fixed information in each entry is inserted by the initialization routine (IXFMON00) at startup, and variable information is recorded in the entry during execution by the File Handler processing routine (IXFMON01).

The first DSCT entry is preceded by a DSCT header containing a count of the number of entries used, and flags for communicating general processing options from IXFMON00 to IXFMON01. The DSCT is a resident table containing 20 entries, assembled as a Csect within the member IXFHND01. As described below, this individual control section may be replaced to change the size of the DSCT to accommodate more files.

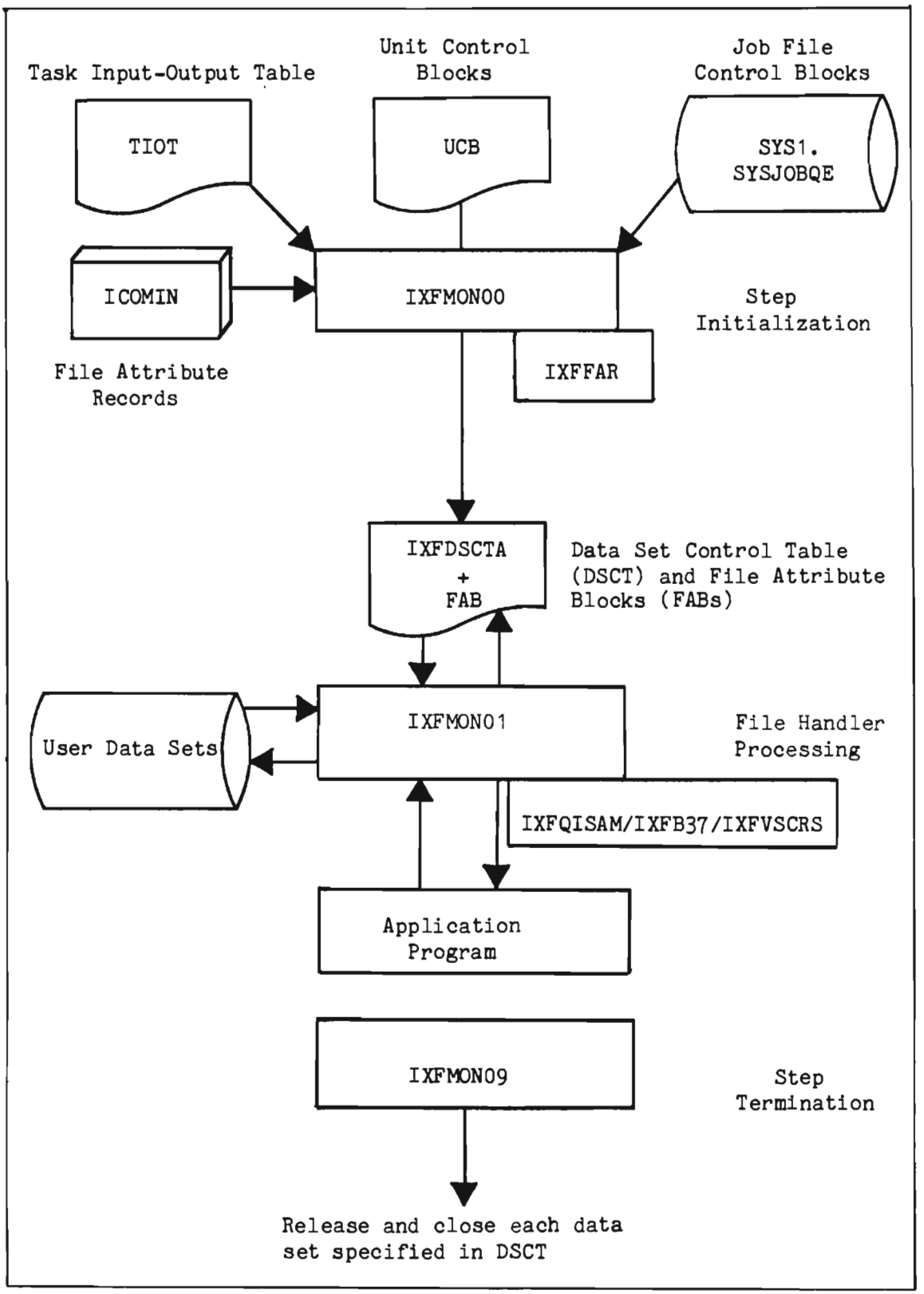


Figure 6-1. File Handler Components.

6.4.1.1 Defining the Data Set Control Table

The File Handler Data Set Control Table (DSCT) specifying the maximum number of data sets to be accessed is created by the Intercomm macro, IXFDSCTA. The File Handler file processing member (IXFHND01) contains a DSCT allowing up to 20 data sets (DD statements) to be accessed during Intercomm execution. Additionally, the Intercomm release contains three other members, one of which may be utilized to allow 50 data sets (IXFDSCT1), 100 data sets (IXFDSCT2), or 200 data sets (IXFDSCT3). Code the DSCT parameter on the ICOMLINK macro, when generating the Intercomm linkedit control statements, to specify which member is to be used. Alternatively, an installation may generate its own DSCT by coding the IXFDSCTA macro to specify a more precise maximum. Any DSCT to be used in lieu of the File Handler DSCT must be included prior to IXFHND00 and IXFHND01 in the Intercomm linkedit.

The IXFDSCTA macro also allows specification of File Handler options and statistics requirements as discussed in subsequent sections. The Intercomm-supplied DSCTs specify no options; statistics are for detailed access statistics. Refer to Section 6.9.4 on IXFDSCTA options and to Section 6.10 on File Handler statistics for procedures to follow if other than a release version of the DSCT is used.

6.4.2 File Handler Initialization (IXFMON00)

This Csect (within member IXFHND00) is executed at system startup to initialize all entries in the Data Set Control Table. The names of all Job Control DD statements in the current job step are found from the operating system Task Input Output Table (TIOT). For each DD statement, the allocated device type is determined (through a system macro instruction) and coded information from the DD statement is accessed (from the associated Job File Control Block). Additional information is determined by opening, and subsequently closing, the DCB or ACB, if VSAM. If the data set cannot be opened, it is flagged as locked (unusable) in the DSCT, and an error message is issued to the system console. If corrective action is taken, see the FILE command (in System Control Commands) for dynamically altering the status of a file. If the device type and data set characteristics are supported by the File Handler, the name and selected information from the above sources is transferred to an entry in the DSCT. Subsequently, additional fixed information concerning the data set is located from FAR options specified for the data set. This FAR information is also transferred to the DSCT entry, or File Attribute Block, as applicable.

When the TIOT has been completely scanned, the DSCT header is then filled in. Should the initialization routine be inadvertently called again at any time after the DSCT has first been initialized, no action will be performed. If, during File Handler initialization, the DSCT becomes filled and unprocessed TIOT entries remain, a console message is written and the job step is terminated.

6.4.3 File Attribute Record Processing (IXFFAR)

This routine is executed in File Handler initialization during Intercomm startup to read and analyze an input data set defining various optional attributes per on-line file, such as input only, update only, name alias, open at startup, exclusive control processing, etc. FAR specifications are described in Section 6.6, and in the Intercomm File Recovery Users Guide.

6.4.4 File Handler Processing (IXFMON01)

This Csect (within member IXFHND01) is composed of one mainline routine for each function (SELECT, RELEASE, LOCATE, GET, PUT, GETV, PUTV, READ, WRITE, RELEX). Each mainline routine verifies the caller's parameter list, maintains the DSCT status information, determines the access method to be used, issues the appropriate Data Management Input/Output macro instructions, checks and moves the record or block, and sets the resulting status code for the caller. Exclusive control processing is also performed if requested and/or applicable depending on data set type. See Section 6.7, "File Handler Service Routine Summary." Other Csects in this module are IXFSUBS which performs save area acquisition and chaining, and IXFABWTO which issues an error message and forces a program check (via ISK-see Messages and Codes) when an unrecoverable logical or physical error occurs.

6.4.5 QISAM Scan Mode via BISAM (IXFQISAM)

IXFQISAM provides the interface so that the function of QISAM Scan Mode is supported by using BISAM. Core requirements are significantly reduced when an indexed sequential file accessed by QISAM and BISAM can be accessed only through BISAM. The set of control blocks, buffers, channel programs and work areas, tied up for QISAM as long as the data set is open, is thereby eliminated.

IXFQISAM must be included in the Intercomm linkedit, along with the other File Handler modules even if IXFHND01 is in the Link Pack Area. If IXFHND01 is resident, the following statements must precede the include statement for IXFHND01:

```
CHANGE GET(GETZ)
CHANGE PUT(PUTZ)
```

These statements are automatically generated by the assembly of the ICOMLINK macro. Thus, a program call to the GET or PUT routines will initially enter IXFQISAM. If the request is not for a file to be processed by this module, control is transferred to the revised entry points (GETZ and PUTZ) in IXFHND01. If this is not used, remove the INCLUDE statement for IXFQISAM and the two CHANGE statements in order to reduce processing overhead for sequential files accessed via GET and PUT.

6.4.6 File Handler Termination (IXFMON09)

This routine, (a Csect in IXFHND00) calls the RELEASE function to close each data set opened by the File Handler. When a file is closed, it is closed for every access method for which it was opened, and all buffers and main storage areas previously acquired for construction of control blocks are released.

Typically, this step termination routine is required only once per job step; it may be incorporated in a nonresident segment of the overlay program structure. The abend intercept routine STAEEXIT conditionally calls IXFMON09; therefore, if it is nonresident, the overlay region it occupied may be overlaid in a dump.

6.4.7 Sequential Output File Abend Control (IXFB37)

IXFB37 receives control from IXFHND01 after an x37 abend has occurred for a sequential output disk file defined for such abend protection. It cancels the outstanding WQE requests (posted with a code of X'40') representing chained writes against the file which has become full, opens the alternate data set, and then restarts the outstanding writes against that data set in the same order in which the writes were initially issued so that sequential record integrity is not lost.

6.4.8 VSAM Cross-region Shared Control (IXFVSCRS)

If included in the same linkedit (Intercomm region or Link Pack) as IXFHND01, IXFVSCRS is called for every access to a VSAM file. If the file was defined as eligible for cross-region processing, IXFVSCRS determines the type of system ENQ to issue (for shared or exclusive control or to CHNG to exclusive control, if applicable) via an Intercomm INTENQ macro. An INTDEQ is issued and ENDREQ processing is performed when the subsystem thread or resource purging (RMPURGE) calls RELEASE for the file.

6.5 DATA SET SPECIFICATIONS

Every data set which may be accessed during the course of Intercomm execution (from startup to closedown) must be defined by appropriate DD statements in the execution JCL. All files must be mounted prior to initiation of Intercomm, except those for which deferred mounting is specified. After initiation, a subsequent requirement for mounting a deferred data set, or a volume of a multivolume tape file, may cause suspension of all message processing activity in the system until the request is satisfied, depending upon the operating system used.

All data sets accessed under Intercomm control must be previously existing data sets (DISP=OLD or SHR), except sequential output data sets (DISP=NEW or MOD). That is, VSAM, BDAM or ISAM data sets to be accessed on-line must be created in a step preceding the execution of Intercomm. The Intercomm-supplied utilities CREATEGF (for non-keyed) or KEYCREAT (for keyed) may be used to initialize BDAM data sets.

Message processing programs will refer to each data set by its one- to eight-character ddname, as specified in the job control data definition statement defining the data set. Each program which uses the same data set must refer to it by the same ddname.

It may be desirable to exclude certain DD statements containing the DSORG parameter (such as data sets controlled by a DBMS attached in the Intercomm region) from being included in the DSCT table. To accomplish this reduction in size of the DSCT, insert

```
//PMISTOP DD DUMMY
```

after the last DD statement to be included in the DSCT. All Intercomm data sets must precede this statement, except those used for snap output and dynamic linkedit, and the JCL for BTAM lines.

6.5.1 Required DD Parameters

All DD statements defining data sets to be processed by the File Handler must specify the DCB subparameter DSORG=(PS, DA or IS) for SAM, BDAM or ISAM data sets, or AMP=AMORG for VSAM data sets. For fixed length VSAM files, specify AMP=(AMORG,'RECFM=F'). Files for which DSORG or AMP are not specified on the DD statement will not be considered by the File Handler when constructing its internal data set control information at system startup.

A DUMMY file (or DSNAME=NULLFILE) may be specified for any data set referenced through the File Handler, and any DSORG listed above may be specified. This is useful in eliminating unnecessary data set definition and I/O operations upon data sets that are not to be used in a given job. For example, the output log file may be eliminated by specifying a dummy data set, or an indexed file containing no existing

records can be simulated for testing program logic by specifying a dummy data set. Any File Handler operation may be called for a dummy data set; successful completion return status will be given to the requesting program for operations other than input; EOF or KEY NOT FOUND return status will be given when an input operation (GET or READ) is attempted. This feature does not apply to x37 abend protected files.

For sequential, multivolume output files, SUL should be coded in the LABEL parameter to avoid subsystem time-outs which could occur between volume mounts.

6.5.2 Required DCB Parameters

The DCB parameters listed in the following table should be contained in each data set label. The label is created from parameters specified in the DD statement when the file is created or is subsequently opened for output or updating. Any parameters omitted from the data set label must be specified in the DD statement used in the processing job step.

DCB Parameter	Function
DSORG	specifies PS (sequential), IS (indexed) or DA (direct). DSORG is required on the DD statement (unless VSAM).
AMP	specifies AMORG for all VSAM files and is required on the DD statement.
RECFM	specifies record format: F, FB, U, V or VB (with A and/or S).
BLKSIZE	specifies exact or maximum block size, including 4 for BDW, if applicable.
LRECL	specifies exact or maximum logical record length, including 4 for RDW, if applicable.
KEYLEN	specifies key length (IS and Keyed DA only).
RKP	specifies relative key position (IS only).
OPTCD	specifies standard DCB macro parameters. E must be added for Keyed BDAM with extended search option.
LIMCT	specifies the number of records or tracks to search when using the extended search option for Keyed BDAM.
NCP	specifies the maximum number of I/O operations that may be started for a sequential data set (BSAM or BISAM).

6.5.3 Read-Only Data Sets

One or more data sets may be specified as read-only by means of FAR parameters. Requests for output operations upon data sets specified as read-only are not accepted. For VSAM alternate index processing, all paths but the one used for update must have read-only specified.

Read-only specification provides a method for protecting a data set for inquiry only when referred to by one ddname, while allowing full access to programmers using another ddname for update of the same data set. However, the inquiry requests may not always access the most recently updated version of the record, depending on the buffer emptying processing of the access method used.

6.5.4 Shareability of Sequential Data Sets (QSAM/BSAM)

A sequential data set is shareable among subsystems executing in the same Intercomm region if:

- The data set disposition is OLD or SHR (read-only) and not on tape (can be repositioned)
- The data set disposition is either NEW or MOD (write-only) and interleaving of output records is immaterial (tape or disk) and the DCB is not closed via RELEASE.

A sequential data set is not shareable if it resides on tape and has disposition OLD or SHR.

If a sequential data set is shareable, the following occurs:

1. The status code returned by SELECT is a 1 if a SYSOUT data set, disk output, or on tape (0 if disk input).
2. Write operations upon the file requested by the same or different threads are performed in the order requested, without repositioning.
3. Processing modes may not be intermixed: If GET or PUT processing is used by any program, no other program may employ READ or WRITE processing upon the same file, and vice versa.
4. A disk data set with DISP=OLD or SHR is repositioned and processed from the beginning for each new subsystem thread (a new DCB is opened for each thread).

6.5.5 Data Set Disposition

The disposition indicated on the DD statement is related to the operations which can be performed upon the file, as follows:

- NEW/MOD--The file can only be a sequential shareable data set (see above), and no input operations are allowed.
- OLD/SHR--Both input and output operations are allowed (provided the data set is not read-only); output operations (depending on access method restrictions and processing options) may be rewrites of existing records, additions of new records, insertions of keyed records, or writing over of an existing sequential file (see FAR WRITEOVER parameter).

6.5.6 SYSIN/SYSOUT Data Sets

If data sets are defined as DD *, DD DATA, or DD SYSOUT=x and are accessed through the File Handler, they are processed in the same manner as shareable sequential data sets, even though the actual assignment is either to a unit record device or intermediate direct-access storage. The implied dispositions are: SYSIN--OLD; SYSOUT--NEW. For SYSOUT data sets, DCB parameters are required: DSORG=PS, RECFM and BLKSIZE, also LRECL if blocked. Under MVS, do not code FREE=CLOSE for a SYSOUT data set because it is opened and closed during File Handler initialization; the close will automatically deallocate the data set. Use the FILE command to dynamically deallocate it.

6.5.7 Reserved ddnames

The following ddnames are reserved for Intercomm System use and should not be assigned to user data files:

- CHEKPTFL--System Checkpoint File
- DES000--File Description Records File (Change/Display)
- DYNLLIB--Dynamic Linkedit Load Module File
- DYNLPRNT--Dynamic Linkedit Print File
- DYNLWORK--Dynamic Linkedit Work File
- FASTSNAP--Used by Fast Snap facility (see Chapter 8)
- FRLOG--File recovery image printing at restart time

- ICOMIN--File Attribute Record input
- INTERLOG--System log (current)
- INTSTORn--Used by Store/Fetch Facility (and MMU)
- LOGDISK--Restart Work File
- NEWSNAP--Alternate snap data set used by Spinoff facility
- NULLFILE-Dummy File (File Handler)
- PAGES--Used by Page Facility
- PMISTOP--Delimits last DD statement to be processed by File Handler
- RCT000--Output Utility Format Table disk-resident entries
- RESTRTLG--System log (for restart)
- RJExxxxx--One per input spool data set and one per RJE terminal (OS RJE facility)
- RJEDD--Used for SCRATCH of output data set (free spool space)
- RJEINQ--RJE Input Stream Data Set
- RJEOUTQ--RJE Writer's Output Stream
- RJERJCTS--RJE Job Status File
- RPT000--Batch reports to Tape File (Output Utility)
- SEC000--Basic Security disk-resident table entries
- SECURITY--Extended Security System File
- SIMCARDS--Front End Simulator parameter cards
- SMLOG--Statistical data and other output from Resource Management (thread dumps, etc.)
- SNAPDD--Snap dumps
- STATFILE--File Handler Statistics File
- STSLOG--System Tuning Statistics Report File
- SYSABEND--Used if abends are to dump all of storage
- SYSPRINT--Used by IJKTRACE, IJKPRINT, messages etc.

- SYSSNAP--ID=15 snaps (test mode)
- SYSSNAP2--ID=20 snaps (test mode)
- SYSUDUMP--Used if abends are to dump Intercomm region only
- THREDLOG--Backout-on-the-Fly facility DDQ File
- VRB000--Edit Control Table disk-resident entries (Edit Utility)

Additional system files with user-assigned ddnames for the following system facilities:

- BTAM output queues--names assigned in BTAMSCTS (BTAMQ)
- VTAM output queues--names assigned in VTAMSCTS (VTAMQ)
- Disk Message Queues--names assigned in Subsystem Control Table (PMIQUE)
- Front End Simulator input data sets (DDNAME=Terminal-ID), and simulated Local 3270 print files (SCRxxxxx)
- Page Facility (in addition to, or instead of, PAGES)
- Multiregion Support (MRS) disk message queues (DDQs)
- Dynamic Data Queuing (DDQ)
- Data Entry (INTBSKRM, INTBDTET, INTBDTnn)
- Autogen (AUTOGPCH)

6.6 FILE ATTRIBUTE RECORDS (FAR)

The FARs are read during File Handler initialization by the module IXFFAR after all internal DSCTs have been initialized, and the information from the FARs is encoded in DSCT appendages called File Attribute Blocks (FABs). The ddname of the FAR data set is ICOMIN; any card image data set accessible via QSAM GET is allowed.

Several types of specification may be made via the File Attribute Record input data set. They are:

- Defining a data set (by ddname) as input only. This means there will be no output activity allowed on the file. Any attempt to alter the file will be treated as an error by the File Handler. Coding this facility has exactly the same effect as coding ddname=R in the EXEC statement PARM field.

- Defining a BISAM data set (by ddname) as update only. The file will be opened for updates, not for inserts; an attempt to insert a record will be treated as an error. A core saving at least equal to the block size of the file is realized by this definition.

- Defining an alias for a data set (by ddname). This causes the File Handler to treat all calls referencing the file as if they referenced its alias. This technique is useful for mixing SYSOUT data from different routines using different hard-coded ddnames without reassembling. Two ddnames that are aliased must have the same DCB parameter specifications.

- Specifying that the file be opened at system startup. Opening DCBs or ACBs at startup reduces storage fragmentation; once storage is allocated for a DCB/ACB it will stay allocated for the rest of the run, unless the file is closed via a RELEASE request to the File Handler with the close option, or the FILE command. Opening the files at startup time segregates long-term storage holdings at the top of the region, hence eliminating fragmentation that would occur when files are opened at first access.

- Specifying that the high-level index of a BISAM file be kept in storage. Index level must be above the cylinder level.

- Specifying the ddname of a duplex output file. This causes all output operations against the primary file to be replicated automatically against the duplex output file. The result of this is to create an on-line backup copy of a critical sequential output file. This specification is allowed only if both files are sequential output. The duplex relationship is not symmetrical. For example, if DD2 is a duplex of DD1, then users selecting DD1 would have their output duplexed on DD2; but users selecting DD2 would not have their output duplexed on DD1. Do not use for the Intercomm log or x37 abend protected files.

- Marking a file permanently down if any I/O call to the File Handler results in a status code of C'1' or C'9'. When a file is marked down, then all calls to perform I/O will result in a status code of C'1', all SELECTs result in a C'9', and all RELEASEs complete normally. After all current users of a down file have released it, the file will be closed.

- Specifying Intercomm logic for BDAM exclusive control, rather than that of the operating system. A significant reduction in CPU requirement is gained, but no other region may request exclusive control on that file. Do not use DISP=SHR.

- Specifying Intercomm logic for ISAM exclusive control. The default assumed is that ISAM exclusive control updates are limited to BISAM access within a single region. This is most efficient and should apply to most users. Users whose requirements differ must specify the XCTL FAR attribute.

- Specifying Intercomm logic for VSAM cross-region shared and exclusive control for VSAM Shareoption 2 and 4 files.

- Overwriting of an existing sequential file (DISP=OLD or SHR)

- Forcing a wait state when NCP is reached for an output sequential file. For example, NCPWAIT is specified for the file with the ddname of QX1, and NCP=2 is coded on the JCL for QX1. A processing thread calls the File Handler which proceeds to write a block to the file. The File Handler does an internal wait (that is, exits to the Dispatcher until the ECB for the write is posted complete). The Dispatcher gives control to a second thread which also calls the File Handler to write a block to QX1. The File Handler issues that write and discovers that an earlier write to the same file is still outstanding, and that NCP for the file is 2. The File Handler issues a HARDWAIT; that is, the Intercomm main task goes into the wait state until the ECB for the first I/O is posted complete. Execution then resumes with the first thread made active and the second waiting on its I/O to the file.

- Preventing x37 abends for sequential output disk files and Intercomm log.

- Provide Local Shared Resources buffer support for VSAM files.

6.6.1 Coding the FARs

The coding format for FARs is:

```
ddname,attribute1,attribute2,...attributen.
```

FAR data may be coded from column 1 to 72; leading blanks are allowed; however, embedded blanks are not allowed.

A complete description of the FAR parameters and syntax for coding is contained in the File Recovery Users Guide. In the simple case of utilizing FARs to specify attributes not associated with File Recovery, the attributes are:

ALIAS=ddname

to define an alias for a data set, in order to route I/O operations to the alias data set. The originating ddname will have the FAR attributes of the alias file; no other attributes may be coded on this statement. x37 abend protection may not be requested for the originating ddname.

B37

applies only to sequential output disk files and the Intercomm Log (if to disk). Invokes an automatic facility to protect Intercomm from an x37 abend resulting from running out of space on this file. Installation specifications are in the section "Sequential Output Disk File Flip-Flop Facility" in this chapter.

COREINDEX

requests that the highest-level index of a BISAM file be kept in main storage. This option applies only to files large enough that the index hierarchy goes above the cylinder level. Cannot be used for IAM files.

DUPLEX=ddname

specifies the ddname of one or more duplex output files. When a duplex output operation is performed, the status code returned to the caller is C'0', if any output operation was successful. Otherwise, the status code from the first operation is returned.

NOTE: When duplex files are specified, all associated files are automatically flagged with the ERRLOCK attribute.

ERRLOCK

to force marking a data set permanently down, when any I/O call to the File Handler results in a status code of C'1' or C'9'.

ICOMBDAMXCTRL

to indicate that Intercomm logic is to be used for BDAM exclusive control, rather than that of the operating system.

LSR

cause a VSAM data set to be connected to the VSAM local shared resource pool at ACB OPEN time. The data set must be a VSAM data set which is currently loaded (LSR cannot be used to load a data set) and the resource pool must have buffers large enough to contain the data set's control intervals. The SPALIST BLDVRP parameter must be coded if LSR is coded. (See "Using a VSAM Local Shared Resources Pool" in Section 6.3.)

NCPWAIT

forces Intercomm into the wait state when the number of pending I/O's to a sequential file has reached NCP for that file. Intercomm becomes active again when the first I/O in the series is posted complete. This option is forced for INTERLOG, the Intercomm log data set.

NOTE: This option should be used with caution. Its improper use can cause the system to enter the wait state excessively and performance will deteriorate as a result. Concurrent I/O requests should be controlled by SYCTTBL parameters as described in Section 6.2.8.

OPEN={BASIC }
{QUEUED}
{BOTH }
{VSAM }

requests that the file be opened at startup time, rather than waiting for the first I/O request. The meanings of the subparameters depend on the file organization:

direct:
BASIC -- open BDAM DCB
QUEUED -- not applicable
BOTH -- not applicable

indexed sequential:
BASIC -- open BISAM DCB only
QUEUED -- open QISAM DCB only
BOTH -- open both BISAM and QISAM DCBs
(If IXFQISAM is used, the only valid specification is BASIC; BOTH or QUEUED will generate unpredictable results.)

sequential:
BASIC -- open BSAM DCB only
QUEUED -- open QSAM DCB only
BOTH -- open both BSAM and QSAM DCBs

VSAM
-- open VSAM ACB

READONLY

to define an input only data set.

UPDATEONLY

to define a BISAM data set allowing updates, but not inserts.

VSAMCRS

indicates that a VSAM Shareoption 2 or 4 file will be shared by more than one region in the same CPU and that updates will be performed by at least one region. Intercomm will augment VSAM shared file processing and provide read integrity for Shareoption 2 files and read/write integrity for Shareoption 4 files by means of OS ENQs: QNAME=INTERCOM, RNAME=VSAM-dsn (up to 44 characters). This FAR specification must be coded for the file in question for every region which will share the file. See also "Sharing VSAM Files Under Intercomm" in Section 6.3.

WRITEOVER

allows a complete rewrite of an existing physical sequential file (DSORG=PS, DISP=OLD or SHR). If this option is not specified, any data written to the file will be added at the end of existing data (that is, DISP=MOD assumed). If WRITEOVER and READONLY are specified for the same file, READONLY will be used and no writing to the file will be allowed. That is, READONLY suppresses WRITEOVER.

XCTL={QISAM}
{MULTIREG}

indicates that ISAM exclusive control updates are performed using QISAM, or from multiple regions. These specifications are functionally equivalent, and result in an OS ENQ at the file level. This is the least efficient means of assuring exclusive control, and can be avoided by restricting the updates to BISAM and to within a single region.

A typical FAR input data set might be:

```
//ICOMIN  DD *
MASTFILE,READONLY,COREINDEX
TRANFILE,UPDATEONLY,OPEN=BASIC
CUSTRECS,ALIAS=MASTFILE
INRECS,READONLY
/*
```

When ALIAS is specified, it must be the only attribute defined for a particular ddname. In other words, coding a FAR as:

```
TRANSIN,UPDATEONLY,ALIAS=INTRANS
```

is invalid syntax. When an ALIAS is defined:

```
ddname1,ALIAS=ddname2
```

any call to SELECT for ddname1 will cause subsequent calls to READ, WRITE, GET, PUT, GETV or PUTV to operate on ddname2. There is no need for a DD statement with ddname1 in the execution JCL; the ALIAS attribute overrides all specifications for ddname1. Any reference to ddname2 thus refers to ddname2 and the associated FARs for ddname2, if any.

To code the FAR for duplexed output:

```
ddname1,DUPLEX=ddname2
```

All WRITEs to ddname1 will be duplicated on the ddname2 data set. DD statements for both data sets must be present in the execution JCL.

IXFFAR will WTO images of each FAR read from ICOMIN in the course of processing. Thus, IXFFAR error messages (FRnnnI) may easily be related to an individual FAR. Once the FAR syntax is correct, you may suppress the image WTOs (80-character card images) by inserting the following card at the beginning of the FAR deck:

```
NOMESSAGES
```

FAR images will be suppressed; error messages will still be printed. This card must be the first record of ICOMIN.

NOTE: No internal DSCT is created for ICOMIN.

6.7 FILE HANDLER SERVICE ROUTINE SUMMARY

The following discussion provides a brief summary of File Handler functions. The specifics of calling procedures are discussed in greater detail in the Intercomm Programmers Guides.

The File Handler Service Routines are entry points within the File Handler Csect IXFMON01. Each service routine is called with a parameter list, as summarized in Figure 6-2. The File Handler determines specific operations to be performed, based upon the parameter list and DCB information. Parameters for File Handler calls are:

EXTDSCT (External Data Set Control Table): 12-fullword control block area supplied (but not modified) by the calling program.

FHCW (File Handler Control Word): four-byte option/status area initialized prior to call to request special functions and analyzed after call to determine status of operation.

Area: I/O area within calling program (ddname in the case of SELECT or RELEASE with close option).

Key: Requested key. For undefined record format, this field contains the record length. RRN for VSAM RRDS.

Block-id: Requested BDAM block-identification (RBN, TTR, or MBBCCHHR), or Relative Byte Address (RBA) for VSAM.

The SELECT function is called before the first access to a file in order to:

- Verify the availability of the file.
- Position the file for subsequent sequential access. A reuseable (direct access input) file will be repositioned to the beginning of the file for subsequent sequential retrieval. A nonreuseable (tape, direct access output) file will be positioned after the last record previously processed.
- Initialize and chain the External DSCT area.

The RELEASE function is called after the last access to a file in order to:

- Free any dynamically obtained buffers and control blocks
- Update file status tables and perform necessary housekeeping functions (unchain External DSCT area).

A special RELEASE function may be used after the above operation has been performed to close all shared control blocks for a given file if there is no currently outstanding operation being performed against the file by the system.

	EXTDSCT	FHCW	AREA	KEY	BLOCK-ID
SELECT	R	R	R	---	---
RELEASE	R	R	---	---	---
GET	R	R	R	I or U	---
PUT	R	R	R	U	---
READ	R	R	R	DI or U	D
WRITE	R	R	R	DI or U	D
RELEX	R	R	---	---	---
GETV	R	R	R	I	E
PUTV	R	R	R	I	E

Symbol	Indicates
R	Required parameter
I	Optional for ISAM or VSAM KSDS and RRDS files, otherwise invalid
D	Optional for Keyed BDAM file (extended search), required for random BDAM (instead of key)
DI	Required for Keyed BDAM, ISAM and DISAM files, otherwise invalid
U	Required for accessing a BSAM or QSAM file with undefined record format (DCB=RECFM=U)--record length
E	Required for address-accessed VSAM (RBA), instead of key
---	Invalid Parameter

Figure 6-2. File Handler Service Routine Parameter Summary

The GET function may be used to access the next sequential logical record from a QSAM or ISAM data set. In the case of application programs requiring QISAM retrieval logic, the GET function is used either to obtain the next sequential record for processing, or to locate a record by key and continue sequential processing with the located record. The File Handler may implement QISAM logic through the Basic Indexed Sequential Access Method (BISAM), transparent to the application program.

The PUT function is used to write or rewrite a record or a block. When creating a new QSAM data set, new records are written using the PUT function. When updating an existing QSAM or (logical) QISAM data set, the last record obtained by a GET function may be rewritten by calling the PUT function as the next operation upon the file.

The READ function is used to access physical blocks located within BSAM, BISAM, or BDAM data sets. For sequential data sets, each request for a READ function will process a physical block of records, which must be deblocked if necessary by additional programming. For indexed sequential data sets, each request for a READ function will locate (through an index search) the block containing the desired record, but will read only the single record specified by the key. For direct access data sets, each request for a READ function will process a physical block indicated by relative block number (RBN), relative track and record (TTR) or actual address (MBBCCHHR). In the case of BDAM without keys, the requested block is retrieved. In the case of BDAM with keys, the key search begins at the block specified, continuing until the search is complete. (Use of the extended search option is based upon DCB parameters including LIMCT.)

The WRITE function is used either to write the next sequential block in a new output BSAM data set, or to update the last block or record obtained by a READ function from a BSAM, BDAM or BISAM data set. The WRITE function can be used to insert records to a BISAM data set specified by key (position located through an index search). A record to be rewritten must have been previously read; an inserted record must not have been previously read. WRITE with key is the only function which will add records to an indexed sequential data set.

The GETV and PUTV functions are used to access VSAM data sets, requesting either sequential or direct access via key, relative byte address, or relative record number. A keyed access call for direct retrieval may provide either a generic (leading portion of a) key or a full key, and may specify either a search for an equal (generic) key or for the first greater-or-equal (generic) key. All retrieval calls are processed assuming that no update is to be performed unless the caller specifies otherwise. All calls allow for subsequent sequential access (key/RBA/RRN parameters not passed). The caller may switch back and forth from any access technique to another, provided VSAM rules are not violated; for example, keyed request against an entry-sequenced data set.

6.8 LOCATE FACILITY

An additional File Handler service routine, LOCATE, provides access to internal DSCT information for Assembler Language programs only. The LOCATE function is intended primarily for system use in altering the normal processing of a file. LOCATE provides access to data management control blocks used by the File Handler. A call to LOCATE will return data set specifications, error indicators and related information. This data, not available via other File Handler calls, can then be examined and/or judiciously altered.

A parameter list of variable length (depending on the amount of information required) is passed to LOCATE. The specific format is:

```
CALL LOCATE,(work-area,fhcw,dsctfld[,dcbfld,decbfld,iobfld]),VL,      X
     MF=(E,list)
```

Each parameter suffixed with "fld" must specify a fullword field. The address of the requested control block will be returned in each of these fields. The first three parameters are required; the remaining three are optional.

The parameters passed to LOCATE are defined as follows:

- work-area--pointer to a File Handler work area which may be:
  - -- A location containing a ddname. If dcbfld is specified, a public DCB is to be supplied (that is, an opened DCB to be shared by all users of this file is returned).
  - -- A File Handler work area (External DSCT) for a previously selected file. If dcbfld is specified, a private DCB is to be supplied (that is, the DCB returned is to be used only for I/O operations referencing the specified work area. The DCB will be closed when the work area is RELEASEd.)
- fhcw--the File Handler Control Word name. The completion status, in character format, will be returned in the first byte. Completion codes are:
  - -- C'0'--control blocks located
  - -- C'9'--file not located or improper type
- dsctfld--pointer to a location on a word boundary. The address of the internal DSCT will be returned here. The IXFDSCTA macro should be used to generate a Dsect.

- dcbfld--pointer to a location on a word boundary. The address of an opened Data Control Block (DCB) will be returned here (see work-area, above). If this parameter is the last coded, the DCB will be one for sequential access (GET/PUT). If additional parameter(s) follow, the DCB will be one for basic access (READ/WRITE). The record length, block size, and other data set characteristics are specified in the DCB. (For details, see IBM System Control Blocks, and macro instruction IHADCB.)

- decbfld--pointer to a location on a word boundary. Address of a Data Event Control Block (DECB) for basic access will be returned here. Contents of the DECB vary by acces method. For BDAM or BISAM, error status indicators are present in the DECB. (See IBM System Control Blocks.) This pointer must be hex zeros if file is not yet selected (address of ddname of the file is supplied for work-area). Address of DECB will not be returned in such a case.

- iobfld--pointer to a location on a word boundary. Address of the last used Input/Output Block (IOB) will be returned here. If no READ/WRITE operation has been performed, a zero value is returned.

LOCATE for VSAM, with dcbfld and decbfld specified, returns an opened ACB and a RPL address, respectively, even if the data set was converted from ISAM.

Programs which must refer to fields within the internal Data Set Control Table may be coded as in this example:

```
CALL    LOCATE,(work-area,fhcw,dsctfld),VL    Locate DSCT Entry
L       register,dsctfld                      Load Entry Address
USING   DSCT,register                         Make Fields Addressable
 :
 :
IXFDSCTA                                      Define DSCT Fields
```

Some of the fields which may be useful are:

DSCTDCBQ	pointer to QSAM or QISAM DCB
DSCTDCBS	pointer to BSAM or BISAM DCB
DSCTDCBD	pointer to BDAM DCB
DSCTDECB	pointer to DECB for BSAM, BISAM or BDAM
DSCTACB	pointer to VSAM ACB
DSCTRPL	pointer to VSAM RPL

Each of the above fields contains a significant value only if the high order bit of the word is 1 (use TM field,X'80').

6.9 FILE HANDLER OPTIONS

A number of File Handler options may be specified to further customize performance for an installation's needs. These options are specified via JCL, tables or conditional assembly of the File Handler.

6.9.1 Exclusive Control Time-Out

This option within the File Handler specifies a maximum time limit that a particular record or block may be held in exclusive control by a particular message processing thread. This time value represents the actual duration of message processing time between a request for exclusive control and the subsequent release of exclusive control by file update, or access to the same External DSCT representing a message thread's access to a file. This value is a constant defined within the member IXFHND01. The standard setting represents two minutes for exclusive control at the physical block level, ten minutes for exclusive control at the data set level. An Intercomm System Engineer should be consulted to adjust this value. This feature does not apply to VSAM files.

6.9.2 Conditional Assembly of the File Handler

Several File Handler options are specified by global settings and subsequent conditional assembly of File Handler modules. The globals are defined in the member INTGLOBE and specified in the member SETGLOBE.

The following members must be reassembled and linked:

IXFHND00,IXFHND01

If the &VSAM or &VSISAM globals are set to 1, the modules must be reassembled whenever a new version of VSAM is installed (setting &VSISAM to 1 internally forces &VSAM to 1). The globals are illustrated below.

Global Definition (INTGLOBE)	File Handler Function	Default Setting (SETGLOBE)
&ISAM	Allow ISAM access	SETB 1
&VSAM	Allow VSAM access	SETB 1
&VSISAM	Allow VSAM/ISAM compatability	SETB 1
&AMIGOS	Allow AMIGOS file access	SETB 0
&IAM	Allow IAM file access	SETB 0
&DYNALOC	Allow Dynamic File Allocation	SETB 1

Note: the &DYNALOC global specifies program usage of the special feature described in Dynamic File Allocation for data sets not defined in Intercomm execution JCL and is unrelated to FILE command processing.

6.9.3 Subtasked GETs

The File Handler has a generalized subtasking facility to allow all GETs (both QSAM and QISAM) to be overlapped with other Intercomm processing. The reason for the facility is that the GET macro does not return control to a task, when it is issued to retrieve a record, until the record is obtained. Without subtasked GETs, the File Handler, and therefore Intercomm, would go into a wait state whenever a GET was issued. Using a subtask to perform the GET allows Intercomm to continue processing while only the subtask remains in the wait state. (The operating system must have the IDENTIFY feature and multitasking capability.) The module ICOMTASK must be included in the linkedit.

At startup a user-specified number of generalized subtasks must be created, which will issue the GETs, when called upon to do so by the main Intercomm task. The user specifies the number of general subtasks to be created in the TASKNUM parameter of the SPALIST macro.

Each subtask executes GETs serially. Therefore, with only one subtask, all GETs will be overlapped with other processing, but not with each other. Specifying a larger number of subtasks allows the GETs themselves to be executed concurrently.

File Handler closedown (IXFMON09) detaches all the subtasks.

6.9.4 IXFDSCTA Options

The four bytes beginning at displacement 4 from the start (header) of the IXFDSCTA Csect are the "options" bytes for the File Handler, and can be coded to give the various options listed in Figure 6-3 either by the appropriate hex digits coded in the OPTIONS parameter in the IXFDSCTA macro, or can be patched into the Intercomm load module at execution time (Csect name: IXFDSCTA, displacements: 4, 5, 6 and 7).

Options	Code
Do not overlap BISAM (single-thread)	XX 40 XX XX
Allow unit record devices in DSCT	XX XX XX 80
Disable automatic initialization	XX XX XX 01
GET: Time-slice option	XX XX 80 XX
Single-thread PS READs	XX 80 XX XX
BDAM: Prevent exclusive control	XX 02 XX XX
BDAM: Force exclusive control	XX XX 02 XX
BDAM: Single-thread nonexclusive READs	XX 20 XX XX
BISAM: Prevent exclusive control	XX 04 XX XX
BISAM: Force exclusive control	XX XX 04 XX
BISAM: Bypass RE-READ option (exclusive control)	XX 01 XX XX

Figure 6-3. IXFDSCTA Options

6.9.5 User-Specified DCBs

The File Handler provides the minimum necessary control blocks and options for processing a file. Certain increased performance processing options require that the user supply the data control block. Such nonstandard options include resident master indexes, main storage work areas for ISAM data sets, etc.

DCBs should be supplied to the File Handler before they are required for I/O operations. The user startup exit (USRSTRT1) in Intercomm is a convenient point at which to supply DCBs.

The user routine must be written in Assembler Language and use standard linkage conventions. The following statement must precede the first named or unnamed control section in the user's module:

```
PUNCH '         REPLACE IXFDSCTA'
```

A COPY statement for the system library member IXFDSCTB must be coded in the user routine. This will generate both a Csect (IXFDSCTA) and a Dsect (labeled DSCT) in the assembled routine. The REPLACE statement generated by the PUNCH command will, during link editing, delete the Csect.

To supply a DCB to the File Handler, call the LOCATE function. This will store the address of the Data Set Control Table (DSCT) in a user-supplied area:

```
CALL LOCATE,(ddname,fhcw,dsctfld),VL,MF=(E,list)
```

The return status (first byte of fhcw) from LOCATE must be tested before proceeding. If the code is nonzero, the file is not available for accessing; no DCB can be supplied.

If the file is available, dynamic main storage should next be acquired (STORAGE macro). The number of bytes obtained should be the length of the user-supplied DCB. The user DCB can then be constructed in this dynamic area. Or, if constructed elsewhere, the DCB can now be moved to the area. Unused bytes at the beginning of the DCB must be copied into the dynamic area. The symbol naming the DCB macro instruction must correspond to the first byte of the area. The DCB need not be opened; however, an OPEN macro can be issued if desired.

Having created a DCB in dynamic storage, load the DSCT address returned by LOCATE into a register. The statement:

```
USING DSCT,register
```

should be in effect at this point. The proper DSCT field to contain the address of the created DCB can now be addressed; this field will be one of the following:

- DSCTDCBQ for a QSAM or QISAM DCB
- DSCTDCBS for a BSAM or BISAM DCB
- DSCTDCBD for a BDAM DCB

Bit zero of the field DSCTDCBx should now be tested:

```
TM    DSCTDCBx,X'80'
```

If the bit is on, a DCB already exists; no new DCB can be supplied. If the bit is off, place the address of the user DCB in the field DSCTDCBx. Next, or (OI) X'80' into the first byte of DSCTDCBx.

The File Handler will now use the supplied DCB for subsequent I/O operations. Use will continue until closing of the data set is executed explicitly or by implication. (Closing of the data set would be implied by a RELEASE with the close option.) Upon explicit or implied closing:

- The supplied DCB will be closed.
- The main storage area occupied by the user DCB will be freed. (Storage freed will correspond in length to the standard DCB for the particular access method.)

A new DCB must be supplied if subsequent processing is desired.

Figure 6-4 illustrates a possible user-coded routine to supply a user DCB to the File Handler.

```
            TITLE       'USER SUPPLIED BISAM DCB TO FILE HANDLER'
            PUNCH       '         REPLACE IXFDSCTA'
USRCSECT    CSECT
            .
            .
            .
            CALL        LOCATE,(MYDCB+40,STAT,ADDRDSCT),VL
            CLI         STAT,C'0'
            BNE         NODDNAME        ERROR. DDNAME NOT DEFINED
            L           2,ADDRDSCT
            USING       DSCT,2
            TM          DSCTDCBS,X'80'
            BO          DCBINUSE        ERROR. DCB ALREADY IN USE
            LA          0,DCBLEN
            STORAGE     LEN=(0),SYS=YES...
            MVC         0(DCBLEN,1),MYDCB
            ST          1,DSCTDCBS
            MVI         DSCTDCBS,X'80'
            .
            .
            .
MYDCB       DCB         DSORG=IS,MACRF=(RUS,WUA),DDNAME=MYDD,...
DCBLEN      EQU         *-MYDCB
ADDRDSCT    DC          F'0'
STAT        DC          CL4' '
            COPY        IXFDSCTB
            END
```

Figure 6-4. Sample User-Supplied DCB

6.10 FILE HANDLER STATISTICS REPORT

The optional program IXFRPT01, when included in the resident Intercomm linkedit, produces statistical reports of File Handler usage. Reports on all files accessed are periodically written to SYSPRINT. Data for these reports is maintained in the internal DSCT and optionally on the disk data set STATFILE. The printed figures reflect cumulative file activity; that is, total activity since Intercomm startup or the last reinitialization of STATFILE (if defined). A second entry point, IXFRPTIQ allows on-line inquiry via the FHST command. To allow terminal commands, a SYCTTBL for a resident subsystem must be defined in the SCT with entry point IXFRPTIQ, along with the appropriate verb definition for FHST in BTVRBTB. In this case, a terminal operator asks for statistics for a particular file or all files; the requested information is returned to his remote location. See System Control Commands.

The general layout of the File Handler Statistics Report is shown in Figure 6-5. The leftmost column lists ddnames of all accessed files in the system. The second column shows how many times each file has been selected. Columns three through six show the number and type of accesses to the file (less detail may be obtained; see below). At the right hand side of the page, total accesses per file are shown. AVERAGE shows the average number of accesses per SELECT. (For SYSPRINT, which has no SELECTs, no average is calculated.) At the end, a summary line showing total activity for all files is printed.

DATE 83.056 FILE HANDLER STATISTICS REPORT TIME 10:19:38.7 PAGE 1

DDNAME	SELECT	GET	PUT	READ	WRITE	TOTAL	AVERAGE
INTERLOG	12	0	0	0	43	43	3.58
STSLOG	2	0	48	0	0	48	24.00
INTSTOR2	9	0	0	11	0	11	1.22
SYSPRINT	0	0	521	0	0	521	
SMLOG	5	0	649	0	0	649	129.80
WAGEMSTR	14	0	0	41	27	68	4.86
STOKFILE	4	4	0	0	0	4	1.00
PARTFILE	6	0	0	6	0	6	1.00
PMIQUE	5	0	0	0	0	0	0.00
RCT000	9	0	0	9	0	9	1.00
SUMMARY	66	4	1218	67	70	1359	20.59

Figure 6-5. File Handler Statistics Report

A terminal request for statistics for a particular file produces one line of output formatted exactly as a body line in the SYSPRINT report.

The number of statistics options is globally specified via the &FHSTATS global in SETGLOBE (released as 5 for selects, gets, puts, reads and writes). A corresponding number of fullword buckets are generated at the end of each internal DSCT entry for each file accessed via the File Handler.

If less detailed statistics options are desired, change the &FHSTATS global value to 3 (selects, inputs, outputs) or 2 (selects, accesses) and reassemble and relinkedit the following modules (if used):

AMGINTFC	IXFCTRL	IXFRPT01
CFMSINTF	IXFDISAM	IXFRVRSE
DDQMOD	IXFDSCTn	IXFSNAPL
DDQSTART	IXFDYALC	IXFVERF1
INTSEC00	IXFFAR	IXFVSCRS
INTSTORF	IXFHND00	LOGPUT
IXFB37	IXFHND01	PMISNAP1
IXFCHKPT	INFLOG	PMITEST
IXFCREAT	IXFQISAM	RMPURGE

IXFRPT01 is initially dispatched by startup. Thereafter it dispatches itself on the time interval specified by the global &RPTINTV in the member SETGLOBE. As released, the report is produced at ten minute intervals. If this value is changed, reassemble STARTUP3. The time interval for dispatching IXFRPT01 can be changed during Intercomm execution. This is accomplished by issuing a DISPATCH macro instruction for IXFRPT01. The address of the new time interval (in timer units) is passed in register 1. The IXFRPT01 rescheduling cycle can be halted by dispatching IXFRPT01 on a time interval of 0. The dispatching of IXFRPT01 is stopped by closedown. A final File Handler Statistics Report is produced, but IXFRPT01 is not rescheduled. As written, IXFRPT01 supports up to 1596 files (internal DSCT table entries). If more are defined in the Intercomm JCL, change the value coded for FLAGTBL in the save/work area to the number of files divided by 8 (each bit represents one file).

At each execution of IXFRPT01, statistics are retrieved from internal File Handler tables. If defined, the STATFILE disk data set is also updated. Updating consists of summing figures from the internal tables with those already accumulated on STATFILE. The internal tables are then zeroed out. A report reflecting the total figures on STATFILE is then written to SYSPRINT. The number of lines per print page may be modified by changing the global setting for &PAGELIN in IXFRPT01 (default=55).

When entered via an inquiry from a terminal, IXFRPT01 also retrieves required data from STATFILE, if defined. Statistics in internal tables are added in and the on-line report is sent to the requesting terminal. STATFILE is not updated, nor are the internal tables zeroed. Statistics for all files, even if never selected, are displayed when an FHST command without a ddname is entered. The number of lines per display (including headers) depends on the terminal line length (minimum=80) and buffer size (defaults to 24).

6.10.1 File Handler LSR Statistics

In addition to the normal File Handler statistics, when a Local Shared Resources pool is present, statistics on all of the buffer pools in the resource pool may be gathered. Since the buffer pools are shared among data sets, the statistics are reported on a pool rather than data set basis. Information about the individual data sets using the pool is displayed as usual in the data set section. To implement LSR statistics, the &FHSTATS global must be set to 5 and the modules listed in the previous section reassembled if &FHSTATS was less than 5.

The following statistics are displayed for Local Shared Resources (see illustration):

BFR SIZE -- one line of statistics for each pool size in the resource pool.

REQ REJ -- number of requests (requiring a given size buffer) which were rejected because there were not enough buffers of that size to satisfy it (the amount reflects all retries of rejected requests).

BFRFND -- number of requests satisfied by data found in a buffer of that pool size (no I/O needed to satisfy request).

BUFRDS -- number of reads to bring data into a buffer of that pool size.

STRNO EX -- number of requests that were rejected because no placeholders were available; reflects all retries of rejected requests (kept for resource pool as a whole).

STRMAX -- maximum number of placeholders in use at any one time (accumulated for the whole resource pool, not on a buffer pool size basis, because placeholders are assigned to the resource pool as a whole).

Note that, when LSR is used, VSAM attempts to use buffers that are the size of a data set's control interval(s). If no buffer pools of that size exist, VSAM uses the next larger size. Thus if X and Y are pool buffer sizes and Z is a control interval size such that Z is larger than X but smaller than Y, buffers for control interval size Z will be taken out of the pool of size Y buffers. When the "request rejected" statistics are displayed, they will show the number of requests rejected for each control interval size rather than buffer size. One should be aware, however, that the buffer pool that had no buffers available for the request was that of the next larger size.

Also, when a key-sequenced data set is used with LSR, both the data component and the index component share buffers from the LSR pool. If the data and index component have different CI sizes, both buffer sizes must be available in the pool (with the exact sizes or the next higher size) and buffers must be free in the pool for the request to be satisfied. Thus, a request may be rejected if either buffer pool size is temporarily out of buffers. VSAM gives no indication as to which buffer size was unavailable, so when a KSDS request is rejected, this is reflected in the statistics under both CI sizes. This fact must be considered when making adjustments to the LSR pool based on the File Handler statistics.

DATE 83.056 VSAM LSR POOL STATISTICS TIME 10:19:38.7 PAGE 2

BFR SIZE	REQ REJ	BFRFND	BUFRDS
512	0	3	1
1024	0	0	0
2048	0	0	0
4096	0	0	0
8192	0	0	0
12288	0	3	1
16384	0	0	0
20480	0	0	0
24576	0	0	0
28672	0	0	0
32768	0	0	0
STRNO EX	0	STRMAX	1

A LSR statistics display may be requested at a terminal via the FHST command.

6.10.2 Creating the File Handler Statistics File (STATFILE)

STATFILE must contain a number of records at least one greater than the maximum number of files in the system. The STATFILE record consists of an eight-byte ddname and four bytes for each statistic. Totals on STATFILE are cumulative and may represent daily or weekly totals, etc., that is, cumulative for several Intercomm jobs. A schedule for reinitializing STATFILE should be established to meet the needs of the particular Intercomm installation.

To create STATFILE, use the CREATEGF utility (see Chapter 12), for example:

```
//stepname   EXEC     PGM=CREATEGF
//STEPLIB    DD       DSN=INT.MODREL,DISP=SHR
//SYSPRINT   DD       SYSOUT=A
//STATFILE   DD       DISP=(NEW,KEEP),DSN=STATFILE,SPACE=(TRK,(2,1)),
//                    UNIT=unit,VOL=SER=volume,
//                    DCB=(DSORG=DA,BLKSIZE=560)
//SYSIN      DD       *
F STATFILE   xx
//
```

where xx is the number of blocks to create based on the value of n below.

At Intercomm execution time, the following DD statement must be present for STATFILE:

```
//STATFILE   DD      DSN=STATFILE,DISP=(OLD,KEEP),
//                   VOL=SER=volume,UNIT=unit,
//                   DCB=(DSORG=PS,BLKSIZE=560,LRECL=n,RECFM=FB)
```

where n is:

16 -- if only collecting SELECT and ACCESS statistics (&FHSTATS set to 2).

20 -- if SELECT, INPUT, and OUTPUT statistics are to be collected (&FHSTATS set to 3).

28 -- if SELECT, GET, PUT, READ, and WRITE statistics are to be collected (&FHSTATS set to 5).

The SYSPRINT data set must be specified in the Intercomm execution JCL as follows:

```
//SYSPRINT   DD       SYSOUT=A,
//           DCB=(DSORG=PS,BLKSIZE=141,LRECL=137,RECFM=VA)
```

The SYSPRINT file should be blocked for optimum throughput since PUTs to the file are not overlapped (see Section 4.2.2).

6.11 USING THE FILE HANDLER SEPARATELY FROM INTERCOMM

The File Handler may be used independently of any other Intercomm components, if desired, by linkediting the modules BATCHPAK, IXFHND00, and IXFHND01 (preceded by IXFDSCTn, if a separate Internal DSCT table is needed) with any application program. File Handler interface coding is exactly the same as used in on-line programs; the same entry points (including SELECT and RELEASE) are called, and the same parameters are used. Unresolved external references, beginning with 'IJK' (Dispatcher entry points), will be bypassed during execution. However, if a VSAM or x37 abend protected file is being processed, IJKDSP01 must also be included.

When the File Handler is used off-line by a processing program, that is, used separately from Intercomm, the initialization routine (IXFMON00) may be called prior to any File Handler processing; however, this module will be automatically called, if necessary, when the File Handler is first used in a job step. If errors occur during initialization, IXFMON00 returns to the operating system with a return code of 16. The File Handler will not use any "unresolved" entry points to other Intercomm modules if these are not available during execution. At the end of processing, the batch program should issue a second call to RELEASE with the close option, to close the file (required for VSAM).

FAR processing will also be performed if IXFFAR is included in the linkedit, along with a DD statement for ICOMIN (and FAR statements) in the execution JCL. For VSAM file processing, if any of the following FAR options are used, additional linkedit considerations apply:

- LSR - include an INTSPA (SPALIST macro assembly with EXTONLY=BOTH and with LSR pool definitions; do not specify the FIX option) before the include statement for BATCHPAK
- VSAMCRS - include KEYFLIP (before the include for IJKDSP01), PMINQDEQ, IXFVSCRS, and then INTSPA before the include statement for BATCHPAK; INTSPA must contain a SPALIST macro assembled with a SETGLOBE in which the Intercomm Interregion SVC was specified (&MRSVC not 13).

6.11.1 Using the File Handler in LINKPACK for Batch Programs

To interface a batch program to the File Handler in the Intercomm Link Pack Module in the Link Pack Area, the following steps are necessary:

1. Prepare the Link Pack Facility as described in Chapter 7.
2. Write an interface routine (INTERFAC) to:
   - CALL MULTISPA
   - CALL LPSTART
   - CALL BATCHPGM

   where BATCHPGM is the entry point of the user batch program.
3. Include in the linkedit
   - INCLUDE SYSLIB(MULTISPA)
   - INCLUDE SYSLIB(LPSTART)
   - INCLUDE SYSLIB(LPINTFC)
   - INCLUDE SYSLIB(IJKDSP01) (if VSAM file accessed)
   - INCLUDE SYSLIB(IXFDSCT1)
   - INCLUDE SYSLIB(IXFHND00)
   - INCLUDE SYSLIB(IXFFAR) (if FAR options used)
   - INCLUDE SYSLIB(BATCHPAK)
   - INCLUDE SYSLIB(BATCHPGM) User Batch Program
   - INCLUDE SYSLIB(INTERFAC) User Interface Routine
   - ENTRY INTERFAC

6.12 DISAM--A FILE HANDLER ACCESS TECHNIQUE

The Indexed Sequential Access Method (ISAM) offers certain conveniences in the maintenance of files that must afford random access to individual records, but also must be maintained in a particular sequence for either batch processing or scanning purposes. These conveniences, however, are accompanied by serious disadvantages with respect to accessing efficiency and parallel program execution, both of which are major factors in the throughput capacity of on-line systems.

In particular, every ISAM operation except rewriting a block involves an index level search and data read. All are performed through time-consuming sequentially chained physical access operations, which monopolize the hardware I/O channel throughout their duration, even during periods when no productive data transfer is occurring. The high minimum elapsed time necessary to complete one logical function, especially where overflow chains are present, is one serious drawback of ISAM files, while the inability to overlap the physical accesses of different programs is another obstacle to total machine efficiency. Finally, the fact that addition of new records may cause relocation of other records places a severe restriction on the ability of different programs to process the same file in other than a batch mode--one program at a time.

Where maintaining a sequence is unnecessary or avoidable, a direct-access (BDAM) file is preferable to an indexed organization. Although this may require more planning and coordination among designers and programmers, a worthwhile conservation of machine usage by programs may be realized. ISAM provides the most carefree random access programming and will function even if proper record distribution and file maintenance are neglected, but the price of automatic arrangement of one's unique data base by a standard program may be a serious degradation of performance.

Where maintaining a sequence is necessary, however, a compromise solution may be more efficient for some applications. Scanning ability and ready insertion and deletion of records may be retained while channel interference, record relocation and parallel processing restrictions may be reduced through a combination of ISAM and BDAM files forming a data base. In this concept, known as DISAM, the ISAM portion of the data base contains only keys and pointers to the remaining data contained in the BDAM portion of the data base. (See Figure 6-6 for an illustration of the DISAM organization.)

While the reduced total size of the ISAM file may slightly decrease average index search time, and the reduced record size may decrease buffer space and overflow area requirements, the average time to access a data record generally will increase. This is due to the termination of the ISAM operation when a pointer to the data record is read, and another BDAM access is required to read the data. The offsetting gains in performance, however, are realized in several ways.

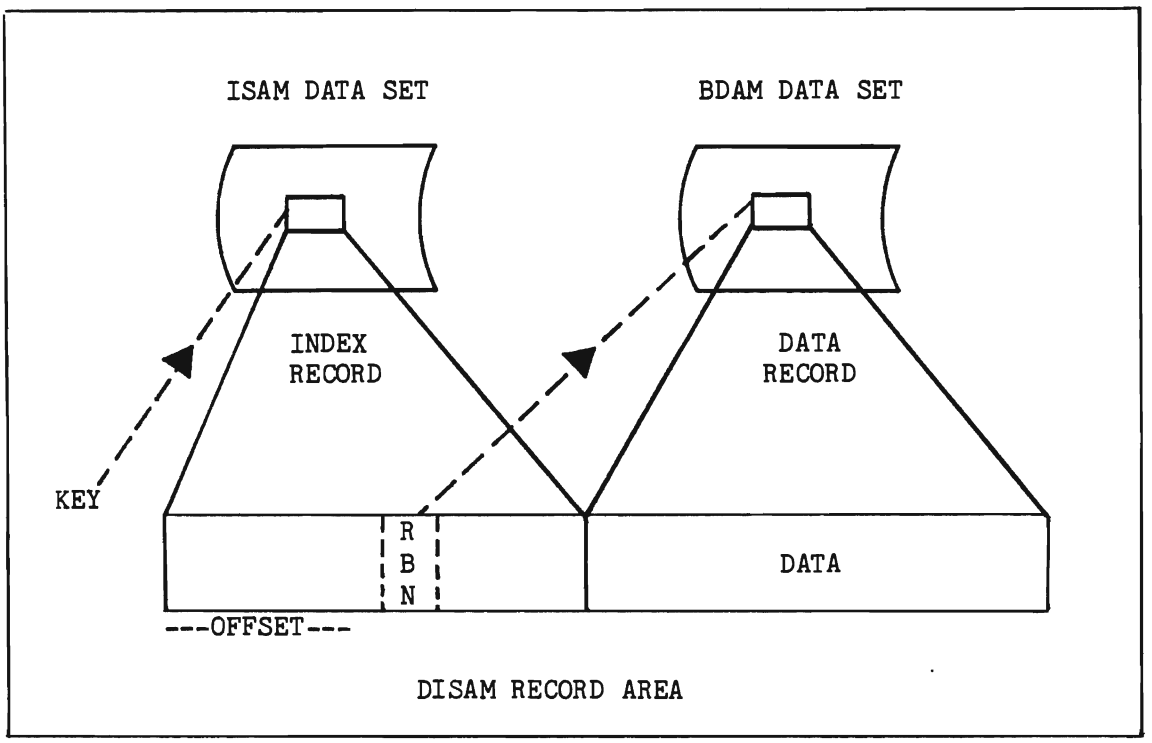


Figure 6-6. DISAM Data Base Structure

Overflow chains of lengthy data records are eliminated, reducing performance degradation as the file content changes. More significantly, a program may reserve exclusive control of a particular data record while other programs process different records of the same file; this, in turn, results from data records not being relocated when other records are added, thereby permitting simultaneous access to the same data base by several programs. For applications where the volume of file updates and additions is high, this factor may be significant in the file design decision.

6.12.1 DISAM File Handler

To simplify coding standard DISAM operations, a DISAM File Handler (module name: IXFDISAM) is provided, with functions and calling sequences similar to those of the standard File Handler; the DISAM module in turn calls the standard modules for basic I/O functions.

The terms defined below are used in the subsequent function descriptions:

- Work-area

  This is a twenty-seven-word main storage area beginning on a word boundary, in which the DISAM File Handler places and maintains both control and status information during processing. This area is initialized by the File Handler when the data base is selected, and its contents are used by the File Handler in performing subsequent operations.

  NOTE: If an application program is to be reenterable, main storage for all work areas and nonconstant data must be dynamically obtained.

- Status-area

  This is a four-byte main storage area consisting of:

  --Byte 1--completion status

  --Byte 2--option byte

  --Bytes 3 and 4--offset value, when used

- The completion status is stored by the DISAM File Handler after every operation, and will be one of the following:

  --C'0'

  Operation completed normally.

  --C'1'

  An I/O error (such as data check) occurred during the operation.

  --C'2'

  During input the specified record could not be found, the end of the file was reached during sequential input, or the offset value or record pointer are invalid or specify no record.

  During output the record being added has the same key as an existing record, or the record being rewritten was already released (by time-out) from exclusive control.

--C'9'

The file cannot be accessed. Possible causes: an omitted DD statement (or omitted DSORG in the statement), a misspelled file name, output operation on a read-only file, permanent file error (such as overflow area full), work area was overlaid, or format is not fixed length.

The option byte must be inserted by the calling program before each call for a function where options can be specified. The options and their codes are given in the individual function descriptions which follow.

The offset value is a binary quantity (relative to zero) used to locate the BDAM record pointer within the ISAM record. A default offset value is specified when the file is selected; thereafter any operation where no value is specified will use the default value. This field is two bytes in length, and must be filled with blanks, binary zeros (low-values), or hexadecimal FFs (high-values) when no value is to be specified.

- Record-area

A main storage area of sufficient size to contain both one index record (ISAM logical record) and one data record (BDAM record). Figure 6-6 indicates the relationship of these records to the record area. The index record begins in the first byte of the record area and its length is one logical record length (LRECL). The data record begins in the next byte after the index record and its length is the block size (BLKSIZE) of the BDAM file.

Within the index record are one or more record pointers, which are three-byte binary fields specifying the relative block number (RBN) of a record in the BDAM file. Valid RBN values are 1 through one less than the number of blocks in the file. Each record pointer field must be filled with either blanks, binary zeros (low-values), or hexadecimal FFs (high-values) when no data record is to be specified.

To locate a specific data base record, the DISAM File Handler first reads the index record having the specified (or next higher) key, locates a record pointer within the logical record using the offset value supplied, and reads the indicated data record. The record area then contains both the index and data portions of the file item desired.

Although the index record may contain data as well as record pointers, the parallel processing advantage of DISAM is realized when the BDAM data set contains all the updatable information pertaining to the file. The use of several record pointers in the index record permits one ISAM data set to be the index to several BDAM data sets, or permits an index record to point to several data records in the same file. The actual record formats, file usage and file creation are freely established by the user to suit the application, while the DISAM File Handler assists in performing all processing operations as required. DISAM is implemented on-line by coding DISAM=YES for the ICOMLINK macro when generating the Intercomm linkedit control statements.

6.12.2 DISAM File Record Formats

The DISAM access method allows for fixed or variable length records in both the index and data files. All combinations of record formats are allowed; that is:

- Fixed-length ISAM and fixed-length BDAM
- Fixed-length ISAM and variable-length BDAM
- Variable-length ISAM and fixed-length BDAM
- Variable-length ISAM and variable-length BDAM

In addition, the ISAM file may be blocked or unblocked.

6.12.3 BDAM Records

For variable-length BDAM records, the first four bytes consist of a block length field containing a two-byte binary length followed by two bytes of zeros; the next four bytes contain a record length field consisting of a two-byte binary length (four less than the block length) followed by two bytes of zeros; that is, there is an eight-byte prefix for every variable-length BDAM record. Input records received by the user will contain the length fields, and output records must be initialized by the user with the length fields containing the correct length. The four bytes for each length field are included in the lengths, and the eight bytes for both fields are included in the BLKSIZE value for the file. A relative track and record number (TTR) must be used as the BDAM record pointer in the ISAM record for variable-length BDAM records. For a description of relative track addressing, see the appropriate Intercomm Programmers Guide.

6.12.4 Size of Record Area for Variable Length Records

The record area must be large enough to contain the maximum size record for each file (the value given as LRECL for the ISAM file and as BLKSIZE for the BDAM file).

The four bytes for the length field are included in the length value and in the LRECL value for the file.

The DISAM user must allow for these four bytes in the offset to the BDAM record pointer within the ISAM record (that is, if the BDAM record pointer begins in the third data byte of the ISAM record, the offset is two for fixed-length ISAM records and six for variable-length ISAM records).

6.12.5 ISAM Offset Value

For variable-length ISAM records, the first four bytes of the record are the length field containing a two-byte binary length followed by two bytes of zeros. Input records received by the user will contain the length field, and output records must be set up by the user with the length field containing the correct length.

6.12.6 DISAM Operations

Each operation is invoked by a CALL, in a form suitable to the programming language used, to the entry point shown in upper case letters. The parameters passed in the CALL are listed between parentheses and are replaced by the name or address of the corresponding field.

NOTE: In Assembler Language, the VL operand must always be specified.

To select a file for accessing, invoke this operation:

```
DISEL (work-area,status-area,name-area)
```

- Work-area

  A twenty-seven-word area aligned on a word boundary. The area is initialized at this time and must be specified in all subsequent function calls.

- Status-area

  --option byte (2nd byte of status-area)

  The data record delete option is specified if this field contains a character 'F'. A record in the data file is considered deleted if the first byte of the record contains X'FF'. An attempt to read a deleted record results in a no record found condition. If the delete option is not specified at SELECT time, no deletion of records is possible in the data file.

  --Offset value (last two bytes of the status-area)

  The value specified at this time will be used as a default value in any subsequent function call that does not specify an offset. If no offset is specified at this time, a default value of 1 is assumed.

- Name-area

  A sixteen-byte area, the first eight bytes containing the ddname of the ISAM data set and the next eight bytes containing the ddname of the BDAM data set. The ddnames are the names of the job control DD statements defining the file for the current job step. A ddname less than eight bytes must be left-justified with remaining bytes blank.

For random input, invoke this operation:

```
DIREAD (work-area,status-area,record-area[,key-area])
```

- Status-area -- option byte

  Leave the option byte blank if none of the special processing options below are required:

  1 or X -- Hold the data record in exclusive control until subsequently released or rewritten by this program. No other program requesting exclusive control can read or update the same data record until it is released.

  2 or I -- Read the index record only and do not use the offset value.

  3 -- Read the index record only, and retain exclusive control of the index data set until released.

4 or D -- Read a data record only. The record area is assumed to contain a previously read index record and the offset value is used, as previously specified, to obtain the data record pointer. The key area parameter must be omitted when specifying this option.

5 -- Combine the functions of 1 and 4.

7 -- Read both index and data, and retain exclusive control of both the index data set and the data record until released or rewritten.

NOTE: Performance is degraded when exclusive control of the index data set is requested. This is normally not necessary, even when record pointers are inserted or replaced, unless variable data fields are made part of the index record.

- Status-area--offset value

  This value is used to locate in the index record, the record pointer to the data record. If no value is specified, a default value is used (see DISEL operation).

- Key-area

  This parameter must be supplied unless reading a data record only. The key area is as long as the key length (KEYLEN) of the index data set, and contains the exact record key to be located.

For random update, you may invoke the following operation after a random input operation to replace the record last read:

```
DIWRITE (work-area,status-area,record-area[,key-area])
```

- Status-area--option byte

  blank -- Update the data record. If the previous operation retained exclusive control of the index and/or data records, control is released.

2, 3, or I -- Update the index record. If the previous operation retained exclusive control of the index record, the record is rewritten. Otherwise, the index record is reread and only the new record pointer is inserted before the record is rewritten.

7 -- Combine the above functions.

- Status-area--offset value

  If the data record is updated, the offset value is used to locate the record pointer in the index record area. If the index record is rewritten, the new record pointer in the index record-area replaces the old record pointer.

- Record-area

  An index record must previously have been read or moved into this area. The record pointer associated with the data record may be changed if the index record is being rewritten; in this case, the current index record may be reread into this area. An updated or new data record is written from the data portion of the area.

- Key-area

  This parameter must be provided if the index record is to be rewritten; otherwise it must be omitted.

For sequential input, invoke the following operation (normally used to obtain the next sequential data base record):

```
DIGET (work-area,status-area,record-area[,key-area])
```

- Status-area

  Option byte and offset value are the same as those listed for random input.

- Key-area

  Supply this parameter only when beginning or restarting sequential processing at the first record having an equal or higher key. When this parameter is omitted in the first operation following file selection, processing starts at the beginning of the file.

For sequential update, you may invoke the following operation after a sequential input operation to replace the record last read:

```
DIPUT (work-area,status-area,record-area[,key-area])
```

- Status-area

  Option byte and offset value are the same as those listed for random update.

- Key-area

  This parameter must be provided if the index record is to be rewritten; otherwise it must be omitted.

For random addition of records to the file, invoke the following operation:

```
DIADD (work-area,status-area,record-area,key-area)
```

- Status-area--option byte

  blank -- Write the data record. Add a new index record using the specified key.

  R -- Write the data record. Update the index record having the specified key.

- Status-area--offset value

  The offset value is used to locate the record pointer in the index record area. If no record pointer is specified, the data record is written in the next available block and the index record is appropriately added or updated.

- Key-area

  For ADD operations there must not be a duplicate key in the index data set. For UPDATE operations, there must be an index record with the specified key.

To delete records from the file, the delete flag will be set, provided the delete option has been specified, in both the index and the data records. Alternatively, the user may mark a data record as deleted by placing hexadecimal 00s or FFs in the record pointer field of the index record. To perform a delete, invoke the following operation:

```
DIDEL (work-area,status-area,record-area[,key-area])
```

- Status-area--option byte

  I -- Delete the index record having the specified key.

  D -- Delete the data record, using the record pointer value in the index portion of the record area.

  blank -- Delete the index record having the specified key. Delete the data record using the record pointer value obtained from the index record.

- Status-area--offset value

  The offset value is used to locate the record pointer in the index record-area.

- Key-area

  The key field is required if the index record is to be deleted; otherwise it must be omitted.

  NOTE: Deletion of a record is performed by rewriting with X'FF' moved into the first byte of the record.

To release exclusive control of a record when no update is to be performed, invoke the following operation:

```
DIRELEX (work-area,status-area)
```

- Status-area--option byte

  I -- Release exclusive control of the index record only.

  D -- Release control of the data record only.

  blank -- Release control of both index and data records.

To release a file from processing, invoke the following as the last operation on a file:

```
DIREL (work-area,status-area)
```

6.12.7 ISAM Conversion Utility--DISCONV

The DISCONV utility is used off-line to convert an existing indexed sequential file to a DISAM data base. The DD statement with ddname ISFILE must specify the existing ISAM data set. The ddnames DISINDX and DISDATA refer to the DISAM index file and the DISAM data file, respectively. All desired DCB attributes must be specified on the DD statements. The logical record length of the index file (DISINDX) must be large enough to contain the record pointer and the key, and the two fields must not overlap.

If fixed length BDAM data files are used and the block size of the data file (DISDATA) is equal to or greater than the logical record length of the ISAM file, the record key will be included in the data, and the record padded with binary zeroes. Alternatively, the block size may be equal to the logical record length of the ISAM file minus the key length; the key will not be included.

The PARM field of the EXEC statement may contain the offset to the record pointer field within the index record, and the number of dummy blocks to be written at the end of the data file, to contain additions. Both these parameters are in keyword form, for example:

```
//STEP1  EXEC    PGM=DISCONV,PARM='OFFSET=2,BLOCKS=25'
```

The default for the offset value is 1 (that is, the second byte of the record). The default for the number of dummy blocks is zero, allowing no additions.

6.12.8 Index File Reorganization Utility--DISREORG

The DISREORG program can be used off-line to reorganize the index file of a DISAM data base. The DD statement with ddname DISINDX must specify the DISAM index file. The DD statement with ddname DISTEMP must specify a temporary work file, on tape or disk, large enough to contain the index records from the DISAM index file. The index file will be reorganized and rewritten in the same area it previously occupied.

6.13 INTERCOMM CFMS SUPPORT

Use of the IBM Chained File Management System is available to Intercomm users via an interface routine. CFMS may be utilized in one or more batch regions where Intercomm is operational. All the functions of CFMS are available to the Intercomm user.

CFMS does not change the standard Intercomm environment. Furthermore, the Intercomm implementation of CFMS requires no modifications to the standard CFMS calling sequence. Standard CALL statements as specified in the CFMS Program Description Manual, IBM reference SH20-0777, are used for all activity against the CFMS files.

CFMS data sets are accessed via the File Handler and as such are eligible for all File Handler options. Of special import to users performing file updates on-line is the File Recovery feature, an extended capability to the Intercomm system.

Since CFMS is not reentrant and not serially reusable, each application subsystem must have its own copy of the CFMS "IOPROCESS", "FILEORG", "MAINLINE", "CB$BM", "CB$CH", "PL$BM" and "PL$CH" modules, as applicable, just as is normally required when executing CFMS directly in a batch environment.

This requirement restricts the usage of CFMS to a single subsystem, since multiple copies of the same module in a single linkedit will yield duplicate entry points, unless the subsystem is either dynamically loadable and dynamically linkedited (see Chapter 3), or the entry point names are changed at linkedit time. Furthermore, because CFMS is nonreentrant, each application subsystem must be single-threaded to prevent access to the subsystem by more than one user concurrently. Single-threading is easily accomplished by specifying a value of one on the MNCL parameter in the SYCTTBL macro describing the subsystem. The interface routine CFMSINTF, which enables CFMS to run under Intercomm, is reentrant; therefore only one copy need be present in the Intercomm system, no matter how many application subsystems are utilizing CFMS.

When the CFMS "IOPROCESS", "CB$BM" and "PL$BM" modules are assembled, certain standard OS macros, performing the OPEN, CLOSE, CHECK and WAIT access method service functions, must be replaced by Intercomm-supplied macros. The replacement of these four macros enables the Intercomm CFMS Interface, CFMSINTF, to intercept all I/O requests made by CFMS, thus allowing the interface to channel these requests through the standard Intercomm File Handler instead of having CFMS funnel them directly to OS.

These macros are supplied with Intercomm as two source members, the first, CFMSMAC1, is to be used only with the "IOPROCESS" module and the second member, CFMSMAC2 is to be used when assembling either the "CB$BM" or the "PL$BM" CFMS language interface modules. The macros are included at assembly time via a concatenation to the Assembler's SYSIN input data set. Since the assembler requires that in-line macros appear before any source code, the member CFMSMAC1 or CFMSMAC2 must be concatenated as the first data set to the assembler's SYSIN file.

```
//ASM.SYSIN DD  DSN=INT.SYMREL(CFMSMACx),DISP=SHR
//          DD  DSN=CFMS.SOURCE({IOPROCESS}),DISP=SHR
                              {CB$BM    }
                              {PL$BM    }
```

The Intercomm implementation of CFMS does not change the standard CFMS calling sequence. However, it is possible that an additional return code, a hexadecimal FFFF, may be passed back to the user, by the interface, to indicate an abnormal condition. To maintain a consistency with CFMS standards for posting error conditions, this return code is placed in bytes 11 and 12 of the work area the user supplies to CFMS.

The following conditions will cause the generation of this return code:

- An OPEN request for one or more of the named CFMS files did not complete successfully. This condition normally occurs as the result of a missing DD statement or the omission of certain required parameters on the DD statement.
- An unrecoverable I/O error has occurred while accessing a CFMS file.
- An invalid request has been made by CFMS to the interface. This condition occurs only if the contents of the I/O control blocks within CFMS were destroyed.

Since all CFMS I/O activity is routed through the File Handler, the user may elect to utilize one or more of the File Handler options to further control I/O activity on the CFMS files. One such option is the explicit designation via a File Attribute Record (FAR), of a file as being READONLY. By declaring a file as READONLY, the File Handler will reject any requests to write to that file and will ignore any user requests, through the File Handler, to obtain exclusive control of that file.

To implement CFMS, include CFMSINTF in the Intercomm linkedit in addition to the IBM CFMS interface modules.

Chapter 7

EXECUTION OF INTERCOMM

7.1 INTRODUCTION

Execution of Intercomm entails a linkedit of all resident user-coded and Intercomm-supplied routines and tables, and resident or overlay subsystems, to produce an executable load module, followed by execution in Test Mode, or in live mode with actual or simulated terminals. The mode of execution is controlled by the EXEC statement PARM data and system logic determining whether or not specific system routines were included in the load module.

The Intercomm System Manager(s) may provide as many as four different linkedit versions of Intercomm for use at an installation:

1. A production system for actual day-to-day operation
2. A terminal testing system, including user subsystems being tested via operator entry at terminals, and/or simulated terminal input
3. A Test Mode system, including production subsystems for volume testing
4. A minimal Test Mode system, including only system programs and service routines required for testing one subsystem.

This chapter documents the following topics:

- Generating a linkedit deck
- The Intercomm linkedit
- Execution JCL
- System startup
- System closedown
- Live operation
- Intercomm quiesce facility
- OS/VS operation
- MVS operation
- Interregion SVC installation
- Link Pack Feature

7.2 GENERATING A LINKEDIT DECK

The required linkage editor control statements to produce an Intercomm load module for execution may be generated initially via the ICOMLINK macro (described in Basic System Macros). Based upon global settings in the SETENV and SETGLOBE members and user-specified parameters or default values for ICOMLINK, assembly of ICOMLINK produces (punches) INCLUDE statements for the required Intercomm routines, and OVERLAY and INSERT statements for their overlay structure (if desired). The required entry point to the Intercomm load module is PMISTUP. Recommended JCL to produce the linkedit deck is as follows:

```
//  EXEC    ASMPC,Q=LIB
//ASM.SYSGO DD  SYSOUT=B
//ASM.SYSIN DD  *
* GENERATE  LINK EDIT DECK
          ICOMLINK   user-defined-parameters....
          END
/*
```

NOTE: the output from SYSGO can be member of a PDS such as SYMUSR, a TSO data set, etc., as desired.

INCLUDE statements must then be added for application subsystems and subroutines (except those dynamically loaded). In addition, appropriate OVERLAY and INSERT statements for some of these modules may be defined if an overlay structure is used. Overlay areas for application subsystems and subroutines are described in Chapter 3 of this manual. Also, if executing under VS1 or MVS, ORDER statements may be placed at the beginning or end of the linkedit.

7.3 THE INTERCOMM LINKEDIT

The actual linkedit may be accomplished via the Intercomm procedure LKEDP. The SYSLIB definition for this procedure references only the Intercomm libraries; the user must provide additional DD statements to reference system libraries, such as SYS1.TELCMLIB (teleprocessing access method modules), SYS1.COBLIB (COBOL modules), and user libraries for application subsystems, etc., as appropriate. An example of the use of the LKEDP procedure is shown in Figure 7-1. The SYSIN data set can be a PDS member (from SYMUSR) or TSO (CMS) data set.

The linkedit error messages should be examined for unresolved references. Many optional features are implemented by Intercomm conditional calls; in this case unresolved references present no problem. The Intercomm S.E.O.D. may be consulted to verify the critical nature of unresolved references.

```
//LINK        EXEC    LKEDP,Q=xxx,LMOD=ICOMEXEC
//*           THE FOLLOWING SUBSTITUTION JCL ADDS THE COBOL
//*           LIBRARY TO THE CONCATENATION SEQUENCE OF INT.MOD&Q,
//*           INT.MODUSR,INT.MODLIB,INT.MODREL:
//LKED.SYSLIB DD
//           DD
//           DD
//           DD
//           DD      DSN=SYS1.COBLIB,DISP=SHR
//LKED.SYSIN  DD      *
            INCLUDE
               .                    LINKEDIT DECK PRODUCED
               .                    BY ICOMLINK MACRO
            INCLUDE
               .                    PLUS REQUIRED INCLUDES
               .                    FOR USER MODULES AND TABLES
/*
```

Figure 7-1. Using LKEDP Procedure to Generate Intercomm Load Module

7.3.1 Linkage Editor External Symbol Table Overflow

If the following error message:

IEW0254 ERROR - TABLE OVERFLOW--TOO MANY EXTERNAL SYMBOLS IN ESD

occurs during linkage editor execution, override the linkage editor SIZE parameter in the following manner:

```
//LINK EXEC  LKEDP,Q=xxx,LMOD=ICOMEXEC,
//     PARM.LKED='SIZE=(512K,100K),XREF,LIST,LET,NCAL'
```

Refer to IBM linkage editor documentation for appropriate SIZE values to use. Add OVLY to the parms if an overlay structure is desired. Do not code either REUS or RENT.

7.3.2 Linkage Editor Parameters

There are two linkage editor parameters which influence the number of overlay FETCH operations and, in turn, the response time for a loaded program.

If the Downward Compatible (DC) option is specified when linkediting, the maximum block size created on the load library will be 1024 bytes. This means that for a 10K overlay program to be loaded, at least 10 FETCH operations will be executed. This will considerably slow the response time of the program because of the extra I/O involved. The solution is to ensure that there is no DC parameter for the linkedit step (see SIZE override example, above).

In the SIZE parameter, the maximum record size of a disk unit will be equal to one half of the value of the second parameter (yyy) of SIZE=(xxx,yyy). If the text record size is too small, there will be additional FETCH operations, again slowing response time. Therefore, specify twice the maximum text record size (for 3330s, yyy should equal 26K; for 3350s, yyy should equal 40K; for 3380s, yyy should equal 80K).

7.4 EXECUTION JCL

The execute (EXEC) statement is the first statement of each job step and contains the load module name and data that pertains to the job step. The principal function of the Intercomm execute statement is to identify the load module to be executed and define Intercomm's mode of execution. The execute statement is coded as follows:

```
//stepname       EXEC     PGM=load-module-name,
//  PARM='mode-of-execution[,WTO
//                ,ddname=R,ddname=R,.....
//                ,ddname=A,ddname=A,.....
//                ,ddname=B,ddname=B,.....
//                ,VOL=number]'
```

load-module-name
indicates the name of the Intercomm load module to be executed.

mode-of-execution
describes the function to be initiated. Acronyms that define the mode of execution and their functions are:

Acronym	Execution Mode Options
STARTUP	Normal startup with terminals
RESTART	Restart mode of startup, including processing of log for message restart
RESTRNL	Restart without log; will call RESTORE if checkpoint is used
TEST	Execute Test Mode of Intercomm
TESTR	Test Mode with full restore and restart capability
TESTRNL	Same as RESTRNL, except in Test Mode

NOTE: To preserve semipermanent DDQs (especially if spooled printer output created via MMU) and/or semipermanent Store/Fetch strings, Intercomm must be brought up in restart mode; if message restart is not used, code RESTRNL.

WTO
if specified, indicates that optional messages (MS003I and MS008I) are to be printed on the CPU console. See Messages and Codes. Refer also to the SPALIST and SYCTTBL WTO parameters.

ddname=R
specifies the ddnames of those data sets that are to be "readonly." Each ddname is coded followed by the equal sign and an R. FAR statements may be used instead of coding this parameter (see Chapter 6).

ddname=A
specifies the ddnames of those data sets that are updated using AMIGOS as an access method. Each ddname is coded followed by the equal sign and an A.

ddname=B
specifies the ddnames of those data sets that are AMIGOS "readonly" files. Each ddname is coded followed by the equal sign and a B.

VOL=number
specifies, for TOTAL users, the value representing the QUIET interval to be passed to TOTAL (in a separate region) when Intercomm closes down. (TOTAL support is described in Data Base Management System Users Guide.)

7.4.1 Global WTO and MCS Routing

Users can force and/or suppress routing of system messages issued in Intercomm via the PMIWTO and PMIWTOR macros. The SPALIST macro has four parameters for this purpose:

- FMCSWTO
- SMCSWTO
- FPMIWTO
- SPMIWTO

The SPALIST parameters specify, for both MCS (CPU console) and Intercomm routing, the options to be suppressed, and those to be forced. This facility could be used, for example, to prevent any system messages (except WTORs) from being sent to the CPU console, or to force all messages to SYSPRINT. See Basic System Macros for coding specifications. See also the WTOPFX parameter of the SPALIST macro for message prefix-ID override feature, and Messages and Codes for message syntax.

7.4.2 STEPLIB or JOBLIB Requirements

Execution JCL must reference the following libraries as STEPLIB or JOBLIB data sets:

- The library containing dynamically loaded subsystems and subroutines.
- The library containing compiler-oriented dynamically loaded service routines, such as SYS1.COBLIB. Frequently used routines should be made resident whenever possible.
- The library containing the Intercomm load module.
- The library containing user versions of Intercomm tables which may be loaded at startup.
- MODREL--required if dynamic linkedit is used, and the ICOMCESD and ICOMVCON modules are not contained on one of the above-mentioned libraries.

Concatenation sequence is critical to performance. The order of the DD statements is installation-dependent, based upon frequency of access. MODREL is infrequently referenced, and should be among the last in the series. If an overlay structure is used, the library containing the overlay loaded routines should be first in the concatenation stream.

7.4.3 DD Statement Requirements

The execution JCL contains Data Definition (DD) statements describing all data sets accessed by Intercomm. The following DD statement names are required:

- INTERLOG

  The system log data set (tape or disk; see Chapter 9).

- SMLOG

  Resource Management statistics reports and thread dumps

- STSLOG

  System Tuning Statistics reports

- SYSPRINT

  For IJKTRACE output, statistics reports, system messages, etc.

- SNAPDD

  For snap output

- SYSUDUMP

  For abend output if SNAPDD unusable

- RCTOOO

  Output Format Table disk-resident Entries

Additionally, for subsystem and terminal queues:

- PMIQUE

  Intercomm subsystem disk queues (Output, Change/Display, etc.) as defined in the SYCTTBLs at system installation time

- BTAMQ

  The Intercomm BTAM/TCAM terminal disk queues defined via SYCTTBLs in the BTAMSCTS module at Front End installation time

- VTAMQ

  VTAM terminal disk queues defined via LUNIT/LCOMP macros for a VTAM Front End

- ddnames

  Additional installation-dependent disk queue data sets

Additionally, for Test Mode execution:

- SYSSNAP

  Test Mode input messages (snaps with ID=15)

- SYSSNAP2

  Test Mode output messages (snaps with ID=20)

Additional Intercomm data sets that may be required are described in this manual with each particular feature; that is, FAR Parameters Input File, File Handler Statistics File, Checkpoint File, Terminal Simulator Input, Dynamic Linkedit, etc., and in the special feature manuals.

NOTE: All Intercomm and user data sets accessed by the File Handler must include the DCB parameter DSORG (or AMP, if VSAM) on the DD statement and, except for sequential output data sets, must be DISP=OLD or SHR. The Intercomm utility CREATEGF may be used to format BDAM data sets such as disk queues. (See Chapter 12, "Off-Line Utilities".)

7.5 SYSTEM STARTUP

System startup is accomplished by the module STARTUP3, consisting of a resident Csect given control by OS, and an overlay Csect performing the main system initialization functions. The resident module, PMISTUP, accomplishes the OS linkage conventions, calls POOLSTRT (if in link) to load Intercomm pools (see Chapter 5), and issues STAE/ESTAE macros if the module STAEEXIT is included in the Intercomm linkedit (see Chapter 8).

The overlay module, STUOVLY, performs analysis of the "mode of execution" parameter and, based on the presence of system modules, performs initialization functions in the following order:

- Set PMIWTO/R messages global Job/Region Identifier
- LPSTART - resolve VCONs from Link Pack Module
- SSINIT - START/STOP command function initialization
- FASTSNAP initialization
  -- if FASTSVC specified and FASTSNAP DCB present
  -- if DCB opened successfully-set on SEXFSNAP
- Configuration initialization

CPU Model	SEXMODEL
Operating System	SEXBITS1
Release Number	SEXVERSN
Region Boundaries	SEXPPBEG, SEXPPEND
Link Pack Area	SEXLPBEG, SEXLPEND

- Set SPINOFF snap spooling for MVS if FREE=CLOSE specified
- Open SNAPDD data set (PMISNAP DCB)
- STOSTART - start Store/Fetch initialization

- Attach ICOMDYNL - Dynamic Linkedit initialization
- VSINIT - VS initialization - fix the Fix Table (if any)
- IXFMON00 - File Handler initialization
  -- IXFFAR - FAR specifications processing
- IJKCESD - initialize Csect/Entry table for debugging reports, etc.
- TASKSTRT - Generalized subtasking initialization
- Initialize log buffers (unless Satellite Region with single region logging)
  -- acquire storage for number/size buffers defined in SPA
  -- SELECT per log buffer
- PMIDATER -- set date in SPA
- ILBOSTP0 -- ANS COBOL initialization
- ASYNCH -- Attach overlay load subtask if present
- ASYNCLDR -- Attach dynamic load subtask if present
- Determine mode

	SPAMODE
Startup	(0)
Restart	(4)
Test	(8)

- Determine restart (Live or Test) options
- MRSTART - initialize Multiregion if applicable
- CKOVLYNO - Overlay A numbers between 4 and 62 and in ascending sequence, if overlay used
- CKLINK - check overlay linkedit structure corresponds to SCT
- USERINIT - user exit for Basic TCAM MCP active test
- SCT initialization - all SYCTTBLs
  Open Basic TCAM process queue DCB, if applicable
  SELECT disk queue; if queue cannot be selected (opened), flag SCT entry to only use core queue
  Initialize auxiliary SCT - point to primary
- CALCRBN - allocate RBNs for Front End and SCT disk queues
- Open Basic TCAM destination queue DCB (QTAMOUT), if applicable
- If SAM modules in link, check SAMTABLE included

- PMIPRIME - If Test Mode, prime input data buffers
- RESTORE - if no restart log; initialize checkpoint file, if checkpointing desired
- DDQSTART - initialize Dynamic Data Queuing facility
- LOGPROC - process restart log: checkpoint, restart, file recovery, data base recovery, serial restart (if used)
- INTERLOG initialization (unless Satellite Region with single region logging)
  -- set log buffer count to NCP (if NCP lower)
  -- free OS buffers
- TRIGGER - Time Zone Table processing
- dispatch CHECKPT - checkpoint processing
- dispatch TRAFFIC - overlay subsystems message traffic control
- DBSTART - data base initialization
- dispatch LOGINPUT - extra log input - threshold testing
- dispatch IXFRPT01 - File Handler Statistics reporting
- RJESTART - OS RJE facility interface initialization
- TCAMSTRT - Basic TCAM user initialization
- BTAMSTRT - BTAM/GFE/Extented TCAM initialization
  -- Front End Table Verification (BTVERIFY, TCAMVER)
- VTSTART - VTAM initialization
- PMICKFTB - adjust block size, try selecting each file in PMIFILET
- PMIFIXA - page fix all specified areas
- BLDL for all dynamically loadable S/S - move resident BLDL to SCT extension - validate load module size against SEXSPMAX (SPALIST--MAXLOAD parameter)
- dispatch DELOAD - dynamic subsystem loading processor
- PMIDEBUG - debugging WTOR (see Messages and Codes)
- dispatch PMIHARDW - allow Intercomm quiesce (adjust timers)
- post Multiregion active, if used

- wait for Dynamic Linkedit to end, detach subtask
- check dynamic subroutines
  -- if in link, flag as permanently resident
  -- if to be permanently resident, load and flag as resident.
- STOSTART - wait for Store/Fetch initialization to complete
- MMUSTART - initialize Message Mapping Utilities
- USRSTART - user exit - issue startup broadcast messages or ESS sign-on messages (call USRSTRT1 from USRSTART)
- INTSTS - start System Tuning Statistics reporting
- GENESIS - dispatch Subsystem Controller for all resident and dynamically loadable SCTs and for all SCTs in first overlay group (if used)
- dispatch TRANGEN - Model System Generator activity
- issue startup complete message with latest SM level, SPA address
- issue SPIE using SEXSPICA ((1,13),15) if SPIEEXIT in linkedit (see Chapter 8)
- dispatch LOWCORE - (core flush routine when cushion released condition) - wait on SEXLOCOR ECB
- indicate startup complete - post SEXSTUPE for VTAM - set SEXSTRUP in SEXSWTCH
- IJKTLOOP - closed loop detection routine initialization
- dispatch exit to the Dispatcher.

At completion of system startup, both the Back End and Front End (if required) have created tasks on the Dispatcher queues to perpetuate their operation. Control is then transferred to the Dispatcher to continue execution and manage the Intercomm multithreading environment.

7.5.1 Startup User Exits--USRSTART/USRSTRT1

A conditional call (CALLIF) is made to the user exit USRSTART prior to completion of system startup and after initialization of the Front End, File Handler, etc. If included in the load module, this routine is given control with register 1 containing the address of the execution parameter list. A member USRSTART is included with the Intercomm release.

The USRSTART routine, as released, formats and sends a message to the broadcast group name TOALL at startup time. The message states:

*** GOOD MORNING *** INTERCOMM IS READY: MM-DD-YY HH.MM

The Output Format Table entry is RPT00045. MORNING will be replaced by AFTERNOON or EVENING at the proper time of day. An entry must be made in the Broadcast Table PMIBROAD (BROADCST Csect) for the group name TOALL. This is provided in the released version of this table. Add to it the names of all terminals to receive the startup message.

If the Extended Security System (ESS) is in use, an internal USRSTART routine generates sign-on prompt messages instead.

Additionally, USRSTART calls a routine named USRSTRT1 if it is coded and included in the linkedit (also called by ESS). USRSTRT1 must be coded in reentrant Assembler Language and use standard linkage conventions. At entry to USRSTRT1, register 1 points to the address of the OS formatted PARM values coded on the EXEC JCL statement for Intercomm execution. No return code is expected. This user exit may perform additional installation-dependent startup processing, if necessary.

USRSTART is written in reentrant Assembler Language. The member name, Csect name and load module name are all USRSTART. See also the USRSTRT parameter of the ICOMLINK macro to provide/prevent automatic linkedit inclusion.

7.6 SYSTEM CLOSEDOWN

The closedown functions are performed upon receipt of the NRCD or IMCD transactions (see System Control Commands) or, in the case of Test Mode, when all subsystem queues are empty. Closedown in a Multiregion Intercomm system is described in Multiregion Support Facility.

Closedown in live mode (or with simulated terminals) consists of routing a message to the closedown subsystem, PMICLDWN, an entry in CLOSDWN3. This subsystem will continue to scan the SCTs and requeue a message for itself until all messages are processed (NRCD) or messages in progress are complete (IMCD). A final checkpoint is taken and control is passed to the Front End to ensure that all messages queued for transmission to operational terminals are sent before closing the line DCBs and/or VTAM ACB. The Intercomm log buffers are flushed and the log is closed before issuing final System Tuning and File Handler Statistics. The File Handler termination routine (IXFMON09) is then called to close the files prior to job termination.

Closedown in Test Mode completes the Back End termination functions described above because an NRCD command is internally generated, or the job terminates with an Abend 999, indicating all input messages have been processed. These options are controlled by the TSTEND parameter on the SPALIST macro.

7.6.1 Closedown Time Limit

Under certain circumstances, a normal closedown (NRCD) may be initiated, and then, due to subsystem or terminal conditions, it is discovered that closedown will take excessive time to complete. There are two methods by which this eventuality may be handled:

- A user-specified maximum time limit on the SPALIST (CLDNLIM) will be set at the beginning of closedown processing; at the expiration of this time interval, closedown will not wait for any further terminal or message processing, but will terminate all Intercomm system functions and return to the operating system. See also the SPALIST macro CLDTO parameter.

- An IMCD may be entered during closedown processing. This will have the same effect as the expiration of the time interval described above.

When forcing a premature closedown by these techniques, an Intercomm restart may be needed to recover messages queued or in process at the time of closedown.

7.6.2 Closedown User Exits--USRCLOSE/USRCLSE1

A conditional call (CALLIF) is made to a user exit (USRCLOSE) prior to completion of closedown cleanup processing (final checkpoint, statistics reports, etc.). A member USRCLOSE is supplied with the Intercomm release.

The released member will send a message to the broadcast group name TOALL during normal closedown processing. The message states:

*** GOOD MORNING *** INTERCOMM IS CLOSED: MM-DD-YY HH.MM.

RPT00045 is used and the MORNING is replaced by AFTERNOON or EVENING at the proper time of day. An entry must be made in the Broadcast Table (BROADCST) for the group name TOALL. (See Section 7.5.2, "Startup User Exits," above.)

USRCLOSE also calls the user exit routine USRCLSE1, if it is included in the linkedit. USRCLSE1 must be coded in reentrant Assembler Language and use standard linkage conventions. At entry, register 1 points to a parameter list containing the address of the entered closedown message and the address of the System Parameter Area. No return code testing is done.

USRCLOSE is written in reentrant Assembler Language. The member name, Csect name and load module name are all USRCLOSE, which is automatically included in the Intercomm linkedit.

7.7 LIVE OPERATION

Execution of Intercomm in live mode necessitates that terminal operation (the Front End) is activated for actual or simulated (see Chapter 8) terminals. When startup functions are complete, terminal input/output processing begins.

The system may be activated as a cold start with no consideration for any previous execution (EXEC statement parameter STARTUP), or as a warm start with message restart/recovery performed (EXEC statement parameter RESTART). Restart/Recovery functions are described in detail in Chapter 9.

7.7.1 HASP Modification to Run Intercomm Under HASP

HASP does not permit the use of the dispatching priority parameter (DPRTY) on the EXEC JCL statement. Since Intercomm requires the use of this parameter to establish its dispatching level above all other jobs, a modification to HASP is required to provide this capability. The following one-statement change to the source code will permit the use of the DPRTY parameter: in module XEQ, replace the instruction at label XJCLXQPR with an unconditional branch to label XJCLEXIT.

7.7.2 Intercomm and ASP

In order for Intercomm to execute successfully, it requires a higher dispatching priority than other jobs operating in the system. When running under ASP, it is necessary to set up a job class which is permitted to use the OS dispatching priority parameter on the EXEC statement for Intercomm. This can be accomplished by modifying the ASP initialization deck, to specify JPRTY=JOB on the CLASS statement which defines the class under which the Intercomm job will execute.

7.7.3 Execution JCL

Execution JCL requires specifications for the network configuration. A typical live execution job with a BTAM Front End is shown in Figure 7-2.

```
//ICOMEXEC   EXEC   PGM=ICOMLIVE,PARM='STARTUP',REGION=1024K
//STEPLIB    DD     DSN=INT.MODUSR,DISP=SHR
//            DD     DSN=INT.MODLIB,DISP=SHR
//            DD     DSN=INT.MODREL,DISP=SHR
//INTERLOG   DD     DSN=&&INTLOG,VOL=vvvvvv,UNIT=unit,
//                  DISP=(,PASS),LABEL=(,SUL),
//           DCB=(DSORG=PS,RECFM=VB,BLKSIZE=4100,LRECL=4096,NCP=8,OPTCD=C)
//*
//*          NOTE THAT INTERLOG BLOCK SIZE MUST BE AS LARGE
//*          AS THE LONGEST EXPECTED LOGGED MESSAGE (+4).
//*
//SMLOG      DD     SYSOUT=A,
//           DCB=(DSORG=PS,LRECL=120,BLKSIZE=multiple-of-120,RECFM=FBA)
//STSLOG     DD     SYSOUT=A,
//           DCB=(DSORG=PS,LRECL=120,BLKSIZE=multiple-of-120,RECFM=FBA)
//SYSPRINT   DD     SYSOUT=A,
//           DCB=(DSORG=PS,RECFM=VBA,BLKSIZE=multiple-of-137-+4,LRECL=137)
//RCT000     DD     DSN=INT.RCT000,DISP=OLD,DCB=(DSORG=DA,OPTCD=RF)
//PMIQUE     DD     DSN=INT.PMIQUE,DISP=OLD,DCB=(DSORG=DA,OPTCD=RF)
//BTAMQ      DD     DSN=INT.BTAMQ,DISP=OLD,DCB=(DSORG=DA,DPTCD=RF)
//USERFILE   DD     DSN=......
           .
           .      USER DATA SET DEFINITIONS
           .
//*          THE FOLLOWING ARE NOT PROCESSED BY THE FILE HANDLER
//PMISTOP    DD     DUMMY
//SYSUDUMP   DD     SYSOUT=A or DUMMY
//SNAPDD     DD     SYSOUT=A           STANDARD SNAPS
//*
//*                 FOLLOWING IS FOR DYNAMIC LINKEDIT
//*
//DYNLLIB    DD     DSN=INT.MODUSR,DISP=SHR
//DYNLWORK   DD     UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(1,1))
//DYNLPRNT   DD     SYSOUT=A
//LINE1      DD     UNIT=nnn
           .
           .      TERMINAL NETWORK DEFINITIONS IF BTAM USED
           .
//*
//PRINTLOG   EXEC   PGM=LOGPRINT,COND=EVEN
//STEPLIB    DD     DSN=INT.MODREL,DISP=SHR
//INTERLOG   DD     DSN=&&INTLOG,DISP=OLD,DCB=BLKSIZE=5000
//SYSPRINT   DD     SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=121)
```

Figure 7-2. Typical Live Execution JCL

NOTE: if executing with VSAM data sets, place STEPCAT DD statement(s) for user catalog(s) after the //PMISTOP DD statement (do not use JOBCAT) so that the File Handler does not process the catalog at startup.

For remote terminals accessed via BTAM, the sequence of the DD statements describing operational lines must correspond to the network configuration definition in the Front End Network Table. The LINEGRP macro defines the ddname in the execution JCL. The order of the DD statements for each line group defines the physical unit addresses relative to the associated sequence of BLINE macros, as depicted in Figure 7-3. Remote lines must be on the byte-multiplexor channel (Channel 0).

For local BTAM (3270) and graphics terminals, the sequence of the DD statements for each line is related to the sequence of the BTERM macros as illustrated in Figure 7-4. Local terminals are defined on a block-multiplexor channel (not Channel 0).

7.7.4 Low-Core Condition--SSPOLL

When a low-core condition exists, the user may optionally prevent additional leased-line terminal and/or TCAM input to the Intercomm system. Polling is automatically temporarily halted and later resumed when sufficient storage becomes available. Include SSPOLL in the resident portion of the Intercomm linkedit to activate this feature. The system control commands SPPL and STPL, may be used at other times to temporarily halt input.

```
* A BTAM NETWORK TABLE CODED AS FOLLOWS REQUIRES JCL DDCARDS
* AS ILLUSTRATED BELOW
R01       LINEGRP   DDNAME=RR1,------
            :
          BLINE     LGNAME=R01,------
          BTERM     TERM=RR101,------
          BTERM     TERM=RR102,------
            :
          BLINE     LGNAME=R01,------
          BTERM     TERM=RR201,------
            :
          BLINE     LGNAME=R01,------
          BTERM     TERM=RR301,------
            :
```

```
//* DDCARDS FOR LINEGRP R01
//RR1     DD        UNIT=031  (TERMINALS RR101,RR102,ETC.)
//        DD        UNIT=032  (TERMINALS RR201,ETC.)
//        DD        UNIT=033  (TERMINALS RR301,ETC.)
```

Figure 7-3. LINEGRP, BLINE Sequence and JCL for Remote Terminals

```
* A NETWORK TABLE CODED AS FOLLOWS FOR GRAPHICS OR LOCAL 3270
* TERMINALS REQUIRES JCL DDCARDS AS ILLUSTRATED BELOW
L01       LINEGRP   DDNAME=LL1,------
            :
          BLINE     LGNAME=L01,------
          BTERM     TERM=LL001,------
          BTERM     TERM=LL002,------
          BTERM     TERM=LL003,------
            :
```

```
//*  DDCARDS FOR LOCAL 3270 BLINE
//LL1     DD        UNIT=301  (TERMINAL LL001)
          DD        UNIT=302  (TERMINAL LL002)
          DD        UNIT=303  (TERMINAL LL003)
            :
```

Figure 7-4. BLINE, BTERM Sequence and JCL for Local Terminals

7.8 INTERCOMM QUIESCE

It is sometimes necessary to stop the CPU while running Intercomm so that maintenance or volume switching can be done. However, if the CPU is stopped for a significant period of time (more than one minute), it is likely that, when processing is resumed, Intercomm's event-timing will have been disrupted, resulting in various time-outs. This causes erroneous cancellation of messages, snaps 114 and 118, etc.

The Quiesce facility allows the CPU console operator to stop all Intercomm processing by replying to an outstanding WTOR prior to stopping the CPU itself. When processing is to be resumed, Intercomm can be reactivated by replying to a subsequent outstanding WTOR. All of Intercomm's internal timings are adjusted to reflect the lost time, thereby avoiding time-outs.

The Quiesce facility is optional, and is provided by including the module PMIHARDW in the Intercomm linkedit. This module, which is dispatched at startup time, puts out a WTOR (MU001R) with the following text:

REPLY "ICOMHALT" WHEN YOU WANT INTERCOMM TO TEMPORARILY STOP PROCESSING

This WTOR will remain outstanding until needed. When the proper reply is given, Intercomm will go into the wait state, after putting out another WTOR (MU002R):

REPLY "ICOMSTART" WHEN YOU WANT INTERCOMM TO RESUME PROCESSING

At this point, it is safe to stop the CPU. When the CPU is again started, Intercomm can be reactivated by replying to this latter WTOR. The first WTOR will then be put out again allowing the procedure to be repeated if and when necessary.

The time interval during which Intercomm is quiesced is lost to the system. If a one-hour time-dispatch was done by some internal routine at 12:00 P.M., this interval would normally expire at 1:00 P.M. If, however, Intercomm was quiesced from 12:20 to 12:25, the interval will expire at 1:05.

7.9 OS/VS OPERATION

All VS support is applicable to both VS1 and MVS. Installation and operation of Intercomm with VS follows the previously described procedures, plus additional considerations documented in this section to take advantage of VS facilities. More considerations applicable primarily to MVS are described in Section 7.10. The VS vocabulary is illustrated below:

Reference	Meaning
EPS	External Page Storage; Page Data Set
Page Fault	A page is referenced that is not residing in real storage, but on EPS.
Page	Segment of main storage, 4K for MVS, 2K for VS1
Page Loading	Transfer of page from EPS to real storage
Page Fixing	Marking a page as nonpageable; that is, remains in real storage full-time

The major difference between OS/MFT or OS/MVT and operation of Intercomm under VS is the unpredictable nature of program loading in the partition/region. Under OS, when a job is loaded, the user knows the job is actually residing in main storage. This is not true for VS, where there are two types of storage: Real Storage and External Page Storage (EPS), also referred to as the Page Data Set. Only a certain portion of a load module actually resides in real storage; most of it (depending on real core availability, number of jobs concurrently running, etc.) will reside on EPS.

When a program references a page that currently resides on EPS, an I/O operation must be performed in order to transfer that page from EPS into real storage. This procedure is called page loading. Each time a page that is residing on EPS is referenced (a page fault), the task's TCB is marked nondispatchable by VS until the referenced page is loaded. This can result in extensive degradation of response time in an on-line system, since the task (Intercomm) must wait until I/O completes.

With Intercomm, there are two alternatives to avoid page faults: page preloading and page fixing. These two facilities can be implemented independently or in conjunction with each other. Under MVS, both facilities require installation of the Intercomm Interregion SVC (see Section 7.11), while under VS1, only page fixing requires the SVC.

7.9.1 Page Preloading

Using the page preloading feature, the same process is executed as when a page fault occurs; that is, page loading from EPS must be requested. However, if the page loading is requested by the user (Intercomm) before the page fault actually occurs, the task's TCB will not be marked nondispatchable. Thus, the task will not be in a wait state until the I/O completes and therefore other processing can continue while the I/O is still in progress. Page preloading under Intercomm is done for pages that are likely to be on EPS at the time they are referenced again; that is, save areas and return points to application programs are preloaded prior to transfer of control from the Dispatcher.

7.9.2 Page Fixing

To fix a page means to allocate a real storage page frame to a virtual address (on a page boundary), and reserve that piece of real core for the page until an UNFIX command (VS macro) is issued (or until re-IPL time). Care should be taken in implementing the page fixing facility for two reasons:

1. There is no VS end-of-job cleanup for releasing fixed pages. Provisions to unfix pages under Intercomm are made in closedown and STAEEXIT (which means, under normal closedown and abend conditions, the pages previously fixed will be unfixed).

2. Real storage allocated to a fixed page is taken away from the system, rather than from the application's partition/region. Therefore, if too many pages are fixed, serious degradation of the system and batch jobs can be encountered.

The page fixing feature is invoked at startup via a call to a nonreentrant module, PMIFIXA. All entries contained in the FIXTABLE (a user-coded table identifying groups of pages) will be fixed. If an unrecoverable error occurs, all pages (including those just fixed) will be unfixed and the user is notified of such action. After startup completes, the pages fixed by PMIFIXA can then be unfixed (and fixed a subsequent time) via the PGFX system control command. This transaction is processed by the single-thread subsystem PMIFIXB. (Installation requirements for the PGFX command are described in System Control Commands.)

7.9.3 VS Installation Procedures

To install VS processing in conjunction with standard Intercomm installation, follow the steps below:

1. Update the member SETGLOBE to specify global settings for the VS system in use (change from SETB 0 to SETB 1). These globals are listed in Chapter 2.

2. All VS page management macros, that is, PGFIX, UNFIX, PGLOAD, PGFREE, IHBPSINR, etc., must be available on SYS1.MACLIB. These macros are distributed on AMODGEN (initial distribution library of your VS system).

3. Provision must be made for installation of the Intercomm Interregion SVC, as described in Section 7.11.

4. Reassembly to incorporate VS macros must be performed for:

   - VSINIT for VS initialization at startup (required)
   - PMIPGLD and LOADPAGE for the Page Preloading facility
   - PMIFIXA and PMIFIXB for the Page Fixing facility

5. For page fixing only, FIXTABLE must be coded. FIXTABLE is generated using the Intercomm macro ICOMFIX. Each entry of the table represents a group of pages specified by the name of the first module in the first page, the number of pages (maximum 32), and a three-character identifier of the group.

   The Csect name of the table is arbitrary. The table must be linkedited as resident, along with all the pages specified in the table (see Section 7.9.4, "Page-Fixing Guidelines," below). Sample FIXTABLE coding is:

```
//       EXEC ASMPCL,Q=USR,NAME=FIXTABLE,LMOD=FIXTABLE
//ASM.SYSIN DD *
FIXTABLE CSECT
         ICOMFIX ADDR=BTVRBTB,PAGES=10,ID=FTD
         ICOMFIX ADDR=SYCTRL,PAGES=10,ID=BKD
          .
          .
         END
/*
```

   Add table specifications for the PGFX command, if used (see System Control Commands).

6. The following INCLUDE statements for the Intercomm linkedit deck are generated automatically by the ICOMLINK parameter VS=YES:

```
    INCLUDE SYSLIB(VSINIT)           -resident or startup overlay
    INCLUDE SYSLIB(KEYFLIP)          -resident
for Page Preloading:
    INCLUDE SYSLIB(PMIPGLD,LOADPAGE) -resident
for Page Fixing:
    INCLUDE SYSLIB(PMIFIXA)          -resident or startup overlay
    INCLUDE SYSLIB(INTPAGE)          -resident (MVS only)
    INCLUDE SYSLIB(PMIFIXB)          -resident or loadable subsystem
    INCLUDE SYSLIB(FIXTABLE)         -resident
```

7. ORDER statements should be used to group modules and force page boundary alignment. For example:

```
Intercomm Nucleus--
          ORDER   SPA(P),SPAEXT,INTSCTDD
          ORDER   PMISUBL2,PMIRTLR,PMILINK2
          ORDER   PMIRTL4,PMIPREL4,PMIPRELR
          ORDER   LOADPAGE,PMIPGLD                          (if used)
          ORDER   IXFDSCTA
          ORDER   BITSECT,LOGPUT,STARTWRC,MSGAC000
          ORDER   MSGCOL,PMIRETRV                (if not in Link Pack)
          ORDER   PMICLZZZ,SCTINDX,FQES,SYCTRL,RMPURGE,RMNADISA
          ORDER   FDITCB,RMPC,RMFNQ,RSMGMNT,RMTRACE
          ORDER   POOLACCT,POOLCONS,ICOMINX,ICOMCHN,ICOMPOOL,POOLEND
          ORDER   KEYFLIP,EXECWAIT,IJKDSP01
          ORDER   STATINDX,BINSRCH,FESEND
          ORDER   PMIEXTRM                       (if not in Link Pack)
          ORDER   INTVRB00              (control region only, if MRS)
          ORDER   BTVRBNDX,FEINDX       (control region only, if MRS)
          ORDER   ATTIDTBL,AIDDATA      (control region only, if MRS)

VTAM Front End--
          ORDER   VCT(P),VTIDTABL,ICIDINDX,VTIDINDX,VTAMCLZZ

BTAM/TCAM Front End--
          ORDER   BLHIN(P),REQBUF00,BTSEARCH
          ORDER   AIDCSECT,HEADSECT,IECTTRNS
          ORDER   BTADPL,BTPOLFLP,BTAMSECT
          ORDER   BMH000,OUT3270,BLHOT,QUEUEMOD,BTAMCLZZ
          ORDER   GFEINTFC,TCAMINTF,TCAMASYN,IEDQB1       (TCAM only)

And, if BTAM switched lines used--
          ORDER   BTBACKSP,PEXCSECT,DIALSECT
```

8. If an overlay structure is not needed, remove all OVERLAY and INSERT statements from the Intercomm linkedit deck generated by ICOMLINK; code OVLYSTR=NO (default) and TRANS=NO (default) on ICOMLINK. Also redefine overlay subsystems (if any) to resident or dynamically loadable in the SCT (see Chapter 3).

7.9.4 Page Fixing Guidelines

Frequently referenced pages, that is, pages referenced by every transaction, such as the Front End line handlers, should be fixed. The following is a list of suggested Intercomm Csects to be fixed, arranged in order of importance:

1. First consider fixing:

   Intercomm nucleus Csects listed above.

2. Then, if possible, fix:

   a. BTAM Front End Csects listed above (or VTAM if used), and the Csect continuing the Front End network definitions. Also if priority queuing used, add:

      PMIPRNDX,BTMPRNDX,VTMPRNDX

   b. File Handler Routines (if not in Link Pack)

      IXFMON01, IXFSUBS, IXFVSCRS (VSAM), IXFLOG (if used)

   c. Store/Fetch and MMU Routines (if not in Link Pack)

      INTSTORF, MAPIN, MAPOUT, MMUVTBL, EDITTRTS, DEVDESC, DEVDESCU, MMUED001-3, MMUED008, MMUDDM, MMUDDMU, etc.

   d. Output Utility & Back End Tables

      PMIOUTPT, SUBOUTPT, PMISERCH, TERMCONV, DVASN
      PMITRTAB, PTRNTBLE, PMIFILET, PMISTATB, PMIDEVTB

   e. Edit Utility, Tables & Subroutines

      EDITCTRL, SUBEDIT, FIXEDIT, VERBTBL, PADDTBLE, EDITRTNS, EDIT3270, EDIT0001-nnn

   f. High Level Language Interfaces

      PREPROG, COBREENT, COBPUT, COBSTORF, PMICOBOT
      PREPLI, PREPL1, PMIPL1
      REENTSB1, DYNLSUBS

   g. Extended Security System

      INTSEC00

3. If executing in a Multiregion environment, then

   a. For the Control Region:

      MRINPUT, MRQMNGR, MRXQMNGR

   b. For each Satellite Region:

      MRINPUT, MROTPUT, MRXOTPUT, MRCSAMOD (if MVS)

4. Then, as required, fix:

   Resident user modules frequently referenced such as message routing subsystems, critical response subsystems, etc.

For a small-scale system (one megabyte or less of real storage), it is not wise to fix too much storage. There will be a trade-off point where pages that are not fixed will be paged in and out extensively. It is advisable to fix not more than one-third of the linkedit size, and then fix more on a trial and error basis.

It is important to realize the interrelationship between the FIXTABLE and the ORDER statements placed in the linkedit. Since pages can only be fixed on a page boundary, the address specified in the ICOMFIX macro will be rounded down to the next lower page boundary (unless it is already on a page boundary). This might result in overlap of pages fixed, which does not cause any error conditions. However, VS might unexpectedly not fix the last part of a certain module. To avoid this, the length of each module to be fixed should be calculated from the Csect sizes in the linkedit, and then corresponding ORDER statements should be inserted in the linkedit deck, and another linkedit executed.

7.9.5 VS System Tuning Considerations

If using LOADPAGE, it is important to have enough ICOMPOOL blocks for LOADPAGE save areas. If the LOADPAGE save area is acquired from subpool zero, two page faults may occur and therefore LOADPAGE saves only one page fault. Thus, the LOADPAGE save area should be obtained via ICOMPOOLs to maximize system performance and prevent unnecessary page faults. The length of the save area is defined by the EQU labeled WORKLEN in the LOADPAGE module.

There must, additionally, be as many LOADPAGE save areas as there are ECBs in LOADPAGE as specified by the PLECBT EQU value in LOADPAGE. The number of ECBs used may affect system performance. The number of ECBs can be changed by altering the number of fullwords (released as 8) defined at the label PLECB in LOADPAGE, then reassembling LOADPAGE.

This technique is also applicable to PMIPGLD save areas.

7.9.6 Subsystem Considerations

In lieu of defining subsystems in an overlay structure, a VS parameter may be specified in the SYCTTBL macro, EXGRP=n, defining groups of subsystems which may be allowed to process messages concurrently. Its purpose is to prevent all resident application subsystems from executing concurrently upon receipt of a message, since each message processing thread requires save areas, message areas, I/O areas, etc., which could result in a massive page-in/page-out operation if enough real storage is not available. If the EXGRP parameter is specified, only one of the execution groups will be processing at a time, and paging will be reduced. Using this scheduling technique, all subsystems are defined as resident (SYCTTBL macro parameter OVLY=0) and the Subsystem Control Table entries must be sequenced by EXGRP number (in ascending order). (The execution group number becomes the "overlay number" in the generated SCT control byte SCTPONU.) The SCT Index must be generated, as described in Chapter 3, via the GENINDEX macro, or if user-coded; each index entry defines an execution group of SCTs. Also, TRAFFIC (Csect in module TRAFFICQ) should be ordered with SYCTRL.

If page preloading is not in use, disk queuing for messages is often more efficient than core queues under VS due to I/O activity overlapped via the standard File Handler facilities.

If page preloading is in use, storage queuing should be used since the paging activity (which is more efficient than BDAM I/O) will be overlapped.

The RTNLINK macro (for Assembler Language subsystems) also has a VS-oriented parameter called PRELOAD, which, when coded, causes the page that contains the high order save area to be preloaded and therefore prevent a page fault.

7.9.7 VS SYSGEN Considerations

For effective performance, observe the following points when specifying VS SYSGEN parameters:

- Execute Intercomm at the highest possible priority.
- If executing under VS1, separate page (EPS) data sets onto different packs (channels) from those containing SYS1.SYSWADS and SYS1.SYSJOBQE data sets.
- Separate JES spooling data sets onto different packs from page (EPS) data sets.
- Separate frequently used Intercomm and on-line user data sets onto different disk drive channels from those used for operating system and page (EPS) data sets.

- Specify locations of data sets carefully; that is, put the System Catalog between SYS1.LINKLIB and SYS1.SVCLIB.

- Choose carefully between resident and SYS1.LINKLIB SVCs; that is, OPEN/CLOSE should be resident, if possible.

7.9.8 VS1: WTP User Message Limit

Appearance of the following message indicates that the problem program WTP message limit has been reached for this job step.

```
VS1:WTP IEF094I WTP USER LIMIT REACHED FOR ...
```

The limit is set at SYSGEN and/or at IPL time. The SYSGEN defaults to a maximum of 15 messages. If your CPU console is also Intercomm's control terminal, messages to it will be limited by the STEPWTP parameter of JES. After the limit has been reached, any Intercomm messages (Intercomm is the problem program) sent to the console will be ignored by the WTP routine. See also the SPALIST macro PMIWTO message routing override parameters described in Section 7.4.1.

7.9.9 VS2: SPIE Macro

SPIE macro expansion under OS differs from VS2. Execution under OS of modules containing the SPIE macro assembled under VS results in abnormal job termination when an OCx program interrupt occurs, despite inclusion of SPIEEXIT software. Such OS-assembled modules execute successfully under both OS or VS, but VS-assembled modules containing SPIE execute successfully only under VS and not OS.

Intercomm members containing the SPIE macro are STARTUP3, FQES, PREPLI, PREPL1 (and PL1INTFC macro), STAEEXIT, STAERTRY, and INTSPA. Also, the off-line modules LOGANAL and MRBATCH. When running a shared VS and OS Intercomm, any assemblies of the above should be done with the OS Macro Library. In general, all of the above modules should be reassembled if included in the linkedit, unless already reassembled at installation time or executing under MVS.

7.10 MVS OPERATION

Installation and operation of Intercomm under MVS require a few considerations in addition to the general VS installation procedures described in Section 7.9. The MVS user should consider:

- Each live Intercomm region must run as a nonswappable task. In order to make Intercomm nonswappable, the name of the Intercomm nucleus may be placed in the operating system's Program Properties Table (PPT) in key 8 to 15, or the SYSEVENT macro may be inserted in STARTUP3 (see Technical Information Bulletins). For a BTAM Front End region with remote terminals, the operating system automatically marks the task nonswappable. Additionally, Intercomm should have the highest dispatching priority in the system (above TSO, if used).

- Because BTAM dynamic buffering is not supported, Intercomm suppresses dynamic buffering under MVS. Therefore, the LINEGRP macro, BUFL parameter, must specify a value at least as large as the longest message expected, with the exception of bisync devices (see Basic System Macros). LINEGRP macro, BUFNO parameter, must specify a value at least as large as the value assigned to the NUMLN parameter (the number of BLINE macro instructions subordinate to the LINEGRP).

- Concatenate SYS1.AMODGEN after SYS1.MACLIB in all Intercomm procedures (including INTASMF) executing the assembler.

- The following Intercomm modules must be reassembled and linkedited: STARTUP3, STAEEXIT, IXFHND01, PMISNAP1, STAERTRY, TRAP, FEMSG, SNAPRTN, and, if used, BLHTRACE.

- If executing in a Multiregion environment, the following must be reassembled: MRLOGOT (if used), MRPURGE, MRQMNGR, MROTPUT, MRINTER and MRCSAMOD (also add to linkedit).

- INTPAGE must be included in the Intercomm linkedit (resident) for page fixing.

- GAMFQES instead of FQES must be used in the linkedit.

- BTAM (and TCAM) Front End modules, particularly BTSEARCH (which contains the BTAM RESETPL macro), must be reassembled, if in the Intercomm linkedit. Also reassemble every time an operating system upgrade is made.

- The entire VTAM Front End, if used, must be reassembled. Also reassemble every time an operating and/or VTAM system upgrade is made. Ensure that the correct VS system library containing VTAM macros and Dsects is in the SYSLIB concatenation stream.

- Reassemble the Front End Network Table due to possible changes in DCB, ACB and RPL macros. Also see Technical Information Bulletins regarding a user modification to the Intercomm LINEGRP macro (due to optional IBM PTF to DCB macro).

- Eliminate the subsystem overlay structure, if at all possible; convert subsystems to dynamically loadable, or define as VS execution groups. Eliminate internal overlay structure subsystem linkedits, if previously used.

- If previously used in a non-MVS environment, the Intercomm Interregion SVC must be reassembled and reinstalled. It is required to use either page preloading or page fixing, a Multiregion system, ESS, and cross-region sharing of VSAM files.

Invalid installation of the Interregion SVC may result in a dump simulating a SOC1. If executing under MVS/SE, use of PMIPGLD may impact system performance; remove if degradation occurs. In addition, if subsystem time-outs occur, remove LOADPAGE.

If executing with a Multiregion environment, further considerations apply under MVS. These are fully documented in Multiregion Support Facility.

An operator cancel (S122 and S222) will not give control to final cleanup processing in the STAEEXIT routine. Therefore, PMIDEBUG should be included in order to cancel Intercomm with a dump. This is also recommended for flushing the Intercomm log buffers and closing the log, and closing VSAM files. A system x22 cancel will not accomplish this. See Messages and Codes for a description of STAEEXIT processing and the use of PMIDEBUG. Coding DEBUG=YES on the ICOMLINK macro for the linkedit generation forces an include for PMIDEBUG.

Intercomm and the MVS operating system components which affect Intercomm execution must be tuned on an ongoing basis. See Chapter 11 for general tuning recommendations, plus those specific to execution under MVS.

7.11 INTERCOMM INTERREGION SVC--&MRSVC

The member IGCICOM on SYMREL is a type 1 SVC routine which provides for interregion communication or general use in protect key zero, and is required for page fixing in a VS1/MVS installation, Multiregion installation, the Extended Security System, and cross-region sharing of VSAM files. The SVC must be reinstalled if converting to MVS.

The Interregion SVC performs the following functions:

- Posts an ECB in another region
- Waits on an ECB in another region
- Executes Intercomm system functions in protect key zero

To implement the Interregion SVC, the following steps must be taken:

1. Assign a number for a type 1 SVC for Intercomm use.
2. Modify the global &MRSVC (member: SETGLOBE) to reflect the number assigned in Step 1. (&MRSVC is released with a value of 013. Execution of an Intercomm routine requiring the SVC, without resetting &MRSVC and assembly and linkedit of the SVC routine, will result in a user abend with a random identification.)
3. Assemble and linkedit:
   - IGCICOM as IGCnnn, where nnn is the assigned SVC number; linkedit parameters are LIST,LET,DC,RENT
   - KEYFLIP (all regions for a Multiregion system)
   - INTSPA (all regions for a Multiregion system)
   - MRBATCH (if used)
   - Relinkedit OS/VS Nucleus (IEANUC01: to include SVC); add a DD statement for library containing IGCnnn (usually SYS1.SVCLIB or MODLIB).
4. When structuring the linkedit deck for the Intercomm load module, KEYFLIP must be included before the Intercomm Dispatcher (IJKDSP01). Do not include KEYFLIP unless the SVC has been installed. Otherwise, miscellaneous program checks will occur.

Invalid installation of the Intercomm Interregion SVC is signaled by an IBM error message.

7.12 INTERCOMM LINK PACK FEATURE

The Intercomm Link Pack Feature allows operation of more than one Intercomm region (live, simulated, batch, or Test Mode) simultaneously without duplicating identical Intercomm routines from region to region. Various Intercomm routines are linked together and loaded into the Link Pack Area and shared among the various Intercomm regions. (See Figure 7-5.) Such routines are generally used more than others; over 100K of storage is saved. Since the Link Pack Area is separate from the Intercomm area, the Intercomm system is therefore divided into two interfacing sections, the Link Pack Module (LPM) containing the Link Pack routines and the Intercomm Region (IR).

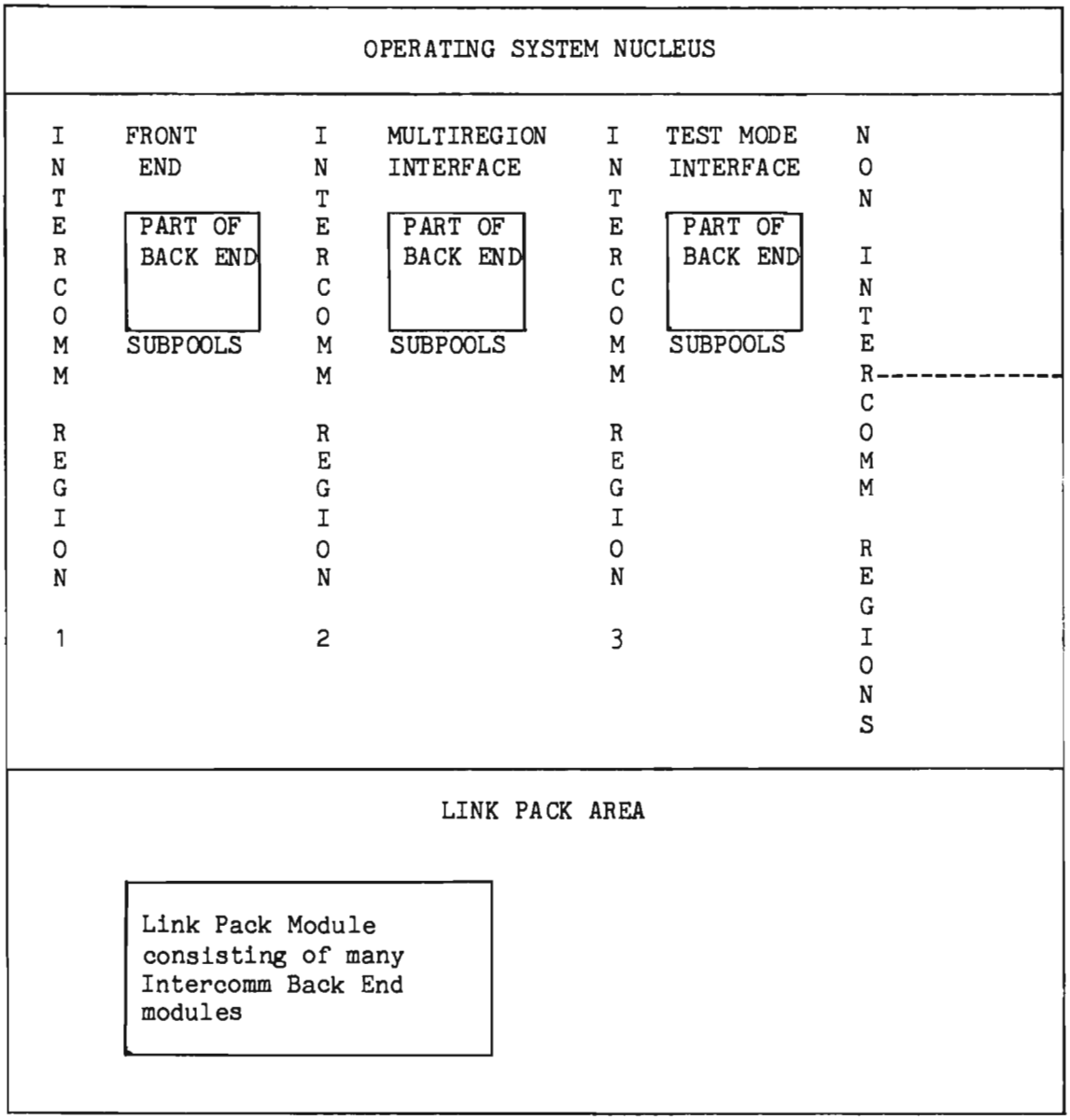


Figure 7-5. Link Pack Module Working in Conjunction With Several Intercomm Regions

An additional advantage is that reentrant user Assembler Language routines to be executed under Intercomm may also be placed in the LPM in the Link Pack Area.

Entry point names for all Intercomm Link Pack modules are defined to the Intercomm region via the interface module LPINTFC. At Intercomm startup, the module LPSTART initializes VCONs for those entry points with actual addresses of Link Pack routines, using the Link Pack resident interface routine LPSPA. It should be noted that the LPINTFC and LPSPA modules provide entries for all Link Pack eligible Intercomm modules. The physical makeup of the Link Pack Module is only determined by the linkedit of the LP and Intercomm regions. VCONs in LPSPA will be unresolved for those Intercomm components that the user has chosen to keep in the Intercomm region. LPSTART will issue a PMIWTO for each of these modules indicating that they have been resolved within the Intercomm region, and not in the Link Pack Area.

During Intercomm execution, LPINTFC loads the actual address of a called Link Pack routine using the initialized VCONs which reside in the System Parameter Area. Startup initializes word 1 of the OS save area with the System Parameter Area address for subsequent use by the LPM routines. Intercomm components eligible for the Link Pack Module are illustrated in Figure 7-6, along with their corresponding entry point names.

Preparation necessary to utilize the Link Pack Feature is provided in the following. Macros discussed in these sections are all described in Basic System Macros.

Component	Entry-Point Name	Module(s) To Include For LPM Linkedit
Message Collection	MSGCOL	BLMSGCOL
Retriever	RTRVER	PMIRETRV
Edit Utility	EDIT	PMIEDIT, PMIFIXED EDIT3270*
Output Utility	OUTPUT	PMIOUTPT, (USROUTCK), PMIVMI56*
Change/Display Utilities	CHANGE or DISPLAY or CHGDIS	CHANGE, DISPLAY FORMAT, CRUNCH
File Handler (IXFQISAM not in Intercomm linkedit)	FILEHND	IXFHND01, IXFVSCRS*, IXFB37*, IXFLOG*
File Handler (IXFQISAM in Intercomm linkedit)	FILHNDQI	IXFHND01, etc.(see above)
Store/Fetch	SFETCH	INTSTORF
Dynamic Data Queuing	DDQ	DDQMOD
Message Mapping Utilities	MMU	MAPIN, MAPOUT, MMUTRTS, MMUED001-003 & -008, LOGCHARS (or username), MMUDDM, MMUDDMU, MMUDDMX, MMUDDMT, MMUDDMF, MMUDDMM MMUCOMM**
Terminal Lookup	PMIEXTRM	PMIEXTRM
Enqueue/Dequeue Functions	NQDEQ	PMINQDEQ
Conversational Support	CONVERSE	CONVERSE

* Include only if used for corresponding feature (3270 terminals, VSAM files, etc.)

** LPENTRY definition required (see 7.12.6)

Note: If the user exit USROUTCK is to be used with PMIOUTPT, it should also be included, and must be reentrant.

Figure 7-6. Applicable Intercomm Components for LPSPA/LPINTFC Macro

7.12.1 Preparation of the Operating System

On the SUPRVSOR macro, code: OPTIONS=COMM. This coding allows modification of the Link Pack Area at Initial Program Load (IPL) time.

Create a member on SYS1.PARMLIB named IEAIGGxx, where:

- IEAIGGxx specifies the name (LPSPA) of the member containing the Intercomm Link Pack Module, and
- xx is the suffix used when the operator replies "RAM=xx' at IPL.

If the Link Pack Module does not reside on SYS1.LINKLIB, LNKLST00 must be properly coded, as indicated in the IBM manuals.

The Link Pack Module should not reside on a STEPLIB or JOBLIB library of an Intercomm region. If it is, the Link Pack Module would then be loaded into the Intercomm region, rather than using the copy in the Link Pack Area.

7.12.2 Preparation of the Link Pack Module (LPM)

Assemble and linkedit the LPM interface routine LPSPA using the LPSPA macro, as follows:

```
//            EXEC     ASMPCL,LMOD=LPSPA,Q=USR,
//            PARM.LKED='XREF,LIST,LET,NCAL,RENT'
//ASM.SYSIN   DD       *
              LPSPA    A=A,MODS=(.....)
              END
/*
```

RENT must be specified to cause the linkage editor to flag the output as reentrant. Additionally, each module in the LPM must be relinked as reentrant: use the LKEDP procedure and override the linkedit parms by adding RENT; place the output load module in MODUSR.

Example of LPSPA macro:

```
LPSPA A=A,MODS=(MSGCOL,RTRVER,FILEHND)
```

This generates an LPSPA Csect to be used to place Message Collection, the Retriever and File Handler modules in the Link Pack Area.

Linkedit the LPSPA Csect, together with all other component modules to go in the Link Pack Area, as follows:

```
//           EXEC    LKEDP,Q=USR,LMOD=LPSPA,
//           PARM.LKED='RENT,....'
//LKED.SYSLMOD DD    Library to contain LPM
//SYSIN      DD      *
           INCLUDE SYSLIB(LPSPA)
           INCLUDE SYSLIB(member-name,member-name,...)
           ENTRY   LPSPA
           NAME    LPSPA
/*
```

- RENT must be specified to cause the linkage editor to flag the output as reentrant.
- NAME statement specifies the Link Pack Module name (LPSPA) as it appears on IEAIGGxx on SYS1.PARMLIB.

member-name specifies the module(s) to be included for each of the applicable Intercomm components, as defined in Figure 7-5.

Under MVS, LPSPA can be placed in the MLPA data set, as long as the data set is in the system catalog.

In order for Intercomm system programs to provide Csect names for Link Pack Modules in messages and reports, add the following DD statement to the Intercomm execution JCL after the //PMISTOP DD statement:

```
//LPSPALIB  DD  DISP=SHR,DSN=name-of-library-containing-LPSPA
```

7.12.3 Preparation of Intercomm Region (IR)

Assemble and linkedit the IR interface routine LPINTFC using the macro LPINTFC, as follows:

```
//           EXEC     ASMPCL,LMOD=LPINTFC,Q=USR
//ASM.SYSIN  DD       *
           LPINTFC  MODS=(.....)
           END
/*
```

Create a standard Intercomm linkedit deck using the output of the ICOMLINK macro, as described in the previous sections. Delete the INCLUDE statements for the modules put in the LPM, or relinkedit the Intercomm Region load module, as indicated in Figure 7-7. Only the REPLACE statements for the Csect names of the specific components in the previously created Intercomm Link Pack Module should be utilized.

```
//          EXEC     LKEDP,Q=xxx,LMOD=name
//SYSIN     DD       *
            ENTRY    PMISTUP
            INCLUDE  SYSLIB(LPSTART)
            REPLACE  MSGCOL              Message Collection
            REPLACE  PMIRETRV            Retriever
            REPLACE  EDITCTRL            Edit
            REPLACE  FIXEDIT             Edit
            REPLACE  PMIOUTPT            Output
            REPLACE  IXFMON01            File Handler
            .
            .
            INCLUDE  SYSLIB(Intercomm)
            INCLUDE  SYSLIB(LPINTFC)
            OVERLAY  AB                  Add to Startup
            INSERT   LPSTART                 Overlay (if used)
            .                            remainder of overlay
            .                            structure from
            .                            original linkedit (if used)
/*
```

Figure 7-7. Relinkediting Intercomm Region for Link Pack Feature

7.12.4 User Routines in the Link Pack Area

Prior to placing a non-Intercomm module in the LPM, the following preparation is required:

1. Coding Conventions

   The module must be coded in Assembler Language and obey certain coding restrictions, as described below.

2. Entry Point Specifications

   A VCON for the module's entry point must be assigned via the User Spa, or the Subsystem Control Table. In the latter case, the module would have to be a subsystem and the SYCTTBL macro will automatically provide the VCON.

NOTE: Verify that the SYCTTBL macro(s) for subsystem(s) in the Link Pack Area do not specify that the subsystem is dynamically loaded. Also, under VS, if messages for the subsystem(s) are to be scheduled as if the subsystem(s) were in overlay (EXGRP not equal to 0) the CKLINK module should be removed from the Intercomm linkedit.

7.12.5 Coding Conventions for User LPM Routines

Modules in the LPM may not have TEST=YES coded for any LINKAGE macros. Since modules in the LPM are not linkedited with the rest of Intercomm, many external symbols cannot be resolved at linkedit time. Any unresolvable external references needed by the module during execution must, therefore, be provided, directly or indirectly, through the System Parameter Area. Likewise, when subsequently passing control to routines not in the Link Pack Area, a normal "CALL entry,..." will not work.

To overcome this limitation, the System Parameter Area and Spa Extension are used as a communication area between the IR and the LPM, and vice versa. The User Spa is also available for any external symbols required by the user which are not already present (see Chapter 3). The SPALIST Dsect contains labels of User Spa fields; therefore, the problem of unresolved external symbols is reduced to obtaining the addresses of the SPA (also for User Spa areas) and SPAEXT.

All subsystems are given the address of the System Parameter Area upon entry to the subsystem. In the exceptional case when the SPA address is not passed as a parameter to a module in Link Pack, the GETSPA macro can be used. The macro obtains the SPA address from word 1 of the OS save area which was initialized during startup. The macro is coded as follows:

```
GETSPA REG=r
USING SPALIST,r
```

Execution of the macro loads the System Parameter Area address in the register specified for the REG parameter. The most frequent uses of the Spa and Spa Extension are for the macros shown in Figure 7-8. The Spa extension address is in the Spa at the label SPAEXTAD and may be placed in a register by use of a load (L) instruction. Do not forget USING statements to establish addressability for the SPALIST and SPAEXT Dsects.

Finally, any module in Link Pack must be reentrant. If it attempts to modify itself, an OC4 program check will result. Furthermore, each load module included as linkedit input when the LPM is created must have been itself linkedited with PARM='RENT,...'.

```
CALL
            L     15,SPAWRITE              V(WRITE)
            CALL  (15)

STORFREE
            L     15,SPAFREE               V(STORFREE)
            STORFREE ..,LINK=(15)

INTWAIT
            L     14,SPAKINT               DISPATCH ON INTERVAL
            DISPATCH ..,INTVL=n,LINK=(14) V(IJKINT)

                              or

DISPATCH
            L     r,SPAEXTAD               SPA EXTENSION ADDRESS
            USING SPAEXT,r
            L     14,SEXKDSP               DISPATCH EXECUTE
            DISPATCH ..,LINK=(14)          V(IJKDSP)
```

Figure 7-8. Frequent Uses of System Parameter Area and SPA Extension in User LPM Routines

7.12.6 Entry Point Specifications for User LPM Routines

Even if a module requires no external symbols after acquiring control, it cannot get control at all unless its entry point can be found. Therefore, each entry point to which control can be passed from the Intercomm region must have a VCON reserved in the User Spa, having a label beginning with SPA. At Intercomm startup, all such VCONs will be initialized to point to the Link Pack entry point, if it is resolved in the LPM and unresolved in the IR.

The LPENTRY and LPVCON macros define user entries in the LPSPA and LPINTFC, respectively. These macros must be coded and assembled with the respective Csects, as described below. For each entry point in the LPM which is to be called symbolically, that is, "CALL name", instead of "CALL (15)", the LPINTFC can be coded as follows:

```
LPINTFC MODS=(....)
LPVCON  ...
LPVCON  ...
LPVCON  ...
END
```

One LPVCON macro must be coded for each such entry point, except if a subsystem is referenced via a SYCTTBL macro.

Similarly, when creating the LPSPA Csect, an LPENTRY macro must be coded defining each user entry point to the LPSPA.

```
LPENTRY ...
LPENTRY ...
LPENTRY ...
LPSPA   ...
END
```

NOTE: All LPENTRY macros must precede the LPSPA macro, and must be coded for subsystems as well as subroutines.

Assume a user subsystem has an entry point of SUBSYST, a subsystem code of 'X', and a user subroutine has an entry point of LPROUTN. First, a VCON would have to be assigned in the User Spa; suppose that---

```
SPAROUTN   DC   V(LPROUTN)
```

was coded in the User Spa. None is required for SUBSYST, since it is a subsystem.

In the LPINTFC, for example---

```
LPVCON   LPROUTN,SPA,ROUTN
```

would be coded to create a pseudo entry point for LPROUTN in the LPINTFC CSECT. No LPVCON macro is required for the subsystem.

In the LPSPA---

```
LPENTRY   LPROUTN,SPA,ROUTN
```

would be coded for LPROUTN, and the following would be coded for the subsystem:

```
LPENTRY   SUBSYST,SCT,SSC=X
```

This would make it possible for these modules to be linkedited into the LPM and receive control from an Intercomm region. Should they have any unresolvable external references, additional entries might be needed in the User Spa.

7.12.7 Accessing LPM Modules in Batch Mode

If a batch program (not Test Mode or simulated Intercomm) needs to access the Intercomm File Handler (for example) in the LPM, an interface program has to be coded as described in the chapter on the File Handler. Additionally, the modules LPSTART, MULTISPA, and LPINTFC must be included in the linkedit. For Store/Fetch, see Store/Fetch Facility for additional linkedit requirements for a batch program.

Chapter 8

INTERCOMM FACILITIES

8.1 INTRODUCTION

This chapter provides general descriptions and implementation procedures for several testing, debugging and tuning facilities available to the Intercomm user. Where necessary, references are made to related Intercomm manuals. The following facilities are defined:

- Terminal Simulator (BTAMSIM)
- Abend Intercept Routines (SPIEEXIT,STAEEXIT)
- Indicative dumps
- System DCBs
- Spinoff snaps
- Fast Snap Facility
- System Accounting and Measurement Reports
- System Tuning Statistics
- Log Input Facility
- Test Mode operation

8.2 TERMINAL SIMULATOR FACILITY

The Intercomm terminal simulator module (BTAMSIM) allows the entire Intercomm system, Back End, BTAM Front End, and application programs to be executed as if it were receiving input and sending output to terminals, without actually having those terminals. The terminal simulator allows the testing and debugging of the system in a manner which closely approximates a live environment. It also allows an early evaluation of system performance and an indication of the response time when the system goes live.

The simulator permits a combination of a real BTAM, TCAM and/or VTAM Front End, and BTAM line groups with only simulated terminals of the following types:

- IBM 3270 Locals (Models 1 and 2)
- IBM 2740 (Models 1 and 2)
- IBM 2780 (output only)
- IBM 2741
- IBM 2260 (remote)

To use the simulator, the user must perform the following steps:

1. Include BTAMSIM as resident in the Intercomm linkedit
2. Create an input data set for each terminal to be simulated
3. Supply a DD statement for each input data set, using the name of the terminal as the name of the DD statement
4. Create a SIMCARDS input parameter card data set
5. Supply a DD statement for the SIMCARDS data set

The following subsections contain detailed discussions of each of these steps. In addition, the BTAM Front End must be installed, and the simulated terminals must be completely defined in the BTAM Front End tables, as described in the BTAM Terminal Support Guide. Also, the terminals must be properly defined in the Intercomm Back End Station and Device Tables. A BLINE and BTERM for the CPU console as a live terminal is optional. A control terminal (live or simulated) is required (same TID as SPALIST--CCNID parameter). Under Multiregion, the simulator may not be used in a satellite region.

8.2.1 Terminal Input Data Set(s)

A data set must be created for each terminal to be simulated containing the pseudo-input from that terminal. Each of these data sets must be variable-format sequential files and is created using the CREATSIM utility program. This program is described in Chapter 12, "Off-Line Utilities."

The DD statement for each data set must have as its ddname the name of the terminal. For example, the simulated input data set for a terminal named CNT01 would have a ddname of CNT01.

The input data set(s) will be accessed using the File Handler; therefore, the DD statement must specify DCB=DSORG=PS. Input records on the simulator data set must be variable length and must follow (in EBCDIC) the exact format of what would normally be received from the terminal. All control characters, EOBs, ETXs, etc., must be contained in the record.

NOTE: When simulating CRTs, if an input message causes multiple output messages to be sent back to the terminal, it should be followed on the input data set by a RLSE system control command for each expected output message after the first. There is no physical output from the simulator.

8.2.2 Input Parameter Data Set

The simulator expects an input parameter data set with the ddname SIMCARDS containing one parameter card for each terminal with an input data set. Each card has the following format, starting in column one:

ttttt,iii,pp,ss

where:

- ttttt represents the terminal name, for example, CNT01 (ddname of input data set).
- iii represents the interval in seconds between input messages, left-padded with zeros to three positions.
- pp represents the number of passes to be made through the data set. The field is optional and has a default of 1.
- ss represents the number of initial records to skip on passes through the data set subsequent to the first pass. This field is optional and has a default of zero. If it is specified, pp must also be specified.

8.2.3 Input Operations

At startup, the Front End attempts to open every TP line. If the line DD statement is missing (as specified via the LINEGRP macro), the line cannot be opened. This is a likely indication that some or all terminals on that line group are to be simulated. If some terminals in the line group are to be simulated, the terminals which are not to be simulated cannot be live (no DD statement for the lines or local units). Also, the BTERMs for nonsimulated terminals on those lines must specify TPUP=NO, while those for simulated terminals must specify TPUP=YES. The control terminal may not be simulated in a mixed environment with live terminals.

The terminals to be simulated are identified by checking which DD statements are supplied with the terminal name ttttt for the ddname. If a terminal is to be simulated, there must be an input data set and a SIMCARDS input statement for that terminal.

Assume that SIMCARDS has the following parameter card:

CNT01,030,03

In this example, any time the Intercomm Front End encounters the terminal name CNT01 for processing, the simulator tests to see if there should be a simulated read from the terminal (or a simulated write to the terminal). If a simulated read is required, a record is read from the data set represented by the CNT01 DD statement and is passed back to the Front End.

The Front End conversational facility is supported because an interval is started only at the time a message is read from a terminal.

The simulator program operates by changing the READ/WRITE routine address in the DCB of the LINEGRP macro expansion for the terminals to be simulated to point to an entry point in BTAMSIM instead of to the normal BTAM READ/WRITE routine. Therefore, whenever the Intercomm Front End issues a read or a write, the simulator acquires control. For read operations, the simulator determines which terminals are being simulated. For each terminal provided with a data set and defined to the simulator by a parameter card on the SIMCARDS data set, records representing a message are read in from the terminal's data set and passed back to the BTAM Front End at the interval specified in the iii field of the parameter card.

If the end of the data set is reached, and if the pp field of the parameter card is greater than 1, the simulator will start over again at the beginning of the data set. It will perform as many passes as prescribed in the pp field. On these additional passes, it will bypass the first ss records of the data set, if requested.

8.2.4 Output Operations

Whenever the Front End issues a write to a terminal, the simulator acquires control in the same manner as for a read. For write operations, there is very little to be done. The simulator delays returning control to the Front End for an interval that it calculates to be the approximate line-time required to send the message to the terminal as if the terminal were really there. The simulator then returns to the Front End with a successful completion code. An exception to this is for buffered hard copy devices, where the simulator will return to the Front End a buffer-busy code, if the current message is being written to the terminal before the terminal's buffer could have dumped a prior message. The Front End then retries at a later time, as it would in a live environment.

Output messages to simulated terminals are ignored (and freed) by the simulator. However, they can be examined on the INTERLOG data set which contains all output queued for the Front End (F2 and F3 log codes).

8.2.5 Local 3270 Message Preparation and Processing

To simulate formatted screen input messages for a 3270, use the CREATSIM utility to create the input message file with SBA sequence cards defined for each input field. CREATSIM can also process positional input messages interspersed with formatted input messages.

To produce a printed listing of all input and output 3270 messages, also include SIM3270 as resident in the Intercomm linkedit and define a SYSOUT DD statement for each local 3270 terminal (CRT or printer) to be simulated, as follows:

```
//SCRxxxxx    DD    SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=121,RECFM=FA)
```

where xxxxx is the terminal-ID of the terminal being simulated.

The printed listing provides a display (with attribute indicators, where applicable) and also the message data in EBCDIC as it would be sent to, or received from, the terminal. Appropriate error messages for invalid 3270 orders, etc., are also printed. A CRT display is updated for each message to illustrate how it would look had the actual terminal I/O been performed. SIM3270 expects each input message to start with a one-byte hex AID key value, followed by a two-byte physical cursor address, then an SBA sequence for the first field if formatted input, or a message text string if unformatted.

Use of SIM3270 requires that the Intercomm Store/Fetch Facility be installed. Transient strings are created in core for the area of each simulated terminal and updated for each input and output message. These strings require definition of a Store/Fetch data set in case a flush is necessary. The default used is INTSTOR9; however, this may be changed to an existing Store/Fetch work data set ddname by modifying the global &SDD at sequence number 01760000 in SIM3270. (See Store/Fetch Facility for further details.)

8.2.6 Simulator Closedown

The simulator maintains a record of the number of terminals that have active simulated input files. If end-of-file has been reached on all of these files (or if the specified number of passes has been made through all of them), and if there are no live terminals operating, then the simulator will internally generate a NRCD closedown message to terminate the simulation. If some live terminals are also operating, however, this will not be done; the system will close down when the NRCD or IMCD transaction is entered from the control terminal, which must be a live terminal.

8.3 ABEND INTERCEPT ROUTINES--SPIEEXIT,STAEEXIT

Two system routines are provided with Intercomm to intercept abends and prevent the termination of task execution. These programs, SPIEEXIT and STAEEXIT, are referenced in the SPIE and STAE (ESTAE, if MVS) macros, issued at system startup. SPIE and STAE will not be issued if the modules are not included in the Intercomm linkedit.

8.3.1 SPIEEXIT

SPIEEXIT receives control in the event of any program check (OCx) condition, and then calls SPIESNAP to issue a Snap 126 and return control to the Dispatcher (thread zero) or Subsystem Controller (nonzero thread). Recovery may or may not proceed successfully based upon the cause of the program check. Many Intercomm service routines force an OC2 program check intentionally (via ISK instruction) when called with an invalid parameter list; in this case the associated message processing thread is terminated, and the system continues execution. If the program check occurs because of invalid table entries, or because a system routine is inadvertently destroyed by invalid program logic, recovery may not be successful. See also Section 7.9.9 for assembly considerations for modules containing the SPIE macro. SPIESNAP and PMISNAP1 must also be in the linkedit. The program check codes trapped via SPIEEXIT are controlled by the STUSPIE parameter of the SPALIST macro.

8.3.2 User SPIESNAP Exit--SPSNEXIT

A user exit routine is conditionally called by SPIESNAP to determine whether or not a snap 126 should be taken. This exit can be used to prevent a buildup of the snap data set by repetitive snap 126 calls. This exit will not suppress any other snap (see also Section 8.4.1, "User Snap Exit," below).

When control is passed to SPSNEXIT, R1 contains the address of the SPIEEXIT save area (formatted SPIE SAVE AREA described in Messages and Codes), and R0 points to the initialized text of the program check MP001I in standard WTO format (4-byte prefix); see the description for the USERWTO exit in Messages and Codes. Standard linkage conventions apply.

Upon return from the SPSNEXIT routine, if register 15 contains zero, the snap 126 will be taken. Any other value in register 15 indicates that the snap should be suppressed.

The routine must be closed; that is, it may not relinquish control to the Dispatcher. The routine must be resident.

8.3.3 STAEEXIT

STAEEXIT receives control in the event of any abend conditions other than program checks. The only valid situations for attempted recovery are that of the Dispatcher abend 909 indicating detection of a closed program loop by the routine IJKTLOOP, or a Multiregion cross-memory post failure (abend 557), if executing under MVS.

Recovery will be attempted only if the module STAERTRY is included in the linkedit (this module contains the IBM SPIE macro; if not executing under MVS, reassemble STAERTRY to ensure that the correct version of the macro is used for abend 909 recovery). Otherwise, STAEEXIT effects job termination via the same abend code, after issuing an informational WTO, capturing the current environment via a snap 122, closing files and flushing the log buffers. A SNAPDD DD statement must be present (not DUMMY) for the snap 122 to be issued; otherwise, an OS system dump will be written to SYSUDUMP, or SYSABEND, if present. If neither SNAPDD nor SYSUDUMP/SYSABEND are DUMMY, two full region snaps are produced. See Messages and Codes for a description of Snap 122, which is easier to debug.

8.4 INDICATIVE DUMP OPTION

When a program check or a time-out occurs, a full region/partition snap is produced by default. It is usually the case that only certain items in the snap are needed for debugging purposes. In order to reduce the size of the snap produced, Intercomm provides an option to produce a smaller indicative dump, which includes only those areas most likely to be needed for debugging. The user selects this option (for snaps issued by Intercomm) by specifying INDUMP=YES as a parameter on the SPALIST macro. When choosing this option, be aware that certain problems (for example, storage destruction) may only be solved from a full snap before the cause of the problem can be determined.

On a user-coded PMISNAP macro, the parameter INDUMP=YES may be coded to request an indicative dump, rather than a full snap. For a user-issued PMISNAP, this option applies to all snap-IDs. If INDUMP=YES was coded for the SPALIST macro, the user-issued PMISNAP option will be honored. Thus, this option can serve as a useful debugging tool, particularly for dynamically loadable subsystems and subroutines, by the insertion of the statement:

```
PMISNAP  ID=n,INDUMP=YES
```

in an Assembler Language program.

Indicative dump processing may be activated and deactivated by the INDUMP parameter of the STRT/STOP system control commands. This command option can be used to dynamically override the SPALIST macro specification. However, this option requires all user-coded PMISNAP macros to additionally contain operands for a normal snap, if INDUMP is turned off.

The indicative dump option is applied to Intercomm-generated snaps 126, 118 and 114. It is not applied to thread 0.

The storage areas printed in an indicative dump are described in Figure 8-1, in the order of their appearance. Other resource types owned by the thread are adequately identified in the associated thread resource dump.

Areas	Length	Note
SPIE SAVE AREA	136	1
Text of PROGRAM CHECK/TIMEOUT Message (MP001I/MS009I)	84	7
Vicinity (-16 thru +16) of Failing Instruction	32	7
Resource Manager Save Area (RMSAVE)	100	2
System Parameter Area and USERSPA (if any)	500+	6
SPA Extension	1500	7
ITCB (Intercomm Thread Control Block) for subsystem	40	7
SCT (of the associated subsystem)	100	7
Subsystem Controller Save Area for subsystem	200	3,7
Subsystem Input Parameter List	20	7
Subsystem Input Message	Variable	7
Subsystem, if COBOL, or nonreentrant Assembler or PL/1	Variable	4,7
SCT Extension, if dynamically loaded subsystem	100	
Loaded Subroutine (if any)	Variable	4
Resources owned by thread	Variable	5
STORAGE/LIST parameter storage areas (if any)	Variable	

Notes

1 Meaningful for snap 126 only (see Messages and Codes).

2 Meaningful for OC2 in Manager only.

3 May be chained down to thread-owned save areas which will appear in the snap as resources obtained dynamically.

4 The SPALIST macro, INDUMP parameter, specifies the length for snapping a nonreentrant subsystem; except that for a dynamically loaded nonreentrant subsystem, the entire load module is produced. The INDUMP parameter length value is also used for user-requested dynamically loaded subroutines, which are snapped with thread-owned resources.

Figure 8-1. Areas Displayed by Indicative Dump (Page 1 of 2)

Notes:

5 Up to 100 areas representing thread-owned resources appear in the snap (in the order acquired) as follows:

Resource Type	Area	Length
Core	Areas acquired	Area Lengths
Dynl	Dynamic loaded subroutine	Variable
File	Internal DSCT	56
	External DSCT	48
	DCB/ACB, if present	256
	DECB/RPL, if present	100

6 If no USERSPA, then only the 500-byte SPA Csect is snapped.

7 If this area is not available (address is zero), the constant THIS AREA IS NOT APPLICABLE TO THIS SNAP will be snapped instead of the control block area. The literal can be easily identified in the EBCDIC representation of the snapped area on the right side of the dump.

Figure 8-1. Areas Displayed by Indicative Dump (Page 2 of 2)

8.4.1 User Snap Exit--SNAPEXIT

A user exit routine is conditionally called by the Csect ICOMSNAP within the module PMISNAP1 to determine whether or not a snap should be taken. This exit routine could be used to prevent buildup of the snap data set by recursive snap calls. The entry point of the exit routine must be SNAPEXIT.

When control is passed to SNAPEXIT, register 1 points to a parameter list, as follows:

1. Address of the one-byte snap-ID in the snap parameter list.

2. Address of the SPIEEXIT save area (See Messages and Codes). This parameter is only meaningful for snap 126.

Upon return from the SNAPEXIT routine, if register 15 contains zero, the snap should be taken. Any other value in register 15 indicates that the snap should be suppressed. Standard linkage conventions apply.

The routine must be closed; that is, it may not relinquish control to the Dispatcher or call any routine which gives up control to the Dispatcher. The routine must be in the same overlay segment as the Csect ICOMSNAP or must be resident.

8.5 SYSTEM DCBs

The member named PMIDCB contains all of the Intercomm system DCBs. The one DCB that is required in all Intercomm systems is labeled PMISNAP, which references the DD statement named SNAPDD. It is used to define output of snap dumps, and may be referenced by any program requiring this facility. (See Figure 8-2).

The QTAMDCB entry illustrated in this figure is required only for the Basic TCAM destination queue DCB as described in the TCAM Support Users Guide.

```
PMIDCB   CSECT
         ENTRY PMISNAP
         ENTRY FASTSNAP
         ENTRY QTAMDCB
         USING *,3
         DC    C'PMIDCB09'
         DS    CL20   PROVIDE BUFFER FOR DCB *-20
*
*
PMISNAP  DCB   DSORG=PS,RECFM=VBA,MACRF=(W),BLKSIZE=882,LRECL=125,    X
               DDNAME=SNAPDD
FASTSNAP DCB   DDNAME=FASTSNAP,DSORG=PS,MACRF=W,RECFM=F
QTAMDCB  DCB   DSORG=PS,RECFM=VBA,MACRF=(W),DDNAME=DUMMY
         END
```

Figure 8-2. Listing of PMIDCB (as released)

8.6 SPINOFF SNAPS

This facility allows the snap data set defined by the SNAPDD DD statement to be renamed dynamically when a user-specified threshold of total output has been reached. The renamed SNAPDD data set may then be printed by a batch program while Intercomm continues to execute. This feature is particularly useful for installations using dynamic program loading. An error condition causing a snap can be analyzed, corrected, and a new version of the program implemented quickly.

The SPINOFF module is called every time a PMISNAP macro is issued, if the DCB parameter is specified as PMISNAP (SEXSNAP in SPA), or omitted. The PMISNAP DCB is defined in the Intercomm member PMIDCB, and specifies the ddname SNAPDD. If SNAPDD is a tape-resident or SYSOUT data set, the SPINOFF facility is meaningless and inoperative, except if SYSOUT spooling is used under MVS, and FREE=CLOSE is specified. If the SNAPDD data set is a disk file, the module checks to see if the total accumulated snap output equals the number of pages specified by the SNAPPGS parameter of the SPALIST macro. If this

threshold has been reached, the snap DCB is closed, the data set is renamed, and the message MP010I is issued giving the new data set name to allow it to be printed. If the threshold is set at fifty pages, for instance, every time fifty pages or more of snaps have been produced (this could be one full snap 126 or snap 118, or many small (indicative dump) snaps), the data set will be spun-off. A new data set is allocated on the same disk pack using the OS/VS allocate SVC and the DCB is then reopened to allow additional snaps.

Under MVS, if the SNAPDD data set is SYSOUT and FREE=CLOSE is specified on the DD statement, the DCB is closed after the SNAPPGS threshold is reached in order to allow immediate printing. Then, a new SYSOUT area is allocated. Also code a space allocation (in cylinders); SPACE=(CYL,20) is recommended. Place the DD statement after the //PMISTOP DD DUMMY statement, as it is not processed by the File Handler.

If allocation of a new SNAPDD data set fails, an informational message is issued and the next snap will be attempted to the file with ddname NEWSNAP. If this fails, a message is issued to inform that future snaps will be lost. If the auxiliary data set (NEWSNAP) is desired, the following DD statement must be added to the execution JCL:

```
//NEWSNAP  DD  SYSOUT=A
```

8.6.1 Implementation

To implement this facility, the following steps must be performed:

1. The module SPINOFF must be included in the Intercomm linkedit.

2. The SNAPPGS parameter must be defined for the SPALIST macro, and then the member INTSPA must be reassembled and linkedited, and a linkedit of Intercomm must be executed.

3. The Intercomm execution JCL must define a disk data set, DISP=(NEW,KEEP), for the SNAPDD DD statement except if SYSOUT under MVS. The space allocation must be large enough to hold a full region dump. If the allocation is too small, an x37 system abend may occur. The SNAPDD data set is referenced by the system DCB labeled PMISNAP in the member PMIDCB. If not SYSOUT, DCB=DSORG=PS must be defined on the SNAPDD statement; other subparameters are already defined on the DCB macro in PMIDCB and may not be changed. For a disk data set, space allocation may be in cylinders or tracks (with primary and secondary extents), and a specific volser may be requested; also a data set name is required. Under MVS, FREE=CLOSE (and a SPACE allocation) must be specified if SYSOUT.

4. The disk pack to which the SNAPDD data set is assigned must have room for subsequent snap data sets to be allocated, once the SPINOFF facility is activated. When a SPINOFF data set is printed or no longer needed, it should be deleted (DISP=(OLD,DELETE)) so as not to waste system resources.

5. Add the NEWSNAP DD statement described above to the Intercomm execution JCL, if desired.

Sample JCL for printing SPINOFF snaps is illustrated in Figure 8-3. This example illustrates the concatenation of two renamed snap data sets produced by SPINOFF.

```
//stepname  EXEC  PGM=IEBGENER,COND=EVEN
//SYSUT1    DD    UNIT=3330,VOL=SER=WORK14,DISP=(OLD,DELETE),
//                DSN=INTERCOM.SLOWSNAP.D83101.T122605
//          DD    UNIT=3330,VOL=SER=WORK14,DISP=(OLD,DELETE),
//                DSN=INTERCOM.SLOWSNAP.D83101.T123819
//SYSPRINT  DD    SYSOUT=A
//SYSUT2    DD    SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=882)
//SYSIN     DD    DUMMY
```

Figure 8-3. Sample JCL for Spinoff Snaps

8.6.2 User SPINOFF Snap Exit--SPINEXIT

SPINOFF conditionally calls a user exit routine which may be coded to determine whether to dispose of the snap data set automatically by generating an internal job to print the data set. This exit may be used to eliminate the need for an external action to print a data set created by SPINOFF. The entry point of the exit routine must be SPINEXIT.

When control is passed to SPINEXIT, register 1 points to a parameter list describing the snap data set just created by the SPINOFF routine, as follows:

1. UCB address

2. Address of the SNAP data set name, a thirty-two-byte character string for slow snaps, 38 bytes for fast snaps.

3. Address of the volume serial number, a six-byte character string.

The exit routine must be serially reusable and may not relinquish control to the Dispatcher, either directly or indirectly. Standard linkage conventions apply. The exit routine must be resident.

8.7 FAST SNAP FACILITY

An optional high-speed Fast Snap facility is available with Intercomm. This is used only to snap the entire Intercomm region. If the issuer of the PMISNAP requests an indicative dump, and indicative dump processing is activated, Fast Snap processing is bypassed. Dramatic improvements in elapsed time (up to 90 percent) have been realized through the use of this facility. The actual improvement depends on the operating system in use and the region size. Intercomm snaps 126, 118 and 114 are issued requesting a Fast Snap. If the facility is not implemented, a normal snap will be taken. Users may request Fast Snaps with a PMISNAP macro. (See Basic System Macros for coding details.) Also code the normal snap parameters, so that a snap will be taken even if any errors occur on the Fast Snap data set.

8.7.1 Restrictions

The implementation of the Fast Snap facility relies upon an operating system capability which is normally for internal use only and is officially supported by IBM only for MVS. This facility may not function properly in some previous (non-MVS) releases of the operating system.

8.7.2 Prerequisites

In order to implement the Fast Snap facility, the following Intercomm components are required:

- Fast Snap SVC defined in SETGLOBE: &FASTSVC SETA svc-number
- Load IGCFAST to SYS1.SVCLIB under SVC number coded above; define as type 2, 3 or 4 (type 3 recommended), not as type 1.
- Reassemble PMISNAP1 and STARTUP3
- Install SPINOFF Snap facility

At execution time, a DD statement is required defining disk space to contain the Fast Snap output. The format of the DD statement is:

```
//FASTSNAP  DD  UNIT=disk,SPACE=(CYL,nn,RLSE,CONTIG)
```

SPACE must be contiguous and large enough to contain the Intercomm Region, Link Pack Area (if applicable) and the operating system nucleus.

8.7.3 Operation

Each Fast Snap taken will allocate and name a new data set on the volume allocated by the system for the FASTSNAP DD statement. A message will be issued providing the data set name and volume serial number. If, for any reason, the Fast Snap operation fails, a message is issued containing a code identifying the reason for the failure. These codes are described in Messages and Codes under the MP011I message.

After a nonzero code, corrective action within SPINOFF may result in the message MP014I or MP013I being issued and then the current snap is processed normally by the Spinoff facility. Allocation will be attempted again on the next Fast Snap. A count of consecutive allocation failures is maintained. When it exceeds three, informational messages are suppressed; however, any successful allocation resets the count.

8.7.4 Printing the Fast Snap--IMDPRDMP

The IBM service aid, IMDPRDMP, may be used to print the data set. The IMDPRDMP service aid may be named differently among operating system versions or releases. JCL required to print the data set is illustrated below.

```
//          EXEC  PGM=IMDPRDMP,PARM='2'
//SYSPRINT  DD    SYSOUT=A                    message data set
//PRINTER   DD    SYSOUT=A                    primary output
//SYSUT1    DD    UNIT=SYSDA,SPACE=(2052,(n,10),,CONTIG) (see NOTE)
//TAPE      DD    DISP=(OLD,DELETE,KEEP),UNIT=disk,VOL=SER=vvvvvv,
//                DSN=INTERCOM.FASTSNAP.Dyyddd.Thhmmss.IDnnn
//SYSIN     DD    *                           IMDPRDMP CONTROL CARDS
     LPAMAP
     QCBTRACE
     PRINT NUCLEUS,JOBNAME=INTERCOMM-jobname
     END
/*
```

where n is the number of blocks calculated as: (core size/2048) + 1, and DSN and VOL are those described by message MP010I at execution time when the Fast Snap data set was created.

NOTE: Consult Service Aids manual relevant to your operating system for appropriate SYSUT1 block sizes and other requirements.

8.8 SYSTEM ACCOUNTING AND MEASUREMENT (SAM) FACILITY

The optional Intercomm System Accounting and Measurement (SAM) facility is used to accumulate resource usage information for each message processed by subsystems operating under Intercomm. The captured resource usage information can be used, for example:

- For report generation purposes to allocate charges for use of a resource
- To fine tune the Intercomm System

The information from SAM is written to INTERLOG in conjunction with the logging of the X'FA' completion record at the time subsystem processing completes for each message. An off-line utility program is provided to extract the data from the sorted log and print it. If LOG=NO is coded on a SYCTTBL macro, SAM information will be unavailable for that subsystem. If message restart is not applicable for the subsystem, RESTART=NO should be coded on the associated SYCTTBL macro.

8.8.1 Specifying System Resource Usage Categories

The SAM facility is capable of capturing information on up to fifty-three system categories, as specified via the MAPACCT macro. The MAPACCT macro is used to specify the following:

- The name (keywords) of the system resource usage categories to be collected
- The grouping of certain categories for reporting purposes
- The title to be used in the report to describe each group

The MAPACCT macro is coded as follows:

```
MAPACCT ('b1',r,r...),('b2',r,r...),...('bn',r,r,...)
```

Only one MAPACCT macro is coded; all statistics categories to be accumulated systemwide for each processing thread must be specified on that macro. Each group of parameters within parentheses defines a single accounting group or "bucket." The value coded for "bn" must be a character string of one to ten characters and represent the title to be used for that bucket in the final report. Each "r" parameter must be a SAM keyword representing a category of resource usage to be included in that bucket. Any number of buckets may be specified, but no resource usage category may appear in more than one bucket. The system resource usage categories and their keywords are listed in Figure 8-4. It should be noted that no count of WAIT time is kept.

Keyword	Resource--Usage Type
CPUTIME	Total thread CPU time in units of 1/1000 second
HIGHSTOR	Thread high water mark of core usage. If specified, the STORAGES keyword must also be specified.
STORAGES	Total number of storage requests
MESSAGES	Total number of messages generated by the thread
PFAULTS	Total number of page faults (VS only)
OLOADS	Total overlay loads through use of CALLOVLY
LOADS	Total module loads via the PMIDLOAD module
ENQS	Total ENQS through use of the INTENQ macro (routine)
OPENS	Total of File OPENs
CLOSES	Total of File CLOSEs
SETLS	Total QISAM SETLs
QISAMG	Total QISAM GETs
QISAMP	Total QISAM PUTs
BISAMR	Total BISAM READs
BISAMW	Total BISAM WRITE Updates
BISAMWKN	Total BISAM WRITE Adds
BDAMR	Total BDAM READs
BDAMW	Total BDAM WRITEs
BSAMR	Total BSAM READs
BSAMW	Total BSAM WRITEs
QSAMG	Total QSAM GETs
QSAMP	Total QSAM PUTs
VSAMG	Total VSAM GETs

Figure 8-4. Resource Usage Categories (Page 1 of 3)

Keyword	Resource--Usage Type
VSAMP	Total VSAM PUTs
VSAMPT	Total VSAM POINTs
VSAME	Total VSAM ERASEs
SELECTS	Total File SELECTs
RELEASES	Total File RELEASEs
ALLOCS	Total calls to ALLOCATE
ACCESSES	Total calls to ACCESS
FETCORE	Total FETCHs from core
FETDISK	Total FETCHs from disk
STORCORE	Total STOREs to core
STORDISK	Total STOREs to disk
STORUPD	Total STORE UPDATES with length change
UNSTCORE	Total UNSTOREs of transient strings (core and disk)
UNSTDISK	Total UNSTOREs from disk (semipermanent and permanent strings)
MAPINS	Total calls to MAPIN
MAPOTS	Total calls to MAPOUT
MAPENS	Total calls to MAPEND
MAPPRS	Total calls to MAPURGE
MAPCLS	Total calls to MAPCLR
MAPFRS	Total calls to MAPFREE
MPPAGES	Total pages created via MMU
QBLDS	Total number of DDQ QBUILDs
QOPNS	Total number of DDQ QOPENs
QRDS	Total number of DDQ QREADs

Figure 8-4. Resource Usage Categories (Page 2 of 3)

Keyword	Resource--Usage Type
QRDXS	Total number of DDQ QREADXs (for update)
QWRS	Total number of DDQ QWRITEs
QWRXS	Total number of DDQ QWRITEXs (for update)
QCLSS	Total number of DDQ QCLOSEs
FESCLS	Total calls to FESEND
FEOTPUT	Total calls to PMIOTPUT (entry in FESEND)

Figure 8-4. Resource Usage Categories (Page 3 of 3)

Any resource usage types not referred to by keywords in the MAPACCT macro are not considered for statistics. All specified categories are collected for all active subsystems.

Resource usage accumulations can be dynamically stopped or started while Intercomm is processing, via the STOP and STRT system control commands.

8.8.2 Specifying User Accumulators

In addition to the fifty-three system-defined resource usage accumulators represented by the keywords in Figure 8-4, up to ten user accumulators can be specified. The MAPACCT macro is also used to specify the user accumulators; coding conventions for system resource usage categories also apply to user-specified accumulators. The user accumulator keywords must have the following format:

USRBKnn

where nn is coded in the range of 01 to 10, inclusive.

The USRTRACK macro with the BUCKTNO parameter is issued by the user, when appropriate, to increment by one a user accumulator defined via the MAPACCT macro. (See Basic System Macros.)

8.8.3 SAM User Exit Routines--USRSAMnn

Optionally, up to ten user-coded exit routines are permitted with the SAM facility for use with USRBKnn accumulators. A SAM user exit routine is specified to the SAM facility as a keyword on the MAPACCT macro, as follows:

USRFNnn

where nn is coded in the range of 01 to 10, inclusive, which corresponds to a user accumulator USRBKnn. The user exit is invoked via the FUNCNO parameter of the user-coded USRTRACK macro. (See Basic System Macros.)

The user-written exit routines are coded with a Csect name as follows:

USRSAMnn

where nn corresponds to the value specified in the USRFNnn keyword.

The following are conventions for user-written exit routines:

- Can be resident modules or reside in the uncontrolled overlay region
- Must follow standard linkage conventions
- Are passed the address in register 1 of the bucket (accumulator) with which the corresponding USRFNnn has been associated (USRBKnn)
- May not give up control to the Dispatcher, whether directly or indirectly.

8.8.4 Implementation

The MAPACCT macro is coded in a member named SAMTABLE with a Csect name of SAMTABLE. This member, along with the processing modules SAMSECT and TRACKMOD must be included in the Intercomm linkedit. The INCLUDE cards for these SAM modules are automatically produced if SAM=YES is specified on the ICOMLINK macro. A typical SAMTABLE is illustrated below:

```
//       EXEC  LIBELINK,Q=LIB,NAME=SAMTABLE,LMOD=SAMTABLE
./       ADD   NAME=SAMTABLE,LIST=ALL
SAMTABLE CSECT
         MAPACCT ('BDAM READS',BDAMR),                              X
               ('CPUTIME',CPUTIME),                                 X
               ('ALL WRITES',BISAMW,BISAMWKN,BDAMW,BSAMW),          X
               ('HIGH CORE',HIGHSTOR),                              X
               ('STORAGES',STORAGES),                               X
               ('MY BUCKET',USRBK01),                               X
               ('MY ROUTINE',USRFN01)
         END
/*
```

The USRTRACK macro may be coded in a user-written Assembler Language subroutine called by a high-level language subsubsystem (COBOL, PL/1) or issued directly in an Assembler Language subsystem. When the BUCKTNO=nn parameter is specified, the corresponding bucket (in the SAMTABLE) to which the keyword USRBKnn was assigned (via the MAPACCT macro) is incremented by one by Intercomm. However, if the FUNCNO=nn parameter is used, the corresponding USRSAMnn user exit subroutine is invoked, with the address of the corresponding bucket (in the SAMTABLE) to which the keyword USRFNnn was assigned (via the MAPACCT macro). The user exit may examine the contents of the bucket (a fullword) and increment or decrement it by any desired value. Also, the user exit could indicate, via a return code to the user's calling routine, the results of examination/manipulation of the bucket. Thus, the next processing step to be taken within the user routine issuing the USRTRACK macro can depend on that return code, if desired. For example, processing action may be different, depending on whether the bucket is or becomes zero or not.

8.8.5 Reports from System Accounting and Measurement

Two main types of reports may be produced from the data captured on the log. The major control can be on the subsystem codes or the terminal-IDs. Totals for accumulated data will be printed either by subsystem or by terminal. For each of these two main report types the user may also request that detail information be printed as well as totals. If detail information is requested, the resource usage information will be printed for every individual message in addition to the totals.

Before executing the report program, INTERLOG must be sorted to produce the input to the report program; DCB=(RECFM=VB,BLKSIZE=nnnnn, LRECL=nnnnn-4,DSORG=PS) must be specified. The SORTOUT data set should be defined as a variable-length blocked file (maximum LRECL is 42 plus the number of accounting buckets (times 4) rounded up to the next doubleword). A sort E15-exit must be used to delete extraneous records from the sort. This exit routine is named SAME15 and is supplied on MODREL. The control cards for the sort to produce a sorted output file to be used for a report whose major control is on the subsystem codes, are as follows:

```
SORT FIELDS=(29,1,BI,A,5,1,BI,A,24,5,CH,A,7,3,BI,A),SIZE=E9000
MODS E15=(SAME15,500,MODREL,N)
```

The following control statements will produce a sorted output file with a major control on terminal-IDs:

```
SORT FIELDS=(24,5,CH,A,29,1,BI,A,5,1,BI,A,7,3,BI,A),SIZE=E9000
MODS E15=(SAME15,500,MODREL,N)
```

In both of the above cases, a DD statement named MODREL must define the Intercomm MODREL library in the sort JCL.

The report program must be linkedited as follows:

```
INCLUDE SYSLIB(SAMREPT,SAMRPTIO,SAMTABLE)
ENTRY SAMREPT
```

(The SAMTABLE member is the same member used in the Intercomm linkedit.)

The report is produced using the following JCL:

```
//          EXEC  PGM=SAMREPT,PARM=pppp
//STEPLIB   DD    DISP=SHR,DSN=library-with-SAMREPT
//SYSUDUMP  DD    SYSOUT=A
//SAMPRNT   DD    SYSOUT=A,DCB=(RECFM=FA,LRECL=133)
//SAMFILE   DD    (Output from the Sort)
```

The PARM field controls the type of report to be produced. The PARM values are detailed in Figure 8-5.

Parm	Value	Type of Report	Notes
PARM=	'SUBO'	Totals by subsystem code	1,3
	'SUBO,DTL'	As above with detail information	1,3
	'SUBT'	Totals by terminal within subsystem code	1
	'SUBT,DTL'	As above with detail information	1
	'TRMO'	Totals by terminal	2,3
	'TRMO,DTL'	As above with detail information	2,3
	'TRMS'	Totals by subsystem within terminal	2
	'TRMS,DTL'	As above with detail information	2

1: File was sorted with major control on subsystem code.

2: File was sorted with major control on terminal.

3: The last character of SUBO or TRMO is the letter 'O', not zero.

Figure 8-5. SAM Report Execution PARM Values

Sample output from a System Accounting and Measurement Report is illustrated in Figure 8-6, and illustrates statistics for multiple terminals accessing subsystem 'C7D7', via a PARM of 'SUBT,DTL'.

```
* DTL  SUBSYSTEM 00D9  TERMINAL ID PAUL1  MSN= 00000147  TIME= 18343487

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0           30              1624              72               1               0               1               0
   BISAMR          BISAMW          BDAMW
            0               0               0

* DTL  SUBSYSTEM 00D9  TERMINAL ID PAUL1  MSN= 00000247  TIME= 18350256

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0            0              1512               6               1               0               0               0
   BISAMR          BISAMW          BDAMW
            0               0               0

* DTL  SUBSYSTEM 00D9  TERMINAL ID PAUL1  MSN= 00000521  TIME= 18370603

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0           21              1624              12               1               0               1               0
   BISAMR          BISAMW          BDAMW
            0               0               0

* DTL  SUBSYSTEM 00D9  TERMINAL ID PAUL1  MSN= 00000718  TIME= 18391546

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0          117              9856             251               5               0               1               0
   BISAMR          BISAMW          BDAMW
            0               0               0

* DTL  SUBSYSTEM 00D9  TERMINAL ID PAUL1  MSN= 00000826  TIME= 18403044

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0          174              2592             234               1               0               1               0
   BISAMR          BISAMW          BDAMW
            0               0               0

  ** TOTALS     SUBSYSTEM    00D9    TERMINAL ID PAUL1

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0          342             17208             575               9               0               4               0
   BISAMR          BISAMW          BDAMW
            0               0               0



    **** TOTALS  SUBSYSTEM    00D9

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0          342             17208             575               9               0               4               0
   BISAMR          BISAMW          BDAMW
            0               0               0

    **** AVERAGES  SUBSYSTEM    00D9          MESSAGE COUNT=  00005

   BDAM READS      CPUTIME         HIGH CORE       STORAGES        MESSAGES        ENQUEUES        OPENS           ALL QISAM
            0           69              3442             115               2               0               1               0
   BISAMR          BISAMW          BDAMW
            0               0               0
```

Figure 8-6. System Accounting and Measurement Report Sample

8.9 SYSTEM TUNING STATISTICS

The System Tuning Statistics facility, using minimal overhead, is optionally available to users of Intercomm. The statistics are accumulated and written to a statistics data set at time intervals specified by the SPALIST macro, STSTIME parameter. The information obtained can be used to tune and optimize the Intercomm system. (System Tuning is also described in Chapter 11.)

8.9.1 Reports from System Tuning Statistics

System Tuning Statistics are accumulated in a report issued at user-specified intervals, and at closedown (also at abend - if STAEEXIT in Intercomm linkedit). The report includes statistics on:

- Message processing and overflow disk queuing
- INTERLOG log records processing
- Dynamic/Overlay subsystem/subroutine loading
- Store/Fetch activity

Figure 8-7 illustrates a sample report produced by System Tuning Statistics routines. Each printed report displays cumulative totals. Hence, reports produced over a given time span can be used independently, or with the SAM Facility or Log Analysis on a comparative basis to determine bottlenecks, activity cycles and tuning possibilities (see Chapter 11).

8.9.2 Implementation

Implementation of the System Tuning Statistics facility requires the following:

1. The member INTSTS must be included in the linkedit of Intercomm (automatic if ICOMLINK used to generate Intercomm linkedit). INTSTS consists of two Csects--INTSTS and INTSTSPR. INTSTS must be resident, while INTSTSPR may be placed in a transient overlay area.

2. The STSTIME parameter in the SPALIST macro must be set to the time interval (in seconds) for which the System Tuning Statistics are to be printed. If this is not set, the default value of 120 seconds is used.

```
NUMBER OF BACK END MESSAGES PROCESSED = 1,417
NUMBER OF FRONT END MESSAGES PROCESSED = 639
NUMBER OF BACK END BLOCKS WRITTEN TO DISK-QUEUES = 15
NUMBER OF FRONT END BLOCKS WRITTEN TO DISK-QUEUES = 475
NUMBER OF MESSAGES PASSED TO SATELLITE-REGIONS = 0
TOTAL NUMBER OF PHYSICAL RECORDS WRITTEN TO INTERLOG = 1,578
TOTAL NUMBER OF SYNCHRONOUS LOGICAL-RECORDS WRITTEN TO INTERLOG = 287
TOTAL NUMBER OF ASYNCHRONOUS LOGICAL-RECORDS WRITTEN TO INTERLOG = 5,676
TOTAL NUMBER OF LOGICAL-RECORDS (BOTH TYPES) WRITTEN = 5,963
TOTAL NUMBER OF BYTES WRITTEN TO INTERLOG = 533,764
AVERAGE NUMBER OF BYTES PER PHYSICAL RECORD = 338
NUMBER OF BUFFER-WAIT CONDITIONS = 3
PERCENTAGE OF BUFFER-WAITS TO BUFFERS WRITTEN = 0
NUMBER OF OVERLAY-SUBSYSTEM SEGLDS = 150
NUMBER OF NON-SUBSYSTEM SEGLDS = 291
NUMBER OF SUBSYSTEM DYNAMIC-LOADS = 20
NUMBER OF SUBROUTINE DYNAMIC LOADS = 1
NUMBER OF STORE/FETCH RECORDS THAT SPANNED BLOCKS = 22
NUMBER OF STORE/FETCH FLUSHES = 35
MAXIMUM AMOUNT OF STORAGE USED FOR IN-CORE STRINGS = 4,808
AVERAGE STORE/FETCH STRINGS SIZE = 237
AVERAGE NUMBER OF DISK BLOCKS SEARCHED(S/F) = 3
```

Figure 8-7. Sample Report from System Tuning Statistics

3. The data set for the statistics reports must be a sequential output data set. SYSOUT may be used for this purpose. A DD statement must be included for the data set with the following specifications:

   - STSLOG must be the ddname.
   - DCB information on the DD statement should be as follows:

     DCB=(DSORG=PS,LRECL=120,BLKSIZE=multiple-of-120,RECFM=FBA)

8.10 LOG INPUT FACILITY

The Log Input Facility (LOGINPUT) allows an Intercomm system log (INTERLOG) created in a previous execution of Intercomm to be used as input to a subsequent execution. LOGINPUT reads the sequential data set (ddname LOGINPUT), extracting all messages queued by the Front End for a subsystem: 01 log records (and C1 log records if Multiregion control region). (Messages queued for the closedown and checkpoint subsystems are ignored.) Input messages are then queued for the appropriate user subsystem as if they had come in from the Front End during this execution of Intercomm. The time interval between executions of LOGINPUT to search for the next message to input to the system is specified by the &LOGINTM or &LGINRTD globals in SETGLOBE.

Normally, any terminal output generated by LOGINPUT is sent in the usual manner. However, the terminal output may be optionally discarded by appropriate SPALIST and SETGLOBE specifications.

The proper function of this facility necessitates that all application subsystems place the sending subsystem code in all messages queued for another subsystem. Otherwise, messages may be found on the LOGINPUT data set (and reprocessed) that were not messages originally input to the Front End (characterized by sending subsystem code of binary zeros).

In the case of errors of a noncritical nature, namely inability to queue a message due to invalid subsystem code, no room on queue, etc., the message will be bypassed. For errors of a critical nature, such as a no storage condition, I/O errors on LOGINPUT, etc., a message will be issued and the Log Input Facility terminated.

When the LOGINPUT data set is completely processed, the facility issues a message and terminates itself.

The Log Input facility is implemented in the following manner:

1. Update SETGLOBE to use one of the two globals &LGINRTD and &LOGINTM in order to control the time interval between input messages from the LOGINPUT data set. &LGINRTD specifies a real time divisor, that is, the actual time interval between input messages (calculated from the log) is divided by &LGINRTD to compute the interval between LOGINPUT's generated messages. If this method is desired, specify:

   ```
   &LGINRTD SETA n
   ```

   If &LGINRTD is set to zero (0), then &LOGINTM is used. This specifies a constant time interval in tenths of a second. To request this method of interval calculation, code:

   ```
   &LOGINTM SETA n
   ```

2. To discard the terminal output, specify on the SPALIST macro:

   ```
   LOGINDO=YES.
   ```

   This indicates that the message output should be discarded (not queued for the Front End). In this case, the output message will be logged with a log code of X'40', as though it were Test Mode output. LOGINPUT substitutes the dummy terminal-ID coded for the SETGLOBE global &GENTERM for the actual terminal name in the requeued input.

3. Reassemble and linkedit INTSPA and LOGINPUT: ensure the revised SETGLOBE is in the first library in the STEPLIB concatenation stream.

4. Include LOGINPUT as resident in the Intercomm linkedit.

5. Define Back End Station and Device Table entries for the dummy terminal name defined by &GENTERM in SETGLOBE (default is $$$$$); device type should be that of the majority of the input terminals.

6. Execution JCL must contain a DD statement for LOGINPUT, as follows:

```
//LOGINPUT DD DSN=INTERLOG-name,
//         DCB=(DSORG=PS,RECFM=VB,BLKSIZE=mmmm,LRECL=blksize-4),
//         additional operands as required
```

7. Execution may be in Test Mode, startup or restart. To execute in Test Mode, at least one input message must be coded for the SYSIN data set (see Figure 8-9).

8.11 TEST MODE OPERATION

Intercomm allows the complete testing of application programs without using terminals at all. The input messages are read in at startup time from a sequential data set with a ddname of SYSIN. All messages are snapped onto a sequential output data set with a ddname of SYSSNAP after they have been read from SYSIN.

All output from Intercomm is similarly snapped onto a sequential output data set with ddname of SYSSNAP2, rather than being passed to the Front End. The snaps issued have a snap-ID to identify them as follows:

- 15--Snap of a complete input message
- 20--Snap of a complete output message

The Test Monitor (PMITEST) effectively replaces the Intercomm Front End. The Test Mode input card MSG contains all the message header fields normally supplied by Front End Table information. In particular, the Receiving Subsystem Codes (MSGHRSCH,MSGHRSC), Terminal Identification (MSGHTID), and Verb/Message Identifier (MSGHVMI) are critical for proper message routing.

All input messages are read as 80-byte logical records. Each message is preceded by a header record defining the start of the message and is terminated by a trailer record defining the end of the message. Detail lines of the input message are read in as separate 80-byte records between the header and trailer records for the message. The format of the various input records is illustrated in Figure 8-8.

Card	Columns	Contents
HEADER	1-3	MSG
	*6-8	Lo-order byte of S/S code (MSGHRSC) (or 8)
	*9-11	Hi-order byte of S/S code (MSGHRSCH) (or 11)
	20-24	Sending terminal-ID (MSGHTID)
	50-53	Front End Serial Number (MSGHBMN)--leading zeros
	*55-57	VMI value (MSGHVMI): leave blank if editing required by the Edit Utility; code 255 if using MMU or if no editing is desired (or 57)
*three-digit integer values from 000 to 255 or a corresponding single alphanumeric character in the column indicated in parentheses (8,11,57).		
DETAIL(s)	1-64	Data for one line of input message. If VMI in header card is left blank, a New Line character is inserted at the end of text on every card, except the last one. If the last nonblank character is a $ sign (X'5B'), it will be replaced by a NL; the preceding character (usually a blank) is kept as part of the input. All NLs are suppressed if editing is not required. If editing is required, the system separator character used between positional fields must be the same as that coded for the SPALIST macro, SEP parameter.
TRAILER	1-3	Generates End-of-Transmission character following the last nonblank character of the previous detail card.

Contents of Card	Ending Character
EMS	EOT (X'37')
EOT	EOT (X'37')
ETX	ETX (X'03')
ETB	ETB (X'26')

Figure 8-8. Test Mode Input Card Formats

NOTE: The user may define new ending characters by inserting DC instructions in PMITEST, as follows:

```
./ NUMBER INSERT=YES,SEQ1=67050,NEW1=67051,INCR=1
          DC   CL4'eee',X'nn'
```

where:

eee is the three-character trailer card value and nn is the hexadecimal code equivalent of the named ending character. Note that a blank will be generated between the eee and nn values at assembly time.

The maximum total message size and the maximum number of text columns per detail card are determined by the global variables &MAXMSG and &MAXCRD, which default to 1000 and 64, respectively. The user may change these values (for example, &MAXMSG SETA 2000 would increase the maximum message size to 2000) by insertion of the appropriate SETA instructions in the PMITEST module at sequence numbers 00002030 and 00002040.

The system log INTERLOG is maintained during Test Mode execution, as in live mode, and provides further information for analyzing the results of Test Mode operation. Output messages passed to FESEND (by the subsystem, the Output Utility, or MMU) are logged with a log code of X'40' so they may be examined for valid data.

After all Test Mode messages have completed processing, the method of step termination depends on the value of the TSTEND operand of the SPALIST macro. The default is TSTEND=NRCD; proceed with normal system closedown with no dump. Other options available are: TSTEND=NODUMP, which causes abend 999 without a dump; and TSTEND=DUMP, which causes an abend 999 with a dump. The Test Mode closedown logic is in PMITEST. It remains the responsibility of the user to determine whether or not the messages were processed successfully by examining SYSSNAP2 and INTERLOG records.

One or more Test Mode jobs may execute concurrently (with or without concurrent execution of a live system), as long as there is no conflict with respect to OS allocation and disposition of data sets, or the dynamic load library. Figure 8-9 illustrates typical Test Mode JCL, including a step to print the system log via the Intercomm LOGPRINT utility (see Chapter 12). User data set DD statements must be inserted before the //PMISTOP DD statement.

The Intercomm linkedit for Test Mode may be generated via the ICOMLINK macro (see Basic System Macros); code TEST=YES and all other parameters applicable to the on-line system (except Front End, security and Multiregion parameters). The Link Pack Facility may be used with a Test Mode system.

```
//EXECTEST  EXEC     PGM=INTCOMM,PARM='TEST',REGION=500K
//STEPLIB   DD       DSN=INT.MODUSR,DISP=SHR
//          DD       DSN=INT.MODLIB,DISP=SHR
//          DD       DSN=INT.MODREL,DISP=SHR
//SMLOG     DD       SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=120,RECFM=FA)
//STSLOG    DD       SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=120,RECFM=FA)
//SYSPRINT  DD       SYSOUT=A,
//                   DCB=(DSORG=PS,RECFM=VA,BLKSIZE=141,LRECL=137)
//INTERLOG  DD       DSN=&&INTLOG,VOL=REF=INT.SYMREL,
//                   DISP=(,PASS),SPACE=(TRK,(2,2)),
//                   DCB=(DSORG=PS,RECFM=VB,BLKSIZE=3200,LRECL=3196,
//                   NCP=2,OPTCD=C)
//PMIQUE    DD       DSN=INT.PMIQUE,DISP=OLD,DCB=(DSORG=DA,OPTCD=RF)
//RCT000    DD       DSN=INT.RCT000,DISP=OLD,DCB=(DSORG=DA,OPTCD=RF)
//INTSTOR0  DD       DSN=INT.INTSTOR0,DISP=OLD,
//                   DCB=(DSORG=DA,OPTCD=EF,LIMCT=3)
//INTSTOR2  DD       DSN=INT.INTSTOR2,DISP=OLD,
//                   DCB=(DSORG=DA,OPTCD=EF,LIMCT=3)
//INTSTOR3  DD       DSN=INT.INTSTOR3,DISP=OLD,
//                   DCB=(DSORG=DA,OPTCD=EF,LIMCT=3)
//SYSIN     DD       *
MSG    W000        CNT01                          0001 255
SNBK,     SEND THIS MESSAGE BACK TO SENDER
EMS
MSG    B000        CRT01                          0002 255
SWCH,(NYC01)      SEND THIS MESSAGE TO ONE OTHER TERMINAL
EMS
MSG    B000        NYC01                          0003 255
SWCH,(CRT01,CNT01,NYC01)   SEND MESSAGE THREE TO THREE TERMINALS
EMS
//PMISTOP   DD       DUMMY              DELIMITS FILE HANDLER ACCESS
//SNAPDD    DD       SYSOUT=A           STANDARD SNAPS
//SYSSNAP   DD       SYSOUT=A           TEST MODE ONLY INPUT ID=015
//SYSSNAP2  DD       SYSOUT=A           TEST MODE ONLY OUTPUT ID=020
//DYNLLIB   DD       DSN=INT.MODUSR,DISP=SHR
//DYNLWORK  DD       UNIT=SYSDA,DISP=(,PASS),SPACE=(CYL,(1,1))
//DYNLPRNT  DD       SYSOUT=A
//*
//PRINTLOG  EXEC   PGM=LOGPRINT,COND=EVEN
//STEPLIB   DD     DSN=INT.MODREL,DISP=SHR
//INTERLOG  DD     DSN=&&INTLOG,DISP=OLD,DCB=BLKSIZE=3200
//*
//*          NOTE THAT INTERLOG BLOCK SIZE MAY BE ANY VALUE THAT
//*          EQUALS OR EXCEEDS THE MAXIMUM DATA SET BLOCK SIZE.
//*
//SYSPRINT  DD     SYSOUT=A,DCB=(DSORG=PS,BLKSIZE=121)
//
```

Figure 8-9. Sample Test Mode JCL

Chapter 9

LOGGING, SYSTEM RESTART, MESSAGE RECOVERY

9.1 INTRODUCTION

Intercomm provides message restart as a standard option; file recovery is a special feature (see the File Recovery Users Guide). This chapter describes only message restart without file or data base recovery considerations. It documents the following subjects:

- System failure and recovery
- Message restart concepts
- System logging
- System checkpoints
- Restart/recovery
- Implementation
- Serial restart

9.2 SYSTEM FAILURE AND RECOVERY

Intercomm is designed to anticipate, detect and recover from most error situations without bringing down the entire teleprocessing system. In most instances following failures, Intercomm can continue to run in a degraded mode without the failing components. Alternatively, Intercomm can come down gracefully after failure by completing all work that is in process at time of failure. Certain conditions, however, may occur that cause immediate termination of all processing in Intercomm; for example, power failures, machine failures, data base destruction or operating system failure. In these and other total failure cases, Intercomm automatically provides for the complete recovery of the teleprocessing environment. This recovery includes the restarting of all messages in progress at the time of failure, the recovery of message queues and the coordinated recovery of files and data bases, the last being a special feature.

Recovery from failure situations is based upon the system log, a sequential data set (INTERLOG) providing a historical record of all message processing, and a checkpoint file, a BDAM data set providing a record of critical tables.

Message recovery in Intercomm is a rapid warm restart. No off-line program need be run following a failure; live Intercomm is merely restarted. Intercomm first reads the INTERLOG file backwards from the point of failure as far back as necessary for recovery. With the necessary items recovered, live Intercomm starts up and all messages and queues are restarted and recovered as appropriate.

In certain cases, previously completed subsystem messages must be restarted to effect necessary interaction with files or data bases to ensure data integrity. Duplicate output to terminals may be controlled by subsystem processing and/or table definitions. (For a description of INTERLOG entries, see Figure 9-1.)

9.3 MESSAGE RESTART CONCEPTS

9.3.1 Mandatory and Desirable Conditions

Users can specify the extent of restart on a terminal-by-terminal and program-by-program basis. In addition to the restart "always" or "never" specifications, a third condition is a restart "desirable" condition. If restart is desirable for a particular program or terminal, then, in the course of restart for mandatory programs and terminals, restart will also be performed for desirable components. However, once restart has completed for all mandatory items, the restart is deemed complete, regardless of whether desirable components have been fully restarted. Thus, restart will proceed as far backwards through the log file as is necessary to retrieve all messages for mandatory restarted components. During this read back process, messages encountered for desirable restart components will be retrieved as appropriate.

For those terminals or subsystems where restart is neither desirable nor mandatory, the logging itself becomes a user option, again on an individual program and terminal basis. This selective logging can reduce system overhead in logging that would otherwise be wasted on noncritical components such as inquiry programs.

9.3.2 User Responsibility in Restart

The mechanics of implementing message recovery are supported by Intercomm-supplied software. User application programs that are to be restarted are no different in format or content than non-restarted programs. The restart facility is transparent to the application programmer. User responsibility in restart is limited to table entries (RESTART parameter on SYCTTBL and BTERM/LUNIT macros) that delineate those programs and terminals for which restart is to be performed. Other table entries specify those programs that may update data bases. These table entries are generally the extent of user responsibility in restart provisions.

9.4 SYSTEM LOGGING

The logging facility of Intercomm maintains a sequential data set of all traffic within the system and provides control and documentation of system performance in several areas:

- Message traffic is recorded at the time of entry on a subsystem queue and at the time message processing begins and ends within each subsystem.
- Output message traffic is recorded by the Intercomm Front End as it arrives from the Back End, and as message transmission is completed.
- Input message traffic is logged only if the supplied version of USRBTLOG is assembled and linked with the Intercomm load module (see BTAM Terminal Support Guide).

Complete documentation of the progress of a message through the system is thus provided.

The Intercomm log facility utilizes a single data set whose ddname is INTERLOG. On this log data set will be entries reflecting the status of messages queued for subsystems and terminals, and of before- and after-images of file data for selected files being updated. Also included on INTERLOG are user log entries, checkpoint time records, message accounting records and Multiregion interface queuing records.

INTERLOG may be specified on its DD statement as residing on tape or disk. If the DD statement for INTERLOG is omitted, then no logging will be performed and no restart is possible. The computer operator is notified at startup time if INTERLOG is not specified by a DD statement or the file cannot be opened. If restart is intended, the computer operator's response to this message should be to cancel Intercomm and request programming assistance before proceeding.

Blocked INTERLOG records appear on the file as standard OS undefined record blocks and can be read using QSAM without deblocking (if RECFM=VB is specified). A special technique is utilized in creating this log; BSAM rather than QSAM is employed, and average length buffers (specified by the SPALIST parameter LGBLK) are used. This means that, where possible, log records are blocked to form a buffer whose size approximates LGBLK. However, a record will be logged even if its size exceeds LGBLK. Thus any size record (up to the BLKSIZE (-4) specified on the DCB parameter of the JCL for INTERLOG) can be logged. BLKSIZE must be a multiple of 4.

Should the user desire to use IEBGENER to copy INTERLOG, it is important to specify RECFM=U, not RECFM=VB. RECFM=U will prevent fields in the Intercomm message header from being overlaid by QSAM or BSAM when copying. Each block starts with a Block Descriptor Word providing the block length. IEBGENER may not be used to reblock INTERLOG data sets.

Certain log records may be written immediately. For example, log entries for an important subsystem are made immediately by adding the entry to the current buffer and then writing out that buffer. Users specify critical subsystems by the SYCTTBL parameter LSYNCH=YES and critical terminals by the BTERM/LUNIT parameter LSYNCH=YES.

The user has the option of entirely suppressing log entries for a particular terminal or subsystem via the LOG parameter of the BTERM/LUNIT and SYCTTBL macros. This applies to both Intercomm and user log entries.

Log entries are identified by a code in the MSGHLOG field of the message header. The time and date stamps (MSGHTIM and MSGHDAT) in the message header are updated for each log entry. Log entries take two formats:

- 42-byte message header and full text when the message arrives from a terminal or is queued for a subsystem, terminal or satellite region
- Header-only entries to mark progress through the system or error conditions.

A Log Analysis utility, supplied with Intercomm, may be used off-line to produce a report of message queuing and processing time. Statistics for messages by terminal, verb, subsystem, and system totals are provided. (See Chapter 12.)

Entries are made on the log by calling the service routine LOGPUT. LOGPUT can be called from any application program with one parameter, the address of the message to be logged. If the length field in the message header contains a number lower than 42, LOGPUT will cause an 0C2 program check, resulting in a snap 126 and termination of the current message processing thread.

LOGPUT uses a translate table to effect internal to external translation of the log code via the table LOGTROUT. Any code that is translated into X'FF' in this table will not be written to the log. This feature can be used to eliminate user log entries without modifying the application program.

Figure 9-1 illustrates INTERLOG entries and their use at restart time. User log entries are ignored at restart time; hence they should be utilized only for user accounting and performance analysis purposes.

Internal Code	External Code	Format	Description	Origin	Restart Use
X'00'	00	HT	Checkpoint Record	Checkpoint	Yes
C'2'	01	HT	Message queued for subsystem by Front End or a subsystem	Message Collection	User
C'R'	02	HT	Message restarted through the system	LOGPROC	User
C'P'	03	HT	Message restarted--related to Data Base Recovery	LOGPROC	User
C'T'	30	HO	Message passed to subsystem for processing	Subsystem Controller	User
C'Z'	40	HT	Message passed to Front End (test mode only)	FESEND	No
X'41'-X'6F'	41-6F	HT	User called LOGPUT	Any Subsystem	No
X'80'-X'8E'	80-8E	HT	File Recovery before-images	IXFLOG	User
X'8F	8F	HO	Checkpoint Records indicator	IXFCHKPT	Yes
X'90'-X'9E'	90-9E	HT	File Recovery after-images	IXFLOG	User
X'9F'	9F	HT	Intercomm Startup	LOGPUT	Yes
X'A0'	A0	HO	Message restart begun	LOGPROC	Yes
X'A1'	A1	HO	Message restart finished: all subsequent log entries produced by live Intercomm	LOGPROC	Yes
X'AA'	AA	HT	Intercomm Closedown	LOGPUT	No
X'C0'	C0	HT	Region started (Multiregion only) (Text=Region-id(s))	MRINTER	No
C'A'	C1	HT	Message successfully queued for Satellite Region	MRQMNGR CR only	User

Internal Code: Log code in core during processing (snaps and dumps)
External Code: Log code after translation by LOGPUT (INTERLOG printout)
Format: HT for header and text, HO for header only
Restart Use: Yes, No, User (specified via user-coded system macros)

Figure 9-1. INTERLOG Entries (Page 1 of 2)

Internal Code	External Code	Format	Description	Origin	Restart Use
C'B'	C2	HO	Message successfully passed to Satellite Region	MRQMNGR CR only	User
C'C'	C3	HO	Message lost (Region/Hold Q full) or flushed (SR/SS down)	MRQMNGR CR only	User
C'I'	C9	HT	Sign on/off processing, security violation messages	ESS	No
C'3'	FA	HO	Normal message complete	Subsystem Controller	User
C'5'	FB	HO	Unprocessed message--invalid subsystem/QPR code	Message Collection	User
C'6'	FC	HO	Unprocessed message--core and disk queue full	Message Collection	User
C'8'	FD	HO	Message cancelled--program error or time-out, I/O error	Subsystem Controller	User
C'9'	FE	HO	Message flushed by Retriever, used when application program does not obtain (via GETSEG) all parts of a segmented message; or message failed security check	Retriever SYCT400	No
C'1'	F1	HT	Message after verb verification	USRBTLOG (optional)	No
C'2'	F2	HT	Message queued for transmission	FESEND	User
C'3'	F3	HO	Message transmitted	Front End	User
C'4'	F4	HO	3270 output message content invalid--message dropped.	BLHOT	No
X'FF'	FF	HT	Intercomm Restart Accounting	MSGAC	Yes

Internal Code: Log code in core during processing (snaps and dumps)
External Code: Log code after translation by LOGPUT (INTERLOG printout)
Format: HT for header and text, HO for header only
Restart Use: Yes, No, User (specified via user-coded system macros)

Figure 9-1. INTERLOG Entries (Page 2 of 2)

9.4.1 Logging User Exit--USERLOGE

An optional user exit, USERLOGE, is called from LOGPUT after message logging processing is performed. It may be used to dynamically gather response time statistics or to duplex (write to a user data set) certain log records used for installation-specific statistics gathering. For example, the user might wish to write all FA (internal log code F3) records to a separate file to reduce the processing needed for SAM statistics reporting. The log code in the header is the internal code, not the external code, as translation is performed only in the log buffer written to INTERLOG. To distinguish the internal log codes F2 and F3, note the following:

Internal Code	External Code	Differences
F2	01	MSGHRSCH and MSGHRSC not binary zeros.
F2	F2	MSGHRSCH and MSGHRSC are binary zeros.
F3	FA	MSGHRSCH and MSGHRSC not binary zeros.
F3	F3	MSGHRSCH and MSGHRSC are binary zeros.

All log records are passed to the user exit (whether or not written to INTERLOG) except the following:

- Startup record (log code 9F)
- Message Accounting record (log code FF)
- File Recovery records (log codes 80-8F and 90-9E)

The exit must be resident and reentrant and use standard linkage conventions. At entry, register 13 points to a dynamic save area chained back to the caller of LOGPUT and register 1 points to the address of the message passed to LOGPUT; the time, date and thread number fields in the header have been updated. The exit may examine or copy the message header (and text), but may not free or modify the area or address in any way. Any processing by the exit which gives up control to the Dispatcher (I/O etc.) will increase response time and may increase subsystem processing time (if exit processes external log code 30 (internal = C'T') or user logging messages).

9.5 SYSTEM CHECKPOINTS

During startup, the CHECKPT3 program is dispatched, via the Dispatcher timer queue, for an interval of time equal to that value specified in the System Parameter List (SPALIST macro parameter TCHP). When the CHECKPT entry in CHECKPT3 is activated (the time has expired), CHECKPT3 will generate checkpoint records for the checkpoint file, organized as a direct access (BDAM) Data Set with ddname CHEKPTFL. When processing is complete, the checkpoint program again dispatches itself and subsequently idles. The cycle will repeat itself when the new time value expires.

The SPALIST operand GENSW defines the number (maximum is 5) of logical checkpoint areas to be utilized on CHEKPTFL. One checkpoint area usually consists of several physical blocks on the BDAM data set. Bits 1-5 of the byte associated with GENSW indicate usability of checkpoint areas. The default value X'7C' indicates no useable areas. A minimum of three checkpoint areas is required.

The records associated with each checkpoint are constructed in a wraparound or flip-flop manner; that is, if the system fails during the checkpoint processing, the previous checkpoint area remains intact. At restart time, the data is restored exactly as it was when the last complete checkpoint was taken.

The checkpoint routine writes certain fields from system tables onto the checkpoint file. If any table is not present in the system, the Checkpoint Program will bypass processing for that table.

Intercomm allows the user to request data to be checkpointed, in addition to the information Intercomm checkpoints in its own tables. To utilize user checkpointing the user must:

- Indicate the label of the starting point of data to be checkpointed in the CKUSR parameter of SPALIST
- Indicate the length in bytes of the user area to be checkpointed in the CKUSL parameter of SPALIST.

In order to take full advantage of this facility, it is necessary to centralize, in a contiguous area, all the data which is to be retained across restart. (Such a contiguous area could be USERSPA.) The area to be checkpointed should contain only data which would not change if it were loaded into another location. Address constants should not be checkpointed, for example.

The CHEKPTFL data set must be formatted in advance by the off-line utility CREATEGF. (See Chapter 12.) An installation may create minimally 40 blocks, each containing all checkpoint data described in Figure 9-2. Or, given that the amount of data to checkpoint exceeds the physical block capacity of the direct access device, some multiple of 40 blocks must be created. Again, the checkpoint/restore routines function with logical checkpoint areas on the CHEKPTFL data set.

The following formula should be used to calculate the minimum number of blocks which must be formatted by CREATEGF:

$$N=5\left(\frac{13S}{B-8}+\frac{13F}{B-8}+\frac{8C}{B-8}+\frac{12T}{B-8}+\frac{U+2+23}{B-8}\right)+5$$

where:

B = block size (minimum allowed is 64 bytes)
S = # of Station Table entries (STATION macros)
F = # of File Table (PMIFILET) entries
C = # of Subsystem Control Table entries (SYCTTBL macros)
T = # of Time Table entries
U = Length of User Area

All divisions must be rounded up to the nearest integer before summing and multiplication by five.

For implementation of message restart/recovery with checkpointing, see Section 9.7. To synchronize Intercomm checkpoints and file recovery and/or data base checkpointing, there is also a checkpoint subsystem (CHCKPTSS) and other required modules, as described in the File Recovery Users Guide or DBMS Users Guide, as applicable.

9.5.1 Checkpointing User Exit--USRCHKPT

After the checkpoint records are written and the checkpoint time message is issued (RR013I), a user checkpointing exit is called if coded and included as resident in the Intercomm linkedit. At entry, register 1 points to the checkpoint time message (two-byte length field, followed by two-byte MCS flags field, followed by message text). Standard linkage conventions must be used.

This user exit could be used for data base checkpointing coordination (when not provided by Intercomm - see DBMS Users Guide), or to record the checkpoint time for internal reporting purposes. There is no return code processing.

Station Table	(one entry per terminal)	
Company Number	2 Bytes	13 bytes per terminal
Use Code (up/down)	1 Byte	
Terminal-ID	5 Bytes	
Alternate Terminal-ID	5 Bytes	
Change/Display Utility File Table	(one entry per File)	
File Name	8 Bytes	13 bytes per file
Last Number Generated	4 bytes	
File Table Switches	1 Byte	
Time Table	(per entry)	
Scheduled time	4 Bytes (Packed)	12 bytes per entry
Time Control Value	2 Bytes	
Time Zone Code	1 Byte	
Message Sent Indicator	1 Byte	
Processed Indicator	1 Byte	
Program Identification Code	2 Bytes	
Program Message Identifier	1 Byte	
System Parameter List Data		
Total Messages processed	4 Bytes	23 bytes in total
Unused	4 Bytes	
Monitor Sequence Number	4 Bytes	
Number of Messages Cancelled	2 Bytes	
Number of Messages Cancelled by Editing	2 Bytes	
Number of Messages Cancelled (Invalid Subcode)	2 Bytes	
Number of Messages Cancelled (I/O Errors)	2 Bytes	
Number of Messages Cancelled (No Queue Space)	2 Bytes	
Checkpoint File Area Check and Midnight Switch	1 Byte	
User Area (if specified)		
Length of area	2 Bytes	
User-specified area	user-dependent	
Subsystem Control Table	(one entry per subsystem)	
Subcode of subsystem	2 Bytes	8 bytes per subsystem
Total cancelled for subsystem	2 Bytes	
Total processed for subsystem	4 Bytes	

Figure 9-2. Checkpoint Data

9.6 RESTART/RECOVERY

9.6.1 The Restart Process

Intercomm is an event-driven system whereby activites are initiated in response to a message. Therefore, the heart of recovery involves the recovery and/or restarting of appropriate messages. The basis for determining what is required for a particular restart/recovery operation is the Intercomm log. This log consists of entries for all messages that are subject to recovery. The log entries allow determination of message status at the time of failure. Every terminal and subystem message will fall into one of these message status categories:

1. Queued and completely processed prior to the last checkpoint
2. Queued and completely processed subsequent to the last checkpoint
3. Queued but not started processing (transmission)
4. Queued and processing/transmitting at failure

The analysis of the message data in the log is performed during restart by reading the log file backwards from the point of failure. A technique of message accounting has been developed that permits this read back to proceed only as far as is necessary to retrieve those messages needed for restart, as described below.

After messages to be restarted are recovered from the log, they are placed on the queues for their destined subsystems or terminals as the last phase of restart processing.

The restart process is initiated when the word RESTART is found in the PARM field of the Intercomm execution (EXEC) JCL statement. This is the only change whatsoever that distinguishes a restarted Intercomm run. When RESTART is recognized, the restart phase of Intercomm analyzes the log and rebuilds the queues. At that time Intercomm starts reprocessing messages placed in the queues by restart while at the same time receiving and processing messages from the live terminal network (the specification of FIFO queues insures that restart messages are processed prior to live messages).

Checkpoint data is automatically restored at restart time.

9.6.2 Message Accounting

To make the warm restart function as rapid as possible, restart involves a reading backward of the log file only as far as required to recover all necessary messages. This information is developed by the Message Accounting routine, MSGAC, a subprogram of LOGPUT. MSGAC examines log entries as they are made by LOGPUT and determines the "read back point" of the log data set. Periodically, MSGAC will insert message accounting records onto INTERLOG. These records reflect a current read back point. Thus, when restart starts reading INTERLOG backwards, the first message accounting record encountered will instruct restart as to the actual read-back point.

Message accounting records are written at the time Front End, Back End and Multiregion "message complete" status occurs for the last message within a group of messages with monitor sequence numbers (MMN) ranging within a multiple of 255 (that is, when message numbers 1-255 complete, when 256-510 complete, etc.).

9.6.3 Message Restart Logic

The Intercomm message restart procedure is straightforward when no file recovery is considered. When reading the log data set backwards, information from certain message headers is temporarily stored. This stored information is the basis for determining what to do with the header/text log entries as they are encountered. The information from the header is such that it can uniquely identify a message within a subsystem (including recursive entries to a subsystem). Since the log data set is read backwards, message log entries will be encountered in this order:

1. Subsystem completed (normally or abnormally)
2. Subsystem started
3. Message queued for a subsystem

When the "message queued for a subsystem" log entry (header/text) is encountered, the information stored from the previously encountered log entries for this message is examined and the following rules apply to the restart analysis:

- If the message successfully completed (a log code FA was encountered), the message is not restarted.
- If the message failed in processing by a subsystem (time-out, program check, failure to acquire all segments, etc.), then the message is restarted, if its monitor message number, MSGHMMN, is greater than the latest message accounting read-back point.

- If the message had started processing, but not completed at the time of failure (a log code entry 30, but no FA, FB, FC or FD was encountered), then the message is restarted, and its log code is set to "R" indicating that it is an in-process message being restarted.

- If an 01 log entry is encountered without any prior entries (it was on the queue at the time of failure), then it is requeued.

These are the criteria applied to a single message out of context; they may be overridden by other considerations:

- If any "ancestor" of a message is restarted for any reason, the message is discarded. This rule requires some clarification: if during the processing of Message A, Message B is generated, Message A is the mother of Message B. Starting at any message, restart logic can work back to the original terminal input, going from the current message to its mother, the mother's mother, and so on. A daughter message is restarted only if all its ancestors are discarded (not logged or not to be restarted). This applies to Front End as well as Back End messages.

- If the message is part of a conversation (subsystem logic uses the CONVERSE facility) and CNVREST=YES is coded in the subsystem's SYCTTBL macro, the message will be restarted if it is the first message in the conversation (even if it completed) and discarded if it is not the first (even if it didn't complete). Note that in order to insure file integrity, conversational subsystems performing data base updates should be designed so that either a message is switched to a nonconversational subsystem to perform the update(s), or the update(s) are performed as processing logic for the last message in the conversation.

- If a message is part of a segmented message sent to OUTPUT or CHANGE/DISPLAY and SEGREST=YES is coded in the sending subsystem's SYCTTBL macro, then the disposition of the trailer (final) segment determines what happens to the other segments. They will be restarted if the trailer was restarted, discarded if the trailer was discarded.

Whether message restart is actually performed depends on the user RESTART specification on each SYCTTBL or BTERM/LUNIT macro. If RESTART=NO was coded, then no messages will be restarted regardless of the circumstances. RESTART=IFPOSBL affects message accounting so that the read-back point for restart analysis may or may not include all those IFPOSBL messages. The read-back point will definitely include all RESTART=YES messages. For Multiregion messages, see Multiregion Support Facility.

The closedown subsystem must have the SCT specification RESTART=NO. Otherwise, system failures during closedown and subsequent message restart would cause the closedown subsystem to be activated immediately. RESTART=NO should also be coded for the GPSS subsystem in case the SNAP or ABND commands are used.

In all those cases where file or data base recovery is not included, the only integrity problem concerning a restart involves those messages that were in process at time of failure. Thus, if a message was being transmitted when a power failure occurred, the restarted Intercomm would retransmit that entire message.

In a complete system failure (example, machine or power failure), Intercomm cannot determine the status of terminal transmissions in process at the time of failure. Therefore, following complete failure, the remote terminal operator must verify the conclusion of his last operation if it was an update operation and if he had not received completely all results from that operation. This is the only terminal operator interaction relevant to restarted Intercomm.

The user can optionally suppress Front End or terminal restart completely. That is, all messages which were queued for terminal output at the time of Intercomm termination will be discarded during restart mode, regardless of terminal restart parameters. This is done by setting location LOGTRT plus displacements hex 'F2' and 'F3' to X'00' in the restart module LOGPROC.

9.6.4 Message Restart User Exit--USRESTRT

This user exit is called by LOGPROC to allow the user to determine disposition of a message eligible for restart. The exit must be serially reusable; standard linkage conventions are used. At entry, register 1 points to the message to be requeued (restarted). The exit may examine the log code (MSGHLOG) to test the message type, as follows:

C'2' or X'F2' -- Front End output--check MSGHTID

C'A' or X'C1' -- Input to be requeued for a satellite region: MSGHMRDX contains the region id number

C'P' or X'D7' -- Data Base update subsystem message to be reprocessed

C'R' or X'D9' -- Non-DB update subsystem message to be reprocessed

X'01' -- Unprocesssed (never started) or failing subsystem message

For the last three message types, check the receiving subsystem codes (MSGHRSCH,MSGHRSC). Multiregion messages can exist only in the control region and are used to recreate the satellite region queues (see Multiregion Support Facility).

Because it is called during initial Intercomm startup, the user exit may not give up control to the dispatcher nor call the File Handler. At exit it must tell LOGPROC whether to restart the message via the value returned in register 15:

- binary zeros = requeue the message
- nonzero = discard the message

The user exit may wish to discard a message if the subsystem no longer exists or will program check, or if the terminal (see MSGHTID) no longer exists or is out of service. The user may alter message header or text fields. Data Base subsystem messages should not be altered or discarded if coordinated checkpointing and backout of DB updates is used.

9.7 IMPLEMENTATION

The following load modules are required for message restart/recovery functions:

Module	Functions
STAEEXIT	Abend Intercept Routine--assures data sets are closed at abend time; in particular required to ensure log buffers are flushed to INTERLOG.
LOGPUT,MSGAC	System logging and message accounting
CHECKPT3,RESTORE3	Checkpoint processing, restore checkpointed values
LOGPROC,INTDBLOK, READBACK	Analysis of Restart log; deblocking, read backward modules.

Coding CHKRES=YES on the ICOMLINK macro will generate the applicable INCLUDE statements.

There are two SPALIST parameters specifying the number of log buffers to get (LGNUM) and the average buffer length (LGBLK). These numbers should be chosen with care, because if logging requests accumulate faster than LOGPUT can handle them, the performance of the whole system degrades. LGBLK should be big enough so that every frequently generated log entry (message or file recovery record) will fit in a buffer. Logging an entry bigger than LGBLK effectively ties up two of LOGPUT's buffers: the active, partially filled buffer is queued to be written, storage is gotten for a temporary buffer to hold the log entry, and then another one of LOGPUT's own buffers is marked as full so the buffer WRITEs can be chained.

The more synchronous logging performed, the smaller and more numerous your buffers should be (recall that synchronous logging, requested by coding LSYNCH=YES, means LOGPUT doesn't return to its caller until the log entry is written). A synchronous logging request causes the buffer containing the entry to be queued immediately for writing, whether or not the buffer is full. Any leftover space is wasted. Thus, there is no point in making buffers big enough to hold ten messages, say, if one log request in five is synchronous.

Aside from RESTART as an EXEC card parameter, JCL requirements for restart/recovery functions are identical to STARTUP mode plus the following DD statements:

- INTERLOG--DD statement for the system log data set to be created in the current run
- CHEKPTFL--DD statement for the BDAM data set containing checkpoint records (if created in the previous execution)
- LOGDISK--DD statement for a BDAM data set used at restart time to hold all messages to be restarted. The maximum message length is thus restricted to the track capacity of the direct access device used. No preformatting is required.
- RESTRTLG--DD statement for the system log to be restarted.

Code the following on the INTERLOG DD Statement:

1. DCB=(...,NCP=number-equal-to-LGNUM,...)

   Recall that LOGPUT writes the log using BSAM, from buffers acquired by the startup routine; this DCB parameter allows LOGPUT to start writes on subsequent buffers before waiting for the first write to complete. If NCP is allowed to default to 1 and LOGPUT issues two writes in a row, the second buffer may be lost for the rest of the run, because BSAM ignores I/O requests once it has NCP operations in progress. It never posts the DECB. Unless volume is very low, this kind of attrition will eventually reduce the number of live buffers to NCP. Therefore, code the NCP parameter to match the LGNUM parameter on the SPALIST macro.

2. DCB=(...,OPTCD=C,...)

   Requests chained scheduling, that is, consolidation of channel programs when more than one write request is queued up.

3. Verify that BUFNO is not specified in the DCB parameter sublist. LOGPUT doesn't use OS buffers.

4. BLKSIZE specifies the actual maximum length block that LOGPUT may use. You can specify BLKSIZE=32760 or maximum track size to make sure everything gets logged, but if the log file is ever used in a restart run you must specify what its real block size is (unless you can spare 32K for a buffer), and it's probably better to settle on one figure for both STARTUP and RESTART modes. The minimum block size is LGBLK+4. The extra four bytes are for the block descriptor word. BLKSIZE must be a multiple of 4. A multiple of 2K (MFT/VS1) or 4K (MVT/MVS) is recommended. For MVS, an NCP of at least 10 and LGBLK of 4K or 8K is recommended, as using many small buffers is more efficient (less paging, etc.) than using a few large ones.

   Sample JCL:

```
//INTERLOG   DD DSN=INTERLOG,DISP=(NEW,KEEP),
//           UNIT=unit,VOL=SER=volser,LABEL=(,BLP),
//           DCB=(DSORG=PS,RECFM=VB,BLKSIZE=blksize,LRECL=blksize-4,
//           OPTCD=C,NCP=lgnum)
```

NOTE: In order to correctly reposition an INTERLOG tape following a loss of power or operating system failure, the tape must be preformatted with tape marks. For restart from a disk data set which was not closed (that is, after a system crash) at the end of a previous Intercomm execution, see the description of ICOMFEOF in Chapter 12. If disk logging is used, omit the LABEL parameter and add a SPACE parameter; SPACE=(CYL,(primary)) is recommended. Do not specify secondary extents if recovery via ICOMFEOF might be executed. In order to reduce disk space utilization, the Intercomm sequential output disk file flip/flop facility described in Chapter 6 should be implemented.

For checkpointing, include a DD statement for CHEKPTFL. This direct access file is used by CHECKPT3 to store checkpoint information. It must be preformatted by CREATEGF (see Chapter 12) before the first execution with checkpointing. The file must have a block size of at least 64, and at least 40 blocks.

Sample JCL:

```
//CHEKPTFL   DD UNIT=direct-access-device,VOL=SER=volser,
//           DSN=INT.CKPOINT,DISP=OLD,
//           DCB=(DSORG=DA,OPTCD=RF)
```

For message restart, LOGPROC uses a temporary disk data set that holds messages to be restarted. The data set is variable-format with one message per block, so its block size must be equal to the length of the longest message that can be produced for a restartable subsystem or terminal. The data set is created by LOGPROC, so preformatting is unnecessary. In the following, 'm' stands for the maximum message size.

```
//LOGDISK    DD  UNIT=direct-access-work-unit,
//               SPACE=(m,(primary,secondary),RLSE),
//               DCB=(DSORG=DA,BLKSIZE=m,RECFM=F)
```

Figure 9-3 illustrates JCL for Intercomm log files for restart. Note that RECFM=U for the restart log data set.

Both INTERLOG and RESTRTLG may be defined as the same tape unit. In this case, logging is suppressed while reading RESTRTLG and performing the restart function. Logging begins when restart functions are complete. For one-tape-drive mode, either make the volume-serial numbers identical so that both data sets are assigned to the same drive, or code UNIT=AFF=RESTRTLG on the INTERLOG DD statement.

The LABEL parameters assume standard labels. If you are using unlabeled tapes, code LABEL=(,BLP) or LABEL=(,NL) in both DD statements. LABEL=(,SUL) is recommended for INTERLOG; this will cause the user label exit to be taken in the File Handler and will prevent time-outs that occur during the mounting of a new tape volume. If restart is from disk, omit the LABEL parameter.

```
Input Log
//RESTRTLG   DD  DSN=anyname,DISP=(OLD,PASS),
//                UNIT=unit,VOL=SER=volser,LABEL=(,BLP),
//                DCB=(DSORG=PS,RECFM=U,BLKSIZE=blksize)

Output Log                              {NEW}
//INTERLOG   DD  DSN=anyname,DISP=({MOD},KEEP),
//                UNIT=unit,VOL=SER=volser,LABEL=(,SUL),
//                DCB=(DSORG=PS,RECFM=VB,BLKSIZE=blksize,
//                LRECL=blksize-4,OPTCD=C,NCP=lgnum)
```

Figure 9-3. RESTRTLG and INTERLOG DD Statement Examples

For restart, the same checkpoint file used in the previous execution is reused. The JCL also remains the same.

9.7.1 Concatenation of Disk Log Files for Restart

If disk logging is used with the flip/flop facility, the user should off-load each log file as it becomes filled so it will be available for reuse at a later time. To unload the disk log, copy the file to another disk file or to tape using IEBGENER; RECFM=U must be specified for both input and output files; do not use RECFM=VB or try to reblock the file.

If Intercomm goes down, all of the disk log files can be concatenated and used for Intercomm restart. If the operating system crashes (IPL required) while Intercomm is executing, use IEBGENER as described above to copy the file to set an EOF at the end of the extent (occurs even if IEBGENER abends due to garbage data). Then execute the Intercomm utility ICOMFEOF to set the true end-of-file. Intercomm will read backwards through as many files as necessary in order to restart. In the JCL, the order of concatenation for the log files should be from the newest to oldest. For example, suppose that 'INT.NEWEST' is the most recent disk log file (the one being used when the system went down), and 'INT.OLDEST' is the oldest (the first log file used or off-loaded). Then, the RESTRTLG DD statements would look as follows:

```
//RESTRTLG   DD  DSN=INT.NEWEST,....
//           DD  DSN=INT.OLDEST,....
```

If there were other disk data sets filled during the run, they would be inserted in the same reverse chronological order between 'INT.NEWEST' and 'INT.OLDEST'.

Note that if any log files are on tape, the tape files may not be concatenated to (nor mixed with) the disk files nor may the tape data sets be concatenated by themselves and be restarted. This is due to an IBM restriction that concatenated tape data sets may not be read backwards. Intercomm handles tape data sets as multivolume data sets - when it reaches the end of one tape, it will request another volume. When a disk log concatenation is used, data set switching is executed internally by READBACK. LOGPROC continues to request a previous log block to be read until it finds a message accounting record (log code X'FF') or until the beginning of the last tape volume or disk data set is reached. Then the RESTRTLG file is closed. It is important to remember the difference between restarting tape logs and disk logs, as summarized in the following table:

	TAPE	DISK
JCL	No concatenation. Code last tape VOLSER used in JCL (LABEL = NL or BLP), followed by previous VOLSERs used (in reverse order).	Concatenate log files via JCL. Data set switching executed internally.
Action at End of Data	Requests another volume via WTO, until last tape read backwards (if necessary); closes RESTRTLG.	Closes RESTRTLG. Does not request another volume or DSN to be read.
Method of Operation	Reads Backward via BSAM (READ SB)	Uses BDAM and EXCP to read backwards using actual track addresses for each block.

9.8 SERIAL RESTART

Normally, restarted messages are multithreaded. That is, all restarted messages will be requeued for their respective subsystems immediately and each subsystem will process as many messages as it can concurrently, up to its maximum. This scheme is undesirable in some circumstances; for example, systems using a data base management system may want updates to take place in the same order as they were originally entered. A multithread environment cannot guarantee this will be accomplished if more than one subsystem updates the same data base or the update subsystem has a MNCL (concurrency) greater than one.

To solve this problem, a serial (single-thread) restart capability is provided with Intercomm which employs the Intercomm Dynamic Data Queuing facility. With the serial restart feature, Back End messages are restarted one at a time and the next message to be restarted for any subsystem is not queued until the previous one is completed. Thus, use of the serial restart feature will ensure that only one restarted message will start processing at any given time.

While processing a restarted message, if a subsystem queues a message for another subsystem, that "daughter" message will be processed on a multithread basis concurrently with the next restarted message. Messages queued for the Output Utility will also be multithreaded. New messages from the Front End which must be processed by a subsystem will be rejected unless a user exit (USRSEREX) is coded to permit the processing of selected input messages (such as sign-on security or inquiry messages unrelated to the serially restarted update subsystem). Front End commands which are not processed by a subsystem (see System Control Commands), and output terminal messages will be

processed normally during serial restart. Message Collection determines that a message is from the Front End by examining the sending subsystem codes (MSGHSSCH, MSGHSSC); if both are binary zeros, the message is assumed to be a new input message. Both message restart and Log Analysis require the coding of the sending subsystem codes for all "daughter" messages (including those passed to the Output Utility). If a "daughter" subsystem performs the critical update, the user may force serial restart on the "daughter" subsystem by coding RESTART=NO on the SYCTTBL for the "mother" subsystem.

In a Multiregion Intercomm using serial restart, each satellite region must have its own log and each region must be restarted using its own log. All data base update subsystems should be placed in the same satellite region if it is desired that data base updates take place in the same order on restart as they did originally. In the Multiregion control region, messages to be passed to a satellite region and messages received from a satellite region will be processed normally. Serial restart should be employed only in those regions containing critical update subsystems. The other regions may use multithread restart or can omit restart processing entirely.

To install serial restart, follow the instructions for installing normal restart with the following additions:

- The Intercomm DDQ special feature must be installed in the system and the default DDQ data set must be defined. (See Dynamic Data Queuing for details.) A transient DDQ is used to hold the restarted messages in the order in which they were originally logged. The block size of the DDQ data set must be at least as large as the longest possible requeued message (including header).

- The module REQONDDQ must be included as resident in the Intercomm region linkedit. For multiregion systems, it must be linked into each region using serial restart.

- If coded, the user exit USRSEREX must also be included as resident in the Intercomm linkedit (sample exit routine provided on SYMREL).

9.8.1 Serial Restart User Exit--USRSEREX

This exit routine is called by BLMSGCOL when message collection is called to queue a message to a subsystem from the Front End while single thread restart is in progress. That is, when serial restart is in progress, BLMSGCOL will call USRSEREX before queuing a Front End message for a subsystem. The exit is never called when queuing a restarted message, or a message from a subsystem to another subsystem or to a terminal, nor is it called after serial restart has completed.

Upon entry to the USRSEREX routine, register 1 contains the address of a two-fullword parameter list. These fullwords point to the input message and the region SPA respectively. USRSEREX can examine the message and must pass a return code in register 15 which tells message collection what action to take; a return code of zero tells message collection to free the message and return to the caller; that is, if USRSEREX returns a return code of 0, BLMSGCOL does not queue the message for the subsystem. Instead, the message is freed and the following message is returned to the originating terminal:

PREVIOUS MESSAGE REJECTED, SERIAL RESTART IN PROGRESS

If USRSEREX returns any nonzero return code in register 15, message collection will queue the message for the subsystem as it normally does and the message may be processed while serial restart is still in progress. USRSEREX may not free the message as that will be done by the system when required. However, USRSEREX may issue STORAGE and STORFREE macros for any storage area the routine requires.

Because BLMSGCOL is Link Pack eligible, a pointer to USRSEREX is kept in the SPA extension for each region (control and satellite) in a Multiregion environment. Implementation of this feature thus becomes a matter of linking or not linking USRSEREX as resident within the region load module. If USRSEREX is not linked in, the call is skipped by Message Collection and the Front End message is rejected. An installation may provide different exit routines for each region or no exit routine for some regions, depending on requirements. For example, the user exit can be omitted for any regions that do data base updates or file recovery to force Front End messages to be rejected until single-thread restart is complete. A region doing mostly inquiry processing may need the user exit to allow most input messages to be processed.

The USRSEREX module must have a Csect name of USRSEREX and should be reentrant, especially if any Intercomm facility, such as the Dispatcher or File Handler, may be used. It is recommended that the user exit avoid admitting messages that will be in the system for a long duration. That is, messages for subsystems that do multiple file I/O's or generate numerous output screens, for example, may remain in the system for a relatively long time, thus slowing down serial restart considerably. Discretion must be used when choosing the number and types of messages that the exit allows to be processed during serial restart.

A message queued by a subsystem for another subsystem will be intercepted by USRSEREX if the subsystem fails to set the sending subsystem codes (MSGHSSC, MSGHSSCH) correctly. The Intercomm Programmers Guides state that a subsystem which queues a message for another subsystem must set the fields if Intercomm restart/recovery is to be used. USRSEREX will not be called to process a message whose sending subsystem code bytes are not binary zeros (binary zero indicates the message originated in the Front End). This allows any messages which result from a message already passed by USRSEREX to be queued without further checks.

One final consideration in designing the USRSEREX module is the processing of Intercomm control commands. Some control commands, such as FILE, LOAD, and TALY, are processed by Back End subsystems under Intercomm and hence must have messages queued for them via message collection (other commands, such as FLSH and RLSE, are processed by the Front End and this discussion does not apply to them). Consult System Control Commands for information on which commands are Front End or Back End commands. The commands processed by the Back End will usually originate in the Front End and so will be passed to USRSEREX for processing before being queued for the processing subsystem when serial restart is in progress. The user, when coding USRSEREX, must decide which, if any, Back End control commands shall be allowed during serial restart. This decision should be based on the command function and the logic of the applications running under serial restart mode. For example, it does not make sense to allow a FILE command to LOCK a data set that may be required by a subsystem which will process a restarted message, but it may be valuable to use the TALY command. The user must decide which control commands to allow in the USRSEREX.

When designing the exit routine, the user must also be aware that some subsystems may process a number of commands. For example, the GPSS subsystem processes FILE, TALY, STRT and STOP commands, among others, and so the exit routine may have to check the message verb as well as the receiving subsystem codes when checking for control commands or user verbs that it wishes to allow to be queued.

The sample USRSEREX released with the system allows only selected Intercomm Back End commands to be processed. This routine may be modified by the user to allow some user functions if desired. If using the Intercomm Extended Security System, SECU commands must be allowed to be processed, so that sign-on prompt screens may be transmitted and terminal operators may sign on to Intercomm while serial restart is in progress. This also applies to any other security scheme employed (Intercomm's Basic Security, etc.).

Note: If EDIT=BQ is specified for the input verb (on BTVERB macro), the input message will be edited before being passed to the user exit. Therefore, the user exit should employ testing of the receiving subsystem codes (MSGHRSCH, MSGHRSC), rather than the verb, except as noted above for multipurpose verbs which are not edited before queuing is attempted.

Chapter 10

SYSTEM SECURITY IMPLEMENTATION

10.1 INTRODUCTION

Under Intercomm, two security systems are provided:

- Extended Security - a special feature - is dynamically created and controlled on-line via commands and provides a full range of security control over all major system resources such as terminals, verbs, subsystems, files, regions, and user-specified functions (such as data base access). Secured terminals require operator sign-on with an ID and, optionally, a password. The ranges of resource accessibility are defined for each operator. Extended Security is fully described in Extended Security System.

- Basic Security - a table-driven system available to all Intercomm installations - provides sign-on security at specified terminals, transaction (verb) security at specified terminals, and optionally, user-coded exits for additional security processing at sign-on, sign-off, and subsystem access time.

This chapter describes Basic Security implementation within the Intercomm system and covers the following subjects:

- Basic Security processing options
- Implementation of sign-on/sign-off security and user exits for additional processing
- Implementation of transaction security
- Coding the Station Table for security installation
- Implementation of user-written security routines for subsystem access control

In this chapter, the conventions for input message formats are denoted by the following:

$ indicates the system separator character defined in the installation's system parameter area (SPA).

@ indicates the end-of-message sequence of the terminal (EOT, EOB, ETX, etc).

Responses to sign-on/sign-off and system security control commands are described in System Control Commands.

10.2 BASIC SECURITY PROCESSING OPTIONS

Under Basic Security, Intercomm provides the user with three types of system security options:

- Station Sign-on/Sign-off Security

  Station sign-on/sign-off security checking allows the installation to limit the use of a specified set of terminals to only those operators who sign on using pre-assigned numerical operator security codes. Not all stations need be under the sign-on/sign-off facility.

- Transaction Sign-On/Sign-Off Security

  Transaction security checking allows the installation to specify which transaction codes (verbs) are allowed entry from a particular station. Not all verbs need be under transaction security.

- User-Written Security

  User-written security allows the user to insert other types of security which may be desired.

Control commands affecting system security are also described. Installation and use of the commands are defined in System Control Commands.

Any one of the above types of security checking or any combination of these types is available to the user under Basic Security. Requirements for security options are specified in system tables by subsystem (SYCTTBL macro), terminal (STATION macro) and in the System Parameter Area (SPALIST macro).

10.2.1 Security Processing Logic

The following examples describe Intercomm Basic Security processing logic as illustrated in Figure 10-1. Assume the following list of terminals are under security check with the associated operator codes:

NYC01	allows only operator codes 1, 5, 7, 10
CHI01	allows only operator codes 2, 3, 4, 8
SFI01	allows only operator codes 1, 6, 7, 9
ABT01	allows only operator codes 1, 5, 7, 9

Assume the following list of verbs are under security check with the associated terminals:

SEND	allowed only through NYC01, CHI01
SHIP	allowed only through SFI01
DELI	allowed only through NYC01
MAIL	allowed only through ABT01, CHI01
TRUC	allowed only through NYC01

1. Operator 1 attempts to sign on at the NYC01 terminal. He is allowed on. He then enters the verb SEND. The message is processed. He signs off.

2. Operator 6 attempts to sign on at SFI01. He is allowed on. He then enters the verb SHIP. The message is processed. He may enter additional transactions using SHIP.

3. Operator 5 attempts to sign on at NYC01. He is allowed on. He then enters the verb THIP. An unknown verb error message is sent to the terminal. He signs off.

4. Operator 5 attempts to sign on at CHI01. He receives an error message; sign-on is cancelled.

5. Operator 4 attempts to sign on at CHI01. He is allowed on. He then enters the verb DELI. The incoming message is cancelled; the operator receives an unauthorized verb error message.

6. Operator 4 is still on CHI01. He has read his error message. He then enters the verb SEND. The message is processed. He may enter other transactions using SEND or MAIL.

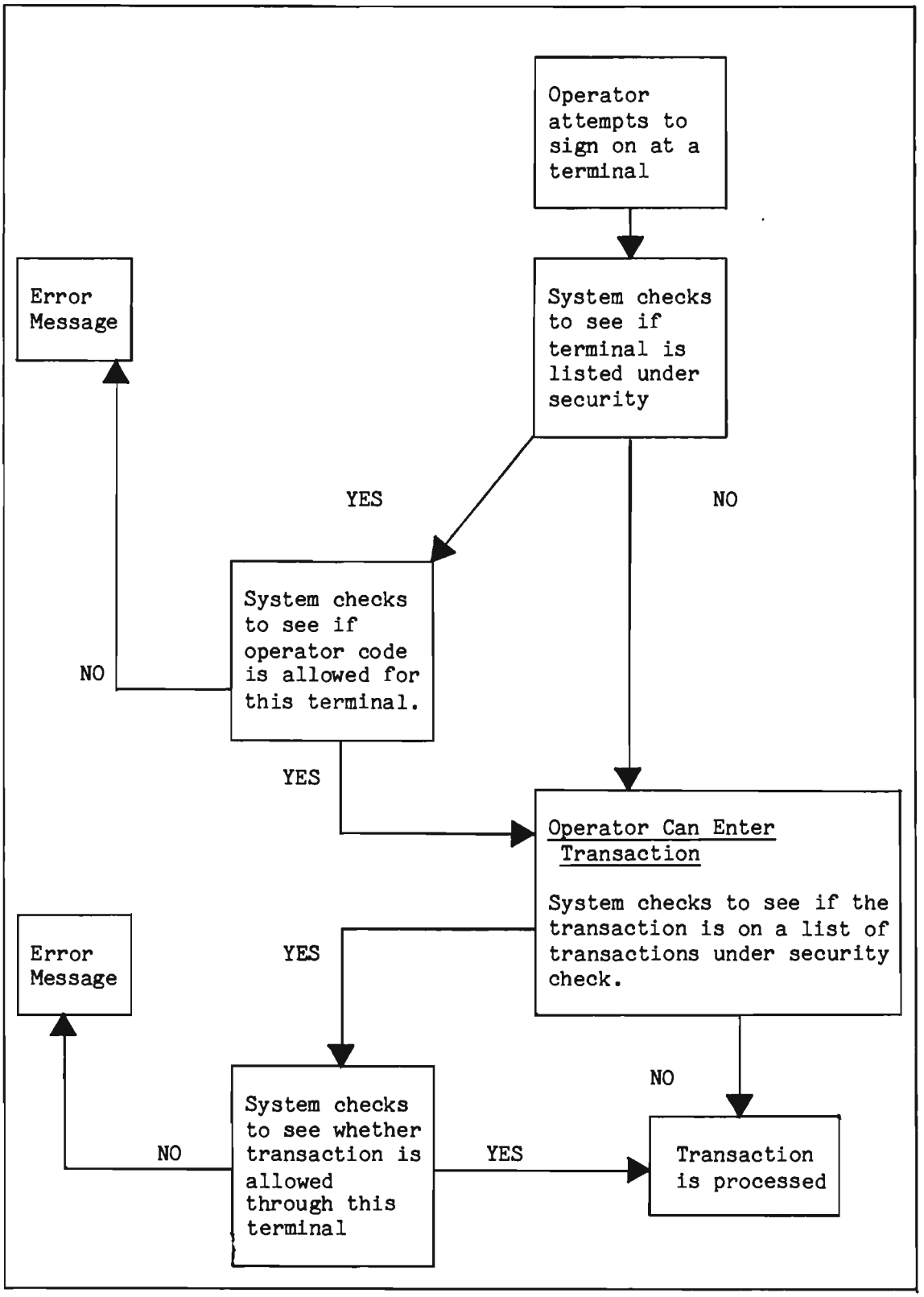


Figure 10-1. Security Processing Logic

10.3 SIGN-ON/SIGN-OFF SECURITY

This section describes implementation of operator sign-on security which involves three interrelated system areas: defining operator codes for secured terminals, activating systemwide terminal security via the SPALIST, and optionally requiring sign-on security before certain subsystems may process an input transaction.

10.3.1 Using a Sign-on/Sign-off Terminal

Before using a station that requires a sign-on/sign-off security check, a terminal operator signs on by entering the following message:

SIGNONoperator-code@

The operator code must be numeric and may be any number from 1 to 2147483647. If the operator code is not defined via the STATION macro, OPER parameter, for that terminal, an error message will be sent to the originating and control terminal. The operator is allowed "n" attempts to sign on before access is terminated for that station. The "n" is set in the STATION macro via the MAXSIGN parameter.

The sign-off command format is:

SIGN$OFF@

An operator is not allowed to sign on at the same time at more than one terminal under security check, and only one operator at a time is allowed to sign on at a secured terminal. If an operator signs on to a terminal and is already signed on at another terminal, an error message will be received. If an operator signs on at a terminal and another operator has already signed on to that terminal, the first operator will be signed off. For instance, operator 111 signs on to terminal NYC01 and does not sign off, and then operator 222 signs on to terminal NYC01. Terminal NYC01 will first sign off operator 111 and then sign on Operator 222.

If an operator attempts to enter a verb at a secured terminal without signing on, an error message will be received. An operator must remember to sign off from a terminal under security check to which he has signed on; otherwise the terminal may be used by other operators without signing on, thereby compromising security.

An automatic sign-off feature is included if certain specifications exist in the SPALIST and the terminal's STATION macro. It is important that the operator be aware of whether or not the automatic sign-off is in effect. If it is, the terminal will be signed off automatically after the prespecified elapsed time expires, whether the operator is ready to be signed off or not.

10.3.2 Sign-on/Sign-off Processing

Sign-on/sign-off (SIGN command) is processed by a subsystem which must be represented by a Subsystem Control Table entry (SYCTTBL macro) for the PMISIGN subsystem as follows:

```
SYCTTBL  SUBC=S,SBSP=PMISIGN,OVLY=0,LANG=RBAL,SECU=00,                X
         TISE=YES,SOSO=NO,NUMCL=4,BLRI=F,ECB=YES,SPAC=1000,           X
         MNCL=5,PRTY=0,RESTART=NO,LOG=NO,...
```

10.3.3 SPALIST Macro Parameter

To use the sign-on/sign-off option, the security parameter SONOFF=YES must be coded on the SPALIST macro. SONOFF=NO takes priority over all SYCTTBL entries which request sign-on/sign-off security.

There are two transactions which can be entered from a terminal and which affect the SONOFF parameter in SPALIST. The verb ASGN activates the sign-on/off feature for the entire system even if SONOFF=YES has not been specified in the SPALIST macro; that is, ASGN activates, or turns on, SONOFF=YES. The verb DSGN deactivates the sign-on/off feature for the entire system, even if SONOFF=YES was specified in the SPALIST macro. DSGN turns off the SONOFF parameter; by changing SONOFF=YES to SONOFF=NO. These special transactions are also processed by the PMISIGN subsystem. Their formats are described in System Control Commands. It is recommended that ASGN and DSGN be restricted to the control terminal via the BTVERB macro SECUR parameter (SECUR=YES).

The SPALIST macro SGNTIME parameter specifies, in minutes, the default time interval to be used to automatically sign off a terminal after an operator has signed on at that terminal; that is, the default duration a terminal may retain a signed-on security clearance before that clearance is to be automatically revoked (code as a decimal from 0-466). This time interval will be used only when the following conditions exist:

- the sign-on/sign-off feature is active
- the terminal's STATION macro specifies AUTOFF=YES
- the terminal's STATION macro specifies TIME=0
- a sign-off transaction has not already been entered

10.3.4 SYCTTBL Macro Parameter

For each subsystem which may only process input from a secured terminal, the SOSO parameter in the SYCTTBL macro defining that subsystem must be coded SOSO=YES. If in the SPALIST macro the SONOFF parameter is coded SONOFF=NO, then that will take priority; that is, SONOFF=NO turns off the sign-on/sign-off security option for the entire system, even if some subsystem SYCTTBL macros have SOSO=YES. If SONOFF=YES, and a subsystem SYCTTBL has SOSO=NO, then the sign-on requirement does not hold for the subsystem.

The relationship of the SPALIST SONOFF parameter and the SYCTTBL SOSO parameter is summarized as follows:

SONOFF	SOSO	Result
YES	NO	Sign-on/sign-off security option does not hold for subsystem
NO	YES	Sign-on/sign-off security option does not hold for subsystem
YES	YES	Sign-on/sign-off security in effect for subsystem

10.3.5 User Exits for Sign-on/Sign-off Security

An installation may add user-coded exit routines which can be designed to accumulate statistical information and to perform additional sign-on or sign-off processing. The two exit routines the user is allowed are USRSGNON and USRSGNOF. The exit routines must be coded using standard linkage conventions and must be linkedited as resident in Intercomm.

The user sign-on exit routine is called before the station is actually signed on, and after the Intercomm checks have been performed. The entry point for user sign-on is USRSGNON.

The parameter list passed via register 1 is as follows:

1. The address of the station's entry in the Station Table(-6). Use the STALIST macro to generate the Dsect for the Station Table entry. The Dsect includes six bytes of header information which appears prior to the first entry in the table only. Thus the address passed is "table-entry minus six" to allow proper reference to the exact table entry.

2. The address of the SPA.

3. The address of the sign-on message. Define a labeled Dsect statement and COPY MSGHDRC to form the message header Dsect. If the field MSGHVMI is X'FF' or X'00', then a normal sign-on is indicated. If the field is X'FD', a sign-on message has been received for a terminal that is already signed on.

4. The address of operator security information for the station in PMISTATB. Use the SECTB macro to define the Dsect.

5. The address of the return code, which is a fullword. At exit, if the word is binary zero, then sign-on will be completed. If it is nonzero, sign-on will be terminated and error messages will be sent to the originating and control terminals. In the latter case, the user exit must free the input message areas using a STORFREE macro; the length is in the first two bytes of the message header.

The user sign-off exit routine is called before the station is signed off. The entry point is USRSGNOF. The parameter list passed to USRSGNOF is similar to that for USRSGNON, as follows:

1. The address of the entry for the station in the Station Table(-6).

2. The address of the SPA.

3. The address of the message. If the field MSGHVMI is X'FF' or X'00', then a normal sign-off is indicated. If the field is X'FE', the message was generated by the automatic sign-off function.

4. The same as for USRSGNON.

NOTE: USRSGNOF cannot cancel the sign-off function.

10.4 TRANSACTION SECURITY

As with terminal security, transaction security involves three interrelated system areas: defining permitted transaction codes for each station, systemwide transaction security via the SPALIST, and optionally requiring transaction security before certain subsystems may process an input transaction.

An additional form of transaction security, which operates independently of, and overrides, systemwide transaction checking options, is provided by the parameter SECUR=YES coded on a BTVERB macro in the Front End Verb Table. Such a verb may be entered only from the control terminal (internally forced for the system commands NRCD and IMCD, used to close down Intercomm). The default is SECUR=NO. Control terminal transaction security may be dynamically controlled by the system commands SECN and SECF (control terminal security on/off for the specified verb).

10.4.1 Using Transaction Security

If the transaction security option is in effect for the system, each verb entered by an operator at a particular terminal is checked. If the transaction code is a secured verb, and if allowed from that terminal, the transaction is processed as usual. If not allowed, the incoming message is rejected and the operator receives an error message. A list of secured verbs is defined in the STATION table via a SECVERBS macro; the allowed verbs from that list which apply to a specific terminal are defined via the VERBS parameter of the STATION macro, as described in section 10.5.

Transaction security checking is performed after message dequeuing by the Subsystem Controller. The option to edit (by the Edit Utility) an input message before queuing (BTVERB macro, EDIT=BQ) may not be used if transaction security is to be effected for that verb.

For each station, the user has the option of adding or deleting allowable transactions from the SECVERBS list via the system commands SWON or SWOF.

10.4.2 SPALIST Macro Parameter

If the user intends to employ the transaction security option, the security parameter TRANSEC=YES must be coded on the SPALIST macro.

There are two system control commands that can be entered on-line which are able to activate or deactivate the transaction security option systemwide. The AVRB transaction activates the security by verb feature, even if TRANSEC=YES was not coded in the SPALIST macro. The DVRB transaction deactivates the security by verb feature even if TRANSEC=YES was specified in the SPALIST macro; that is, it sets TRANSEC=NO.

10.4.3 SYCTTBL Macro Parameter

TISE=YES must appear in the subsystem SYCTTBL if the user requires transaction security for transactions going to that subsystem.

If, in the SPALIST macro, TRANSEC=NO, then the transaction security option is turned off for the whole system, even if in a subsystem's SYCTTBL macro TISE=YES. If TRANSEC=YES, but TISE=NO is coded on a subsystem SYCTTBL macro, then transaction security is not in effect for that subsystem. The relationship of TRANSEC and TISE is as follows:

TRANSEC	TISE	Result
NO	YES	Transaction security turned off systemwide
YES	NO	Transaction security turned off for that subsystem
YES	YES	Transaction security in effect for the subsystem

10.5 CODING THE STATION TABLE

This section describes the macros and parameters for the Station Table which are necessary to implement terminal and/or transaction security.

10.5.1 Structure of the Station Table with Security Processing

When sign-on/sign-off and/or transaction security is to be implemented, the Back End Station Table (PMISTATB Csect) must be expanded to identify security requirements.

The structure of the Station Table when security processing is utilized, and the positioning of user-coded macros, is illustrated below:

```
PMISTATB    CSECT
            GENSEC    OPER=CORE
            SECVERBS  - - - - -
            STATION   - - - - -
            STATION   - - - - -
              .
              .
              .
              .
            PMISTOP
            END
```

10.5.2 GENSEC Macro

If any of the Intercomm Basic Security checking options are going to be used, the user must supply a GENSEC macro. Only one GENSEC macro is coded, and it must appear before all SECVERBS and STATION macros in PMISTATB. It notifies the macro processor that one or more security table entries are to be generated in a separate Csect PMISECTB.

The OPER parameter of the GENSEC macro indicates whether all the operator security codes covered by the sign-on/sign-off option will reside in core or on disk. Further details on how the codes are made resident on disk (data set SEC000) appear later; OPER=CORE must be used to indicate only core-resident entries.

10.5.3 SECVERBS Macro and STATION Macro/VERBS Parameter

If the transaction security option is used, one or more SECVERBS macros must be coded, and must precede all STATION macros. The SECVERBS macro has two parameters: VERBS and TABLE.

All transactions to receive a transaction security check must be specified in the VERBS parameter of SECVERBS macros. The maximum total number of transaction-ids permissible within the parameter sublists is 2048. The STATION macros define, for each individual terminal, the subset of transaction-id's allowed entry from that terminal. The transaction-ids in the STATION macros must come from the list in the SECVERBS macro, VERBS parameter.

The TABLE parameter specifies whether or not an in-line table consisting solely of the transaction-ids supplied by the VERBS parameter is to be generated. If TABLE=YES, an in-line table is generated. If TABLE=NO, an in-line table is not generated.

If a Front End does not exist (Test Mode or Basic TCAM interface), the parameter TABLE=YES must appear in the SECVERBS macro.

If a BTAM/TCAM/VTAM Front End does exist, and the user wishes to conserve main storage, then TABLE=NO is allowed. In this case, the transactions under security must be specified by a set of BTVERB macros, in the BTVRBTB table, in the identical order in which they appear in the VERBS parameter of the SECVERBS macro. In addition, they must precede all other BTVERB macro instructions defining transaction-id's that do not require security checking.

The following examples illustrate these coding requirements. In this example, an in-line table of the transactions in the VERBS parameter will be generated (the order and placement of BTVERB macros in the BTVRBTB is irrelevant). Figure 10-2 summarizes use of the SECVERBS and BTVERB macros.

Example 1: SECVERBS macro where Front End does exist:

```
            SECVERBS VERBS=(MLER,DLVE,INVE,RPTE,RDEQ,LWRE,TFQZ,      X
                  GRTE,BRNI),TABLE=YES

-------------------------------------------------------------------------

BTVRBTB     CSECT
              .
              .
              .
              .
              .
            BTVERB  VERB=BRNI,...
            BTVERB  VERB=DLVE,...
            BTVERB  VERB=MLER,...
            BTVERB  VERB=INVE,...
            BTVERB  VERB=RPTE,...
              .
              .
            BTVERB  VERB=RDEQ,...
            BTVERB  VERB=LWRE,...
            BTVERB  VERB=GRTE,...
            BTVERB  VERB=TFQZ,...
              .
              .
            PMISTOP
            END
```

Example 2: SECVERBS macro where Front End does not exist:

```
            SECVERBS VERBS=(MLER,DLVE,INVE,RPTE,RDEQ,LWRE,TFQZ,      X
                  GRTE,BRNI),TABLE=YES
```

SECVERBS	BTVERB	Result
VERBS=(list) TABLE=YES	Front End does not exist (Test Mode).	The list of transactions that will be in the VERBS parameter of subsequent STATION macros will be under transaction security check for the specified stations and will be generated in main storage.
VERBS=(list) TABLE=YES	Front End exists. Order of BTVERB macros does not matter.	The list of transactions that will be in the VERBS parameter of subsequent STATION macros will be under transaction security check for the specified stations and will be generated in main storage.
VERBS=(list) TABLE=NO	Front End exists. Each transaction in the SECVERBS VERBS parameter must have a BTVERB macro in the same order as the transactions appear in the SECVERBS macro, and preceding all other BTVERB macros in BTVRBTB.	The set of transaction-ids under security will be the set specified in the VERBS parameter of the SECVERBS macro, but will be listed in the BTVRBTB.

Figure 10-2. Summary and Use of SECVERBS and BTVERB Macros

10.5.4 STATION Macro/UNIVER and OPER Parameters

This is a STATION macro that, if used, has only two parameter entries, UNIVER and OPER. It is important to remember that not all stations need be under the sign-on/sign-off security option; for each station under terminal security, there is a set of operator codes associated with the station, which will be the only operator codes allowed to sign on at that station. However, if the user chooses, he may specify in the first STATION macro a set of universal operator codes which will be allowed to sign on to all terminals under the sign-on/sign-off option. If used, the macro appears as:

```
STATION     UNIVER=YES,                                          X
            OPER=(all codes for universal entry)
```

This STATION macro must precede all other STATION macros in PMISTATB, and only one STATION macro specifying UNIVER=YES is allowed to appear. It has no other parameters except UNIVER and OPER.

10.5.5 Other STATION Macro Parameters in PMISTATB

The STATION macro contains parameters for both the sign-on/sign-off security and transaction security options. Only one STATION macro per terminal is allowed, whether one or both types of security are in effect. Only those operands of the STATION macro pertaining to security are described here; consult Basic System Macros for coding details.

The OPER parameter indicates whether or not sign-on/sign-off security is required at the terminal. An absence of assigned operator codes indicates sign-on/sign-off security is not required. OPER specifies, in a sublist, the operator security codes to be considered as the only operator codes permitted entry at the terminal (unless overridden by a universal STATION macro as described above).

The RBN parameter provides a pointer to the relative location on a SEC000 file at which the subject terminal's security codes can be located. If all the associated security codes are to be core-resident, this parameter is not meaningful. However, if all the codes are to be located within the SEC000 file, then the RBN value is the last five digits of the member-name used to place the entry in the file (see Section 10.5.7).

The MAXSIGN parameter specifies the maximum number of times an operator can reenter an Intercomm SIGN transaction after failing in the attempt to pass sign-on/sign-off security for the terminal. Failure to sign on will be recorded at the Intercomm control terminal, and failure to sign on within the specified number of attempts will result in an immediate terminal down condition, with notification again sent to the Intercomm control terminal. An Intercomm TPUP (or STLU) transaction will be required to place the terminal on-line again.

The AUTOFF parameter specifies whether or not the terminal is to use the automatic sign-off feature of sign-on/sign-off. This parameter is meaningful only if security codes have been assigned to the OPER parameter. Code YES to use this feature, NO to bypass this feature. If YES is coded, the automatic sign-off duration interval is provided via either the TIME parameter of the STATION macro or the SPALIST macro, SGNTIME parameter. The default code is YES.

The TIME parameter indicates whether or not the otherwise default sign-off duration interval specified by the SPALIST macro, SGNTIME parameter, is to be overridden. This parameter is meaningful only if AUTOFF=YES has been specified. A zero code indicates that the SGNTIME interval is not to be overridden. A nonzero code indicates it is to be overridden and specifies, in minutes, the overriding sign-off interval, that is, the specific duration the subject terminal may retain a signed-on security clearance before that clearance is to be automatically revoked.

The VERBS parameter indicates whether or not transaction-id security is required at the terminal. The transactions listed in this parameter must come from the VERBS parameter in SECVERBS in the same order as in the SECVERBS list. If more than one transaction-id is to be specified, they must be coded as a parameter sublist. An absence of VERBS indicates that transaction-id security is not required.

10.5.6 Definition of Range of Verbs per Terminal for Transaction Security

The SWON and SWOF transactions can be used only on the transactions within the transaction security range of a terminal, which is defined below.

Consider the transactions in the VERBS list of the SECVERBS macro as being numbered sequentially, starting at 1. For a particular terminal, find the corresponding lowest number verb in the VERBS list of its STATION macro. Let the number of that verb be L. L is divided by 8, yielding a quotient (m) and remainder (q) (that is, q is less than 8):

$$L=8m+q$$

However, if L is an exact multiple of 8, then q=8, and m=m-1 (m=0 if L=8). The lowest number verb in the range for that terminal is then:

$r_1=L-q+1$ (for the first through eighth verbs, $r_1=1$)

Now take the corresponding highest number verb in the VERBS list of the same terminal's STATION macro and call it H. H is divided by 8 yielding a quotient (m) plus a remainder (q), that is,

$H=8m+q$

The highest number verb in the range for that terminal is then:

$r_2=8(m+1)$ (if H is an exact multiple of 8, $r_2=H$)

In other words, all verbs from the SECVERBS list with numbers equal to or greater than r_1, or less than or equal to r_2, can be acted upon by the SWON and SWOF transactions, for that terminal. Furthermore, no other secured verbs outside the range may be entered at that terminal.

The reason for all this is that a bit string is generated for the secured verbs list created by SECVERBS. The string consists of a bit for each verb, but the bits are grouped in units of 8. A corresponding bit string is also generated for each terminal, with a bit set on for each verb defined on the STATION macro. The bits are grouped in corresponding units of 8, so that even if a verb is in the middle of a SECVERBS verbs list unit, the whole unit from that list is included.

If the user wishes to simplify operating instructions, he can pretend that the range for the station consists only of those verbs falling between and including the lowest number transaction listed in the STATION macro and the highest number listed in the STATION macro. But the user must be careful to remember that the actual range for the terminal may be larger.

Following is an example of determining a range for a terminal:

```
SECVERBS    VERBS=(TBYV,CLYE,NRMY,LYRE,JALY,ALLI,TPQR,LFTY,          X
                   SNPQ,LLNO,FGRS,KDYO,LPQR,ATST,BSST,NYCE,          X
                   PLAU,PTER,FLFS,JWSP,JQRL,JMNO,FLOP,RWYE),         X
            TABLE=YES

STATION     TERM=(NYC01),                                            X
              .                                                      X
              .                                                      X
              .                                                      X
            VERBS=(KDYO,ATST,PTER)
```

The verbs in the SECVERBS list can be considered as consecutively numbered from 1-24, that is, TBYV is the first verb, RWYE is the 24th verb. The verbs in the VERBS parameter for the STATION macro for NYC01 terminal are:

KDY0 - 12th verb from SECVERBS
ATST - 14th verb from SECVERBS
PTER - 18th verb from SECVERBS

Thus, the lowest number verb on the NYC01 STATION macro is the 12th verb listed in SECVERBS, KDY0:

$L = 12$ $\qquad r_1 = 12-q+1$
$L = 8m+q$ $\qquad r_1 = 12-4+1$
$12 = 8*1+4$ $\qquad r_1 = 9$

The lowest number verb from SECVERBS possible in the range of NYC01 is the ninth verb from SECVERBS, namely SNPQ.

The highest number verb appearing in the STATION macro is the 18th verb from SECVERBS, PTER:

$H = 8m+q$ $\qquad r_2 = 8(m+1)$
$18 = 8*2+2$ $\qquad r_2 = 8(2+1)=24$

The highest verb possible in the range of NYC01 is verb number 24 from SECVERBS, or RWYE.

Thus, an operator from a terminal can use the SWON instruction to activate transaction security for NYC01 on the verbs SNPQ, LLNO, FGRS, LPQR, BSST, NYCE, PLAU, FLFS, JWSP, JQRL, JMNO, FLOP, RWYE, even though they are not in the VERBS parameter of the STATION macro for NYC01.

10.5.7 Loading Operator Codes on Disk for Station Security Option

To have operator codes on disk, the user must create a symbolic library, that is, SYMSEC with members consisting of the operator codes under security for each terminal; each STATION macro is an entry in the library.

Members of SYMSEC appear as:

```
SECxxxxx    CSECT
            GENSEC      OPER=DISK
            STATION     OPER=(list)
            END
```

The member names in SYMSEC must be SECxxxxx, where xxxxx is all numeric. All xxxxxs must begin at 00001 and be in sequential order with no five-digit omissions.

The order in which each station appears in the library SYMSEC must correspond to its RBN number, that is, if the STATION macro for terminal NYC01 has RBN=00005, it must be the fifth entry in the library (member-name SEC00005).

For each set of operator codes on disk listed in SYMSEC, it is necessary to assemble and linkedit the member with the same name into a load module library, that is, MODSEC. Create the SEC000 file by executing the off-line File Load Utility PMIEXLD against MODSEC. (See Chapter 12.)

10.6 IMPLEMENTATION OF USER-WRITTEN SECURITY ROUTINES

Under Basic Security, the user can supply his own security checks at the subsystem level in addition to or instead of the sign-on/sign-off and transaction security checks supplied by Intercomm. If this option is chosen, he must perform the coding steps described below.

10.6.1 Coding Security Subroutines

Before the Subsystem Controller passes a message to a subsystem, it checks the Subsystem Control Table entry for the subsystem to determine whether incoming messages for this particular module are to be passed through a security subroutine. If a security routine has been provided for the subsystem, the message will be passed to this subroutine before being passed to the actual application program. When called, the security subroutine is passed, via register 1, the address of a parameter list consisting of:

1. Address of message
2. Address of System Parameter Area
3. Address of Subsystem Control Table entry

The user-coded security routine will determine whether this message is or is not to be passed on to the application program. If the message passes the security check, the security routine will return to the Subsystem Controller after placing a return code of 0 in register 15.

If the message does not pass the security check, a return code of 16 or greater should be placed in register 15. Under this condition, the user must provide coding within his security routine to generate and output any required error message.

Also, in the case where the message does not pass the security check, the security routine must free the incoming message area if the subsystem was coded in Assembler Language. If the subsystem was coded in a high-level language, the Subsystem Controller will free the message when cancelled by the security routine. To determine language type, use the Dsect SCTLISTC (COPY member) for the subsystem SCT entry.

The user-coded subroutines must be resident and use standard linkage conventions.

10.6.2 SPALIST Macro Parameter

USERSEC=YES must be coded on SPALIST to indicate user security routines are to be honored.

10.6.3 SYCTTBL Macro Parameter

If a security routine is to be provided for a particular subsystem, the parameter SECU of the SYCTTBL macro must be coded. Otherwise, the default value of 0 (no security routine) will be placed in the Subsystem Control Table entry. Any number from 1 to 63 may be coded in the SECU parameter. This number will be used as an index to access the actual address of the security routine, found in a table of user security routine VCONs coded in a Csect named SECURITY.

10.6.4 Security Table

The SECURITY Csect must contain address constants pointing to each user security routine. The first VCON in this table is the address of security routine number 1; the second VCON is the address of routine 2; etc. No PMISTOP macro is needed at the end of this table. User security routines may have any name which does not conflict with system module names (see operator disk member names above, and linkedit below).

```
SECURITY        CSECT
                DC    V(SEC01)
                DC    V(SEC02)
                 .
                 .
                 .
                END
```

10.6.5 Linkedit Requirements

The proper include cards for Basic Security logic modules are produced when the user specifies SECUR=YES when assembling the ICOMLINK macro to generate an Intercomm linkedit deck. The modules are PMIAUTOF, PMIHEADR, PMISIGN, SECURE00, SECURE01 and SECURE02. User-supplied terminal security routines must have member and entry point names USRSGNON and USRSGNOF as previously discussed. User-supplied subsystem security routines must be specially included, using the names in the VCONs coded in the Security Table, which must also be included via a user-coded INCLUDE statement.

10.7 Multiregion Intercomm Considerations

If SOSO=YES is coded for any SYCTTBL in the Intercomm system, then, when executing under a Multiregion Intercomm system, terminal (sign-on/sign-off) security only applies to the single region where the PMISIGN subsystem is resident. This restriction may be overcome by using RAP processing (operator signs on only to the region to which the terminal is locked). Alternatively, all SYCTTBLs for which SOSO=YES is required can be grouped in one SCT in one region (the same region where PMISIGN processes the SIGN command).

Transaction security and user-coded subsystem security exit routines are processed just before a message is passed to a subsystem and are therefore not affected by execution in a Multiregion system. Table coding and linkedit requirements described in the above sections for these two security types must be present in each region for which either type of security is desired.

Chapter 11

SYSTEM TUNING TECHNIQUES

11.1 INTRODUCTION

One of the major areas of concern in any on-line system environment is that of system tuning: those procedures involved in optimizing system performance from the points of view of response time, throughput and resource utilization.

This chapter presents techniques for system tuning from the following points of view:

- System tuning and performance evaluation
- System statistics reports and display commands
- Tracing a message on the log
- Factors affecting performance
- The Fine Tuner commands
- Response time considerations
- MVS Tuning recommendations
- Debugging and tracing facilities

Debugging an on-line system is a task ranging in complexity from simple errors in application program code to virtually random errors in the interaction of program logic, due to time-dependent combinations of message processing. Debugging techniques are described in the Intercomm Messages and Codes manual.

11.2 SYSTEM TUNING AND PERFORMANCE EVALUATION

System tuning and subsequent performance evaluation in the on-line system environment involves consideration of the following:

1. Transaction response time, typically measured as elapsed time from request for entry of a message from the terminal until the first character of response is received
2. Message throughput, typically measured in terms of messages per hour
3. CPU utilization, derived from operating system accounting statistics

Chapter 12

OFF-LINE UTILITIES

12.1 INTRODUCTION

The utility programs discussed in this chapter are provided with the Intercomm system to assist the user with operations common to the on-line environment and/or to provide data set creation for Intercomm facilities.

12.2 LOG PROCESSING PROGRAMS

At the completion of execution of an Intercomm job, one of the following programs may be used to process the system log (INTERLOG) for further analysis of message processing:

- LOGPRINT--formatted printout of log
- LOGANAL--log sort/analysis

12.3 INTERCOMM LOG DISPLAY (LOGPRINT)

An off-line utility program may be used to print the Intercomm system log when execution of Intercomm has terminated. LOGPRINT contains routines that select specified records for printing. Selection of records can be by date, time, terminal, subsystem code, log code, etc. Selection criteria are established by a SYSIN file. The default is to print all INTERLOG entries. The first page of the report contains only the title line and the parameter selection statements, or, if none, the legend NO CARDS FOUND.

Figure 12-1 illustrates a sample output page from this program, where the circled notation indicates:

(A) Each page contains a title line defining the standard message header field names.

(B) Each message is printed in the following format:

(1) message header, spaced as per the title line heading

(2) message text, 32 characters per line

(a) offset (in decimal) relative to zero within text

(b) hexadecimal format

(c) EBCDIC format

```
DATE 82.293    TIME 20.58.55        **** I N T E R C O M M   L O G   D I S P L A Y ****                    PAGE     4

MSGLEN  THREAD   QPR   RSC      SSC          MMN     DATE        TIME         TID    PID     CO   USR    RMN  LOG BLK  VMI
------------------------------------------------------------------------------------------------------------------------
    42      1    02  ../0000  .U/00E4         15    82.293    20.17.4176    TEST1  .....  0001   00      2   F3  00   50
------------------------------------------------------------------------------------------------------------------------
    56      0    02  ../0000  ../0000          0    82.293    20.17.4610    TEST1  .....  0000   00      3   F1  00   00
000000   C6C8E2E3 6DC9D5E3 C5D9D3D6 C726                                              *FHST,INTERLOG.                    *
------------------------------------------------------------------------------------------------------------------------
    56      0    F2  .R/00D9  ../0009         16    82.293    20.17.4610    TEST1  .....  0001   00      3   01  00   FF
000000   C6C8E2E3 6DC9D5E3 C5D9D3D6 C726                                              *FHST,INTERLOG.                    *
------------------------------------------------------------------------------------------------------------------------
    42      1    F2  .R/00D9  ../0000         16    82.293    20.17.4761    TEST1  .....  0000   00      3   30  00   FF
------------------------------------------------------------------------------------------------------------------------
   137      1    02  .U/00E4  .R/00D9         17    82.293    20.17.4771    TEST1  .....  0001   00      3   01  00   50
000000   FF02002F 14010001 0901C9D5 E3C5D9D3    D6C70209 01404040 404040F1 F2030901  *..........INTERLOG...      12...*
000032   40404040 404040F4 04090140 40404040    40404005 09014040 40404040 40400609  *       4...      ...       ...*
000064   01404040 40404040 40070901 40404040    404040F4 08090140 404040F8 4BF3F3    *.         ...       4...    0.33 *
------------------------------------------------------------------------------------------------------------------------
    88      1    F2  .R/00D9  ../0000         16    82.293    20.17.4775    TEST1  .....  004E   00      3   FA  00   FF
000000   00000000 00000000 00110000 02400000    009B0000 00010000 00000000 00010000  *.............. ................*
000032   00000000 00000000 00000000 0000                                             *..............                    *
------------------------------------------------------------------------------------------------------------------------
    42      1    02  .U/00E4  .R/00D9         17    82.293    20.17.4775    TEST1  .....  0000   00      3   30  00   50
------------------------------------------------------------------------------------------------------------------------
   252      1    02  ../0000  .U/00E4         18    82.293    20.17.4779    TEST1  .....  0001   00      3   F2  00   50
000000   40404040 40404040 40404040 40404040    40404040 C6C9D3C5 40C8C1D5 C4D3C5D9  *                        FILE HANDLER*
000032   40E2E3C1 E3C9E2E3 C9C3E240 D9C5D7D6    D9E315C6 C9D3C540 D5C1D4C5 404040F2  * STATISTICS REPORT.FILE NAME    S*
000064   C5D3C5C3 E3404040 40C1C3C3 C5E2E240    40404040 40404040 40404040 40404040  *ELECT    ACCESS                  *
000096   40404040 40404040 40404040 40404040    4040F3D6 E3C1D340 40404040 C1E5C5D9  *                      TOTAL      AVER*
000128   C1C7C5C9 D5E3C5D9 D3D6C740 40404040    404040F1 F2404040 40404040 4040F440  *AGEINTERLOG          12         4 *
000160   40404040 40404040 48404040 40404040    40404040 40404040 40404040 40404040  *                                 *
000192   40404040 4040F440 40404040 40F04BF3    F337                                 *      4      0.33.                *
------------------------------------------------------------------------------------------------------------------------
    88      1    02  .U/00E4  .R/00D9         17    82.293    20.17.4779    TEST1  .....  0000   00      3   FA  00   50
000000   00000000 00010000 00060000 1FC00000    001A0000 00010000 00010000 00000000  *................................*
000032   00000000 00000000 00000000 0000                                             *..............                    *
------------------------------------------------------------------------------------------------------------------------
    42      1    02  ../0000  .U/00E4         18    82.293    20.17.5199    TEST1  .....  0001   00      3   F3  00   50
------------------------------------------------------------------------------------------------------------------------
    47      0    02  ../0000  ../0000          0    82.293    20.17.5697    CNT01  .....  0000   00      4   F1  00   00
000000   D5D9C3C4 26                                                                 *NRCD.                             *
------------------------------------------------------------------------------------------------------------------------
    47      0    F2  .J/00D1  ../0000         19    82.293    20.17.5700    CNT01  .....  0001   00      4   01  00   FF
000000   D5D9C3C4 26                                                                 *NRCD.                             *
------------------------------------------------------------------------------------------------------------------------
    42      1    F2  .J/00D1  ../0000         19    82.293    20.17.5700    CNT01  .....  0000   00      4   30  00   FF
------------------------------------------------------------------------------------------------------------------------
   108      1    02  .U/00E4  ../0000         20    82.293    20.17.5700    TOALL  .....  0001   00      0   01  00   50
000000   FF02002D 013C5C5C 5C40C7D6 D6C440C5    E5C5D5C9 D5C7405C 5C5C4040 C9D5E3C5  *.........*** GOOD EVENING *** INTE*
000032   D9C3D6D4 D440C9E2 40C3D3D6 E2C5C47A    404040F1 F96CF2F0 60F8F240 40F2F04B  *RCOMM IS CLOSED:   10-20-82  20.*
000064   F1F7                                                                        *17                                *
------------------------------------------------------------------------------------------------------------------------
    88      1    F2  .J/00D1  ../0000         19    82.293    20.17.5700    CNT01  .....  00C0   00      4   FA  00   FF
000000   00000000 00000000 00010000 02F00000    00060000 00010000 00000000 00000000  *............0.................*

MSGLEN  THREAD   QPR   RSC      SSC          MMN     DATE        TIME         TID    PID     CO   USR    RMN  LOG BLK  VMI
```

Figure 12-1. Sample Output Page from LOGPRINT Utility

The JCL required for execution is shown in Figure 12-2.

```
//          JOB
//JOBLIB    DD DSN=INT.MODxxx,DISP=SHR
//          EXEC PGM=LOGPRINT[,PARM=nnn,REGION=rrrK]
//INTERLOG  DD DSN=_____,
//             DISP=OLD,
//             DCB=BLKSIZE=bbbbb
//SYSPRINT  DD SYSOUT=_____,
//             DCB=(DSORG=PS,BLKSIZE=(multiple of 121))
//SYSUDUMP  DD SYSOUT=A
//SYSIN     DD * or DUMMY,DCB=BLKSIZE=80
```

xxx	is the library containing the LOGPRINT load module. This will be REL, unless using File Selection in which case the LOGPRINT load module must be relinked and placed on MODLIB.
nnn	is the number of lines per page. The default is 58.
rrrK	is at least twice the block size plus 15.
bbbbb	is the length of the largest block on the data set. (This parameter may be omitted when using standard label volumes.)

Figure 12-2. JCL for LOGPRINT Execution

12.3.1 Description and Function of Control Records (SYSIN)

All of the selection records are fixed-format, fixed-position. There is no validation of data. The control field begins in column 1 and indicates the following selective options:

- Date Selection

  Select records within a date range.

```
cc   cc    cc
1    6     12

DATEbyyddd byyddd
```

ending year and date

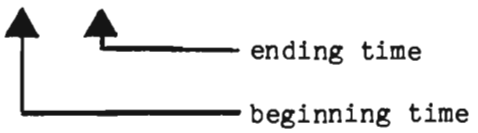
beginning year and date

specify date selection

- Time Selection

  Select records within a time range. If both time and date are entered, the range is from the beginning time and date to the ending time and date.

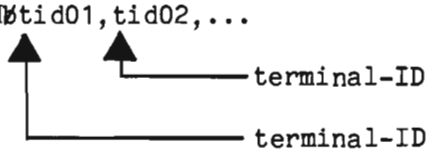
  ```
  cc   cc   cc
  1    6    11

  TIMEbhhmmbhhmm
  ```

- Terminal Selection

  Select records for specified terminals. Records from several terminals can be selected, up to a maximum of 41 terminals.

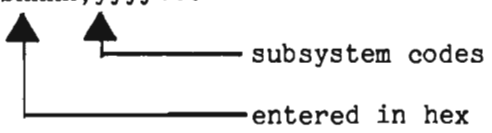
  ```
  cc   cc    cc
  1    6     12

  TERMbtid01,tid02,...
  ```

- Subsystem Selection

  Select records with specified sending or receiving subsystem codes. Any number of subsystem codes can be entered on multiple entry records, up to a maximum total of 41 codes.

  ```
  cc   cc   cc
  1    6    11

  SSCbbxxxx,yyyy...
  ```

- Log Code Selection

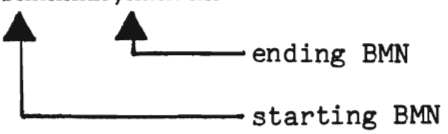
  Select records with specified log codes (in hex). Any number of log codes can be entered on multiple entry records, up to a maximum total of 41.

```
cc    cc cc
1     6  9

LGIDØxx,yy,...
```

- BMN Selection

  Select records within a BMN range.

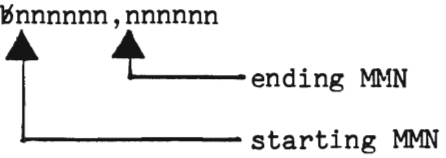
```
cc    cc      cc
1     6       13

BMNØØnnnnnn,nnnnnn
```



- MMN Selection

  Select records within an MMN range.

```
cc    cc      cc
1     6       13

MMNØØnnnnnn,nnnnnn
```



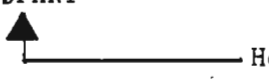
- Print Selection

Print only message header or message header and 1 line of text.

```
cc   cc
1    6

PRNTbHEAD
```

Header only

```
cc   cc
1    6

PRNTbPART
```



Header and 1 text line

- File Selection

Select File Recovery log records. To use this option, the LOGPRINT module must first be linkedited as follows:

```
//   EXEC LKEDP,Q=LIB,LMOD=LOGPRINT
//LKED.SYSIN  DD  *
     INCLUDE SYSLIB(LOGPRINT)
     INCLUDE SYSLIB(IXFSNAPL)
```

Note that the execution JCL STEPLIB must specify INT.MODLIB.

The format of the File Selection control record is as follows:

```
cc   cc
1    6

FILEbddnamebddname...ddname
```

where ddname is the ddname of a file which is to be selected. ddnames must be separated by one or more blanks. Any number of files may be selected, up to a maximum of 41. If TIME and/or DATE selection is also specified (see above), then only File Recovery records within that TIME/DATE range will be printed. If no ddnames are specified on the FILE statement, then all File Recovery log records are printed.

The format in which File Recovery records are printed differs from messages. An example of this format may be found in the File Recovery Users Guide, "Sample IXFSNAPL Output".

The following are examples of selective log printing.

Example: Print only records from 12:39 through 12:45 with subsystem codes 00E4 or 00D3.

```
//SYSIN DD *
TIMEb1239b1245
SSCbb00E4,00D3
/*
```

Example: Print only records from 82219 through 82221 for terminals CNT01 and CRT03 with subsystem codes 0805 or C912 and log code 01.

```
//SYSIN DD *
TERMbCNT01
DATEb82219b82221
SSCbb0805,C912
TERMbCRT03
LGIDb01
/*
```

Example: Print only File Recovery records for ddnames ISAMX and BDAMY from 15:21 on 82218 through 07:00 on 82219.

```
//SYSIN DD *
TIMEb1521,0700
DATEb82218,82219
FILEbISAMXbBDAMY
/*
```

12.4 LOG ANALYSIS PROGRAM (LOGANAL)

The log analysis program operates on Intercomm log data sets from one or more successive executions of Intercomm to produce traffic histograms or response time reports. The PARM field on the EXEC statement invoking LOGANAL indicates reporting options.

12.4.1 Traffic Histograms

Traffic histograms portray the number of inputs during each half-hour interval. Depending upon the ANALYZE option specified at LOGANAL invocation, histograms are produced by terminal (TERM option), the entire run (TOTAL option) and the parent subsystem (SUBSYS option) or verb (VERB option). Note that, if the VERB option is specified, the SUBSYS option is ignored.

The report for a terminal or the run consists of two parts: a summary of inputs for this terminal or the run for each subsystem (SUBSYS option) or verb (VERB option), and an input traffic histogram. The report for a parent subsystem or verb consists of a traffic histogram. (A parent subsystem is the subsystem that processes the input message from the outside world or from the control region if analyzing a satellite region log.) Figure 12-3 illustrates a sample output.

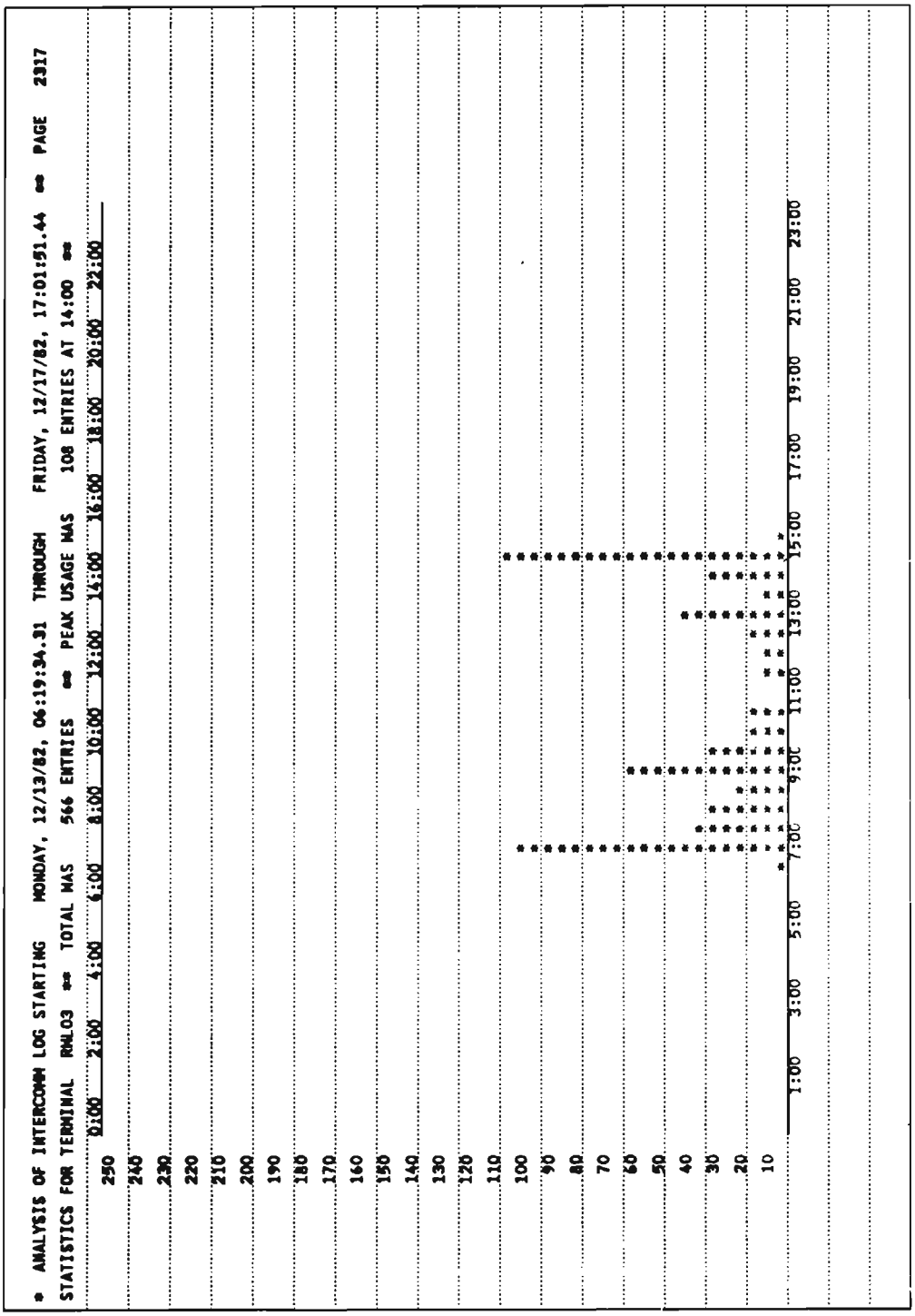


Figure 12-3. Sample Histogram for a Terminal (Page 1 of 2)

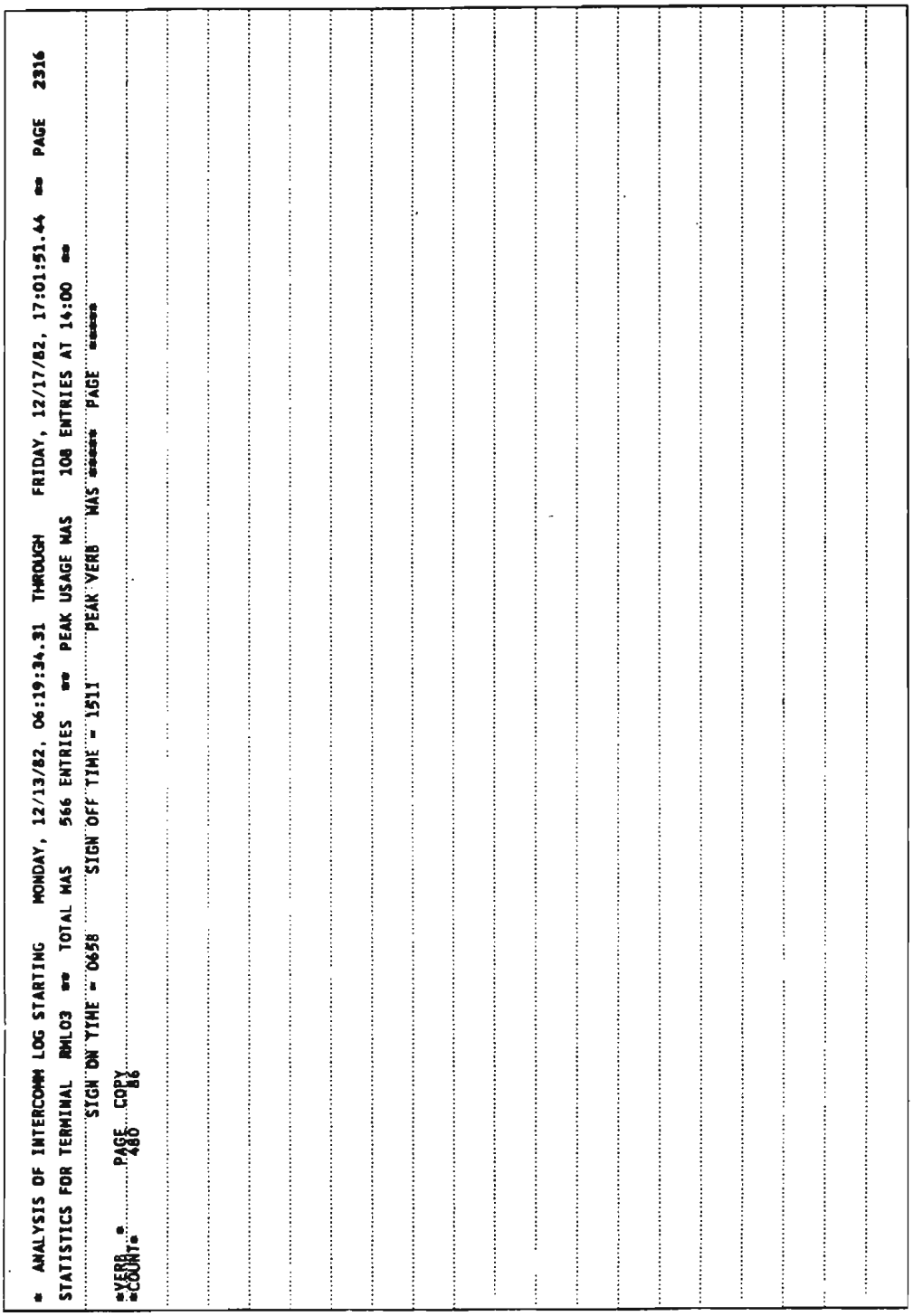


Figure 12-3. Sample Histogram for a Terminal (Page 2 of 2)

12.4.2 Response Time Reports

Response Time Reports generate analysis of input message response times, and message queue and process times. Reports can be displayed for the entire run (TOTAL option), by region (REGION option), by terminal (TERM option), by parent subsystem (SUBSYS option), and by verb (VERB option), depending upon the ANALYZE options specified when LOGANAL is invoked.

ANALYZE and OUTPUT options are used to control response time reporting. ANALYZE controls the breakdown level. OUTPUT is used to capture the input log record data used by LOGANAL. See Section 12.4.4 for permissible values.

All reports display hourly and 24-hour totals of response time. Response time is defined as the time elapsed between receipt of input message from the outside world (or from the control region for Multiregion Intercomm) and the transmission of the first message back to the input terminal (or control region). The report shows response time by interval with maximum and mean times and standard deviations, counts of messages with no response to the input terminal, and messages lost due to queue full, cancelled or flushed conditions; additionally, parent subsystem and verb reports display queue and process time statistics. A line-by-line explanation of a sample page of a report is described in the following subsections.

Report Terminology

Terms used in these reports are:

- parent subsystem--the subsystem receiving the message from the outside world or from the control region.
- child subsystem--a subsystem initiated by messages from a parent subsystem or other child subsystem, such that the parentage can be traced back to a message from the outside world.
- Front End--the Intercomm Front End, which receives messages from a parent or child subsystem.
- transaction--the collection of messages associated with the same input message. A transaction is created by a message to a parent subsystem, and includes that message, the messages to the parent subsystem's children, messages to its children's children, etc.

Line-by-Line Report Analysis

Figure 12-4 is a sample page of a reponse time report. Numbered lines in the figure are explained below:

Line 1

This line gives the earliest and latest dates and times encountered on the log.

Line 2

This line indicates the breakdown level of the report. One response time report (hourly totals and 24-hour total) is produced for each breakdown level, as specified by the ANALYZE option.

The rightmost legend indicates the breakdown level (VERB SIGN in this example).

Region breakdown is displayed only for Multiregion log files. Control region log file reports will indicate REGION CONTROL for messages processed within the control region and REGION rrrrrrrr for messages sent to satellite region rrrrrrrr. Satellite region log file reports will indicate the satellite region.

The parent subsystem is given in EBCDIC and hexadecimal. The verb is given in EBCDIC, except when it is not available. The verb may not be available if it specifies edit-before-queuing. The user may code a table (LOGVRBTB) to define subsystem/verb/VMI correspondence to LOGANAL. If an edit-before-queuing verb is in this table, the verb is displayed in EBCDIC. If this type of verb is not in the table, or no table is coded, the VERB legend gives the VMI in hexadecimal (VERB 'vv', where vv is the VMI).

Line 3

This line is the title of the report.

Line 4

This line displays the headings for the next line. The right-hand columns of the line specify the total number of response time statistics available (TOTAL), and maximum (MAX TIME), mean (MEAN), and standard deviation (STD DEV) of the response times.

Lines 5, 6

These lines give the response times for a one-hour period.

The HOUR STARTING column indicates the hour in which the transaction started.

```
 1  *  ANALYSIS OF INTERCOMM LOG STARTING    MONDAY, 12/13/82, 06:19:34.31  THROUGH    FRIDAY, 12/17/82, 17:01:51.44  **  PAGE   2326
 2  REGION CONTROL      TERMINAL RWL04     PARENT SUBSYSTEM K /D20D     VERB '01'
 3                                  RESPONSE TIME IN SECONDS FROM INPUT MSG TO FIRST OUTPUT MSG
 4  HOUR                                                                                                        MAX    MEAN    STD
    STARTING        0-1     1-2     2-3     3-4     4-5     5-6     6-8     8-10   10-15   15-30  OVER 30   TOTAL   TIME   TIME    DEV

 5     9   NUM         1       1                                                                              2      1.1    0.7     0.5
 6         PCEN     50.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0
 7         NUM TRANS       2      NO RESPONSE      0     MSGS LOST:  Q FULL     0   CANCELLED     0   FLUSHED     0
 8         BREAKDOWN             NUM MSG                 QUEUE TIME                             PROCESS TIME
 9         PARENT    K /D20D        2       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.5  MEAN    0.2  STD    0.3
10         OFFSPRING  U/00E4        2       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.0  MEAN    0.0  STD    0.0
11         FRONT END                2       MAX   0.8  MEAN     0.5  STD    0.3

    12     NUM         2                                                                                      2      1.0    0.7     0.5
           PCEN    100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0
           NUM TRANS       2      NO RESPONSE      0     MSGS LOST:  Q FULL     0   CANCELLED     0   FLUSHED     0
           BREAKDOWN             NUM MSG                 QUEUE TIME                             PROCESS TIME
           PARENT    K /D20D        2       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.3  MEAN    0.2  STD    0.2
           OFFSPRING  U/00E4        2       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.0  MEAN    0.0  STD    0.0
           FRONT END                2       MAX   0.8  MEAN     0.6  STD    0.4

    14     NUM         1       1                                                                              2      1.1    0.7     0.5
           PCEN     50.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0
           NUM TRANS       2      NO RESPONSE      0     MSGS LOST:  Q FULL     0   CANCELLED     0   FLUSHED     0
           BREAKDOWN             NUM MSG                 QUEUE TIME                             PROCESS TIME
           PARENT    K /D20D        2       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.4  MEAN    0.2  STD    0.3
           OFFSPRING  U/00E4        2       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.0  MEAN    0.0  STD    0.0
           FRONT END                2       MAX   0.8  MEAN     0.6  STD    0.4

12  TOTAL  NUM         4       2                                                                              6      1.1    0.7     0.4
13         PCEN     66.7   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0
14         NUM TRANS       6      NO RESPONSE      0     MSGS LOST:  Q FULL     0   CANCELLED     0   FLUSHED     0
15         BREAKDOWN             NUM MSG                 QUEUE TIME                             PROCESS TIME
16         PARENT    K /D20D        6       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.5  MEAN    0.2  STD    0.2
17         OFFSPRING  U/00E4        6       MAX   0.0  MEAN     0.0  STD    0.0   MAX    0.0  MEAN    0.0  STD    0.0
18         FRONT END                6       MAX   0.8  MEAN     0.6  STD    0.3
```

Figure 12-4. Sample Response Time Analysis

The columns headed 0-1, 1-2,... head statistics about transactions with response times in the interval. For example, for interval 1-2, the response time is from 1 to 2 seconds, inclusive.

NUM gives the number of transactions that responded in the interval.

The second line (PCEN) indicates the cumulative percentage of responded transactions. For example, for interval 1-2, the percentage of transactions with response time 0 to 2 seconds, inclusive.

The response time is defined differently for single or control region Intercomm and for satellite region Intercomm.

- For single region and control region Intercomm, response time is the elapsed time between receipt on an input message from the outside world (log record with log code X'01' or X'C1') and the transmission of the first output message back to the input terminal (log record with log code X'F3').

- For satellite region Intercomm, response time is the elapsed time between receipt of a message from the control region (log record with log code X'01') and the transmission of the first message back to the control region by the Multiregion output subsystem (log record with log code X'FA').

NOTE: Response time is defined only for output messages back to the original input terminal. If there is no output, or if it is associated with a terminal other than the original input terminal, the transaction will be considered "no response."

Line 7

This line shows other transaction-related statistics.

NUM TRANS is the total number of transactions within the period, including those that have no response time due to design or error conditions.

NO RESPONSE counts are totals of logged error conditions that may have occurred to any of the messages of a transaction. Possible error conditions are:

- Q FULL--the number of subsystem messages lost due to a queue full condition (log record with log code X'FC').

- CANCELLED--the number of messages lost due to a program error or time-out (log record with log code X'FD').

- FLUSHED--the number of messages flushed by:

  -- The Retriever (segment input retrieved by the GETSEG service routine--log record with log code X'FE')

  -- Message Collection for an invalid destination subsystem code (log record with log code X'FB').

  -- Multiregion Queue Manager, because a stopped or inactive satellite region specified the flush option (log record with log code X'C3').

Lines 8-11

These lines are present only on parent subsystem and verb response time reports. They give statistics about the subsystems used in processing a particular transaction.

The (subsystem) BREAKDOWN entries are PARENT, OFFSPRING and FRONT END. Each breakdown entry (lines 9, 10, 11) gives the number of messages for the subsystem and queue and process times.

PARENT is always present.

OFFSPRING--one or more OFFSPRING, that is, child subsystem, entries may be present.

FRONT END will always be present in single region or control region Intercomm when response times are available. In satellite regions, an OFFSPRING entry for the Multiregion output subsystem will be present instead of FRONT END when response times are available. This offspring will show the subsystem code of the destination subsystem in the control region or other satellite region. If the destination is the Front End queueing routine, FESEND, the subsystem code will be X'00E9' (Z).

QUEUE TIME is the time elapsed between queuing of a message (log record with log code X'01', X'C1' or X'F2') and subsystem processing (log record with log code X'30') or transmission (log record with log codes X'C2' or X'F3').

PROCESS TIME, available only for subsystems executed within the region, is the time elapsed between start of processing (log record with log code X'30') and end of processing (log record with log code X'FA'). For queue and process times, maximum (MAX), mean (MEAN), and standard deviation (STD) times are given. This example illustrates the breakdown of a simple transaction. The parent subsystem X'00E2'(S) creates a message for the child (offspring) subsystem X'00D5'(N), which in turn creates a message to the Front End.

Lines 12-18
These lines are similar to lines 5-11. They detail response time and breakdown statistics for the hour 14:00 (2:00 PM).

Lines 19-25
These lines are similar in format to lines 5-11. They detail the 24-hour total statistics.

All times are in seconds and tenths of seconds. The standard deviation reflects the distribution of times about the mean. When it is too large or otherwise unable to be computed, a value of 9999.9 is displayed.

12.4.3 Installation of LOGANAL

Installation of LOGANAL requires the following steps:

1. Examine the LOGANAL generation parameters in the member LOGSETGB.
2. Optionally change LOGSETGB values, if necessary.
3. Optionally generate the LOGVRBTB, using the LOGVERB macro.
4. Create the LOGANAL Load Module.

12.4.3.1 LOGANAL Generation Parameters

The LOGANAL generation parameters are in the member LOGSETGB. Examine the parameters and determine the settings suitable to your installation's needs. The default values of these parameters were chosen to accommodate common requirements. The parameters described below are defined as globals in the member LOGDCLGB. The global settings are in the member LOGSETGB. The &MXSS, &MXMIPFT, &MXMIFAM and &MXDIFSS parameters control table capacities.

LOGSETGB settings are illustrated below:

```
         COPY   LOGDCLGB
         GBLA   &NBRETRN          NUM OF BYTES RETURNED TO OS BY LOGANE15
         GBLA   &MXMIPFT          MAX NUM OF MSGS IN PROG FOR A TRANSACTION
         GBLA   &MXSS             MAX NUM OF SUBSYS OR VERBS FOR LOGHIST
         GBLA   &MXMIFAM          MAX NUM OF MSG IN 'FAMILY'=PARENT &
                                  CHILDREN FROM ONE INPUT MSG
         GBLA   &MXDIFSS          MAX NUM OF DIFFERENT SUBSYS USED IN PROC
                                  ALL TYPE OF INPUT TO ONE PARENT SUBSYS
---------------------------------------------------------------------------
         COPY   LOGSETGB
&NBRETRN SETA   64*1024           SYNCSORT NEEDS 64K, SM1 NEEDS 4K
&MXMIPFT SETA   16                MSGS IN PROGRESS
&MXSS    SETA   100               MAX NUM OF SUBSYS (MULTIPLE OF 20)
&MXMIFAM SETA   16                MAX MSGS IN A 'FAMILY'
&MXDIFSS SETA   10                MAX NM OF DIF SUBSYS
```

&MXDIFSS

specifies the maximum number of different subsystems used in processing all types of input to one parent subsystem. The value is set at 10. The parent subsystem and the Front End count as two toward that limit. If the error message LA035I is issued, increase the value and reassemble LOGRESP and LOGRSRPT. Refer to LA035I for further information.

&MXMIFAM

specifies the maximum number of messages generated in a family (that is, parent and children) from one input message. The value is set at 16. If error message LA036I is issued, then increase the value and reassemble LOGRESP. This value should never be set higher than 254.

&MXMIPFT

specifies the maximum number of messages in progress for a transaction. A message in progress is one that has not been processed to completion; that is, LOGANAL encountered its first log entry but not its final log entry. The maximum value is set at 16. If error message LA046I is issued, increase the value and reassemble LOGANE15. This value should never be set higher than 76.

&MXSS

specifies the maximum number of subsystems or verbs for LOGHIST. There is one entry in the table for each verb in the input file (if the verb option of ANALYZE is used), or for each parent subsystem (if only the SUBSYS option of ANALYZE is used). The value is set at 100 (multiples of 20). If error message LA051I is issued, increase the value and reassemble LOGSSTAB and LOGHIST.

&NBRETRN

specifies the number of bytes returned to OS by LOGANE15. This parameter controls table allocation in the E15 exit routine invoked by the SORT program. It is set to handle the requirements of SYNCSORT, which is 64K. The IBM sort can use a smaller value of 4K; this reduces the minimum region size of LOGANAL to 160K. &NBRETRN is set at 64*1024. If there is a problem with the SORT program, or if a S804 abend occurs, increase the size and reassemble LOGANE15.

12.4.3.2 Changing LOGANAL Generation Parameters

To change LOGSETGB values, use the following JCL:

```
//* CHANGE PARAMETERS IN LOGSETGB
//  EXEC     LIBE,Q=LIB
./  CHANGE   NAME=LOGSETGB,LIST=ALL
******** PLACE REVISED PARAMETERS HERE ********
//* REASSEMBLE MODULES INDICATED IN DESCRIPTION OF PARAMETERS
//  EXEC     ASMPCL,Q=LIB,NAME=module1,LMOD=module1
//ASM.SYSIN  DD INT.SYMREL(module1),DISP=SHR
//* IF module1 NOT IN INT.SYMLIB ADD ABOVE STATEMENT
...
...   reassemble other modules as required
...
```

12.4.3.3 Generating the LOGVRBTB

If edit-before-queuing is used, the verb is not in the initial log record for a transaction. The user may code a table of subsystem code/VMI/verb combinations. This table is the member LOGVRBTB, which is coded using the LOGVERB macro. The macro format is:

```
(blank)  LOGVERB   SUBSYS=subsystem-code,
                   VMI=verb-message-identifier,
                   VERB=transaction-ID
```

The following JCL creates a sample LOGVRBTB:

```
//* REPLACE DUMMY LOGVRBTB IN INT.SYMREL WITH USER-CODED TABLE
//  EXEC     LIBE,Q=LIB
./  REPL     NAME=LOGVRBTB,LIST=ALL
./  NUMBER   INCR=1000,NEW1=1000
*   CODE LOGVERB MACROS TO RELATE SUBSYSTEM AND VMI COMBINATIONS
*   TO VERBS IF EDIT-BEFORE-QUEUING IS USED.
          LOGVERB SUBSYS=X'0102',VMI=X'01',VERB='VRB1'
          LOGVERB SUBSYS=X'0103',VMI=X'02',VERB='VRB2'
//* REASSEMBLE AND LINK LOGSSTAB WHICH COPIES NEW LOGVRBTB
//  EXEC     ASMPCL,Q=LIB,LMOD=LOGSSTAB
//ASM.SYSIN  DD  DSN=INT.SYMREL(LOGSSTAB),DISP=SHR
```

12.4.3.4 Creating the LOGANAL Load Module

If executing under an operating system other than MVS, reassemble LOGANAL to ensure that the correct version of the SPIE macro is used. The LOGANAL load module is created by executing the following JCL:

```
//LK   EXEC   LKEDP,Q=LIB,LMOD=LOGANAL
//LKED.SYSIN  DD *
              INCLUDE  SYSLIB(LOGANE15,LOGRESP,LOGRSRPT)
              INCLUDE  SYSLIB(LOGHIST,LOGSSTAB,JULIAND)
              INCLUDE  SYSLIB(LOGANAL)
              ENTRY    LOGANAL
/*
```

12.4.4 Execution of LOGANAL

Execution of LOGANAL is controlled by the EXEC statement PARM option coded as follows:

```
//  EXEC     PGM=LOGANAL,PARM='parm-options'
```

The 'parm-options' are coded within a set of single quotation marks as keyword parameters, each separated by a comma. The names of the execution parameters may be shortened. For example, M=yyddd is interpreted as MAXDATE=yyddd. Refer to parameter descriptions for specific abbreviations.

If no PARM field is specified, the parameter defaults used are equivalent to specifying the following:

```
PARM='HISTOGRAM=YES,RESPONSE=YES,
     ANALYZE=(TOTAL,REGION,SUBSYS,TERM,VERB),
     OUTPUT=NO,SCALE=1,MAXPAGE=2'
```

LOGANAL can be invoked to obtain LOGOUT output only, with no reports, by using the following parameters:

```
PARM='HISTOGRAM=NO,RESPONSE=NO,OUTPUT=YES'
```

These parameters will bypass sorting and all reports (sort program JCL is not required). Parameters are summarized below; detailed descriptions follow on the next page.

Parameter/Value(s)	Description Summary
ANALYZE={opt | {(opt,opt,...)}}	Criteria for information breakdown.
HISTOGRAM={NO | YES}	Histogram reports request.
MAXDATE=date	Latest transaction date.
MAXPAGE={nnn | 2}	Maximum histogram report pages.
MAXTIME=time	Latest transaction time.
MINDATE=date	Earliest transaction date.
MINTIME=time	Earliest transaction time.
OUTPUT={YES | ddname | NO}	Save/discard sorted log records.
RESPONSE={NO | YES}	Response time reports request.
SCALE={nnn | 1}	Number of messages per vertical line.

NOTE: The parameter choices are all optional, with defaults as indicated. The underlined characters in each parameter keyword indicate the minimum recognized abbreviation for each keyword.

ANALYZE=
specifies the criteria used for information breakdown. The value choices for opt may specify:

- TOTAL--for system totals
- REGION--for breakdown by region (Control Region log data sets only).
- TERM--for breakdown by terminal
- SUBSYS--for breakdown by parent subsystem
- VERB--for breakdown by verb

Only the first letter of each option need be coded (except TO for TOTAL), thus A=(T,S,V) is valid.

If this option is omitted, the default ANALYZE option provides statistics for all levels of breakdowns and traffic histograms by terminal, entire run and verb.

HISTOGRAM=
specifies whether or not histogram reports are required. Code YES if they are required; code NO if they are not required. The default is YES.

MAXDATE=
specifies the maximum date of transactions to be selected by LOGANAL. Transactions that started before MAXDATE, but ended after, are also included. Code as a Julian date yyddd (yy=year, ddd=day of year). The maximum date must be greater than or equal to the minimum date. The default is no maximum date.

MAXPAGE=
specifies the maximum number of pages for a histogram report (up to 999 pages). The default is 2. If an individual report would exceed the number of pages coded for this option, scale is temporarily increased so that every report remains within bounds.

MAXTIME=
specifies the maximum time of transactions to be selected by LOGANAL. Transactions started before MAXTIME, but ended after, are also included. Code as hours only (hh); hours and minutes (hhmm); hours, minutes, and seconds (hhmmss); or hours, minutes, and seconds in hundredths (hhmmssth). The maximum time must be greater than or equal to the minimum time. The default is no maximum time.

MINDATE=
specifies the minimum date of transactions to be selected by LOGANAL. Code as Julian date yyddd (yy=year, ddd=day of year). The minimum date must be less than or equal to the maximum date. The default is no minimum date.

MINTIME=

specifies the minimum time of transactions to be selected by LOGANAL. Code as hours only (hh); hours and minutes (hhmm); hours, minutes, and seconds (hhmmss); or hours, minutes, and seconds in hundreths (hhmmssss). The minimum time must be less than or equal to the maximum time. The default is no minimum time.

NOTE: When a time range is specified over multiple days (MAXDATE ≠ MINDATE), the transactions for those days are summed together and the reports produced are based on the sum. That is, if MAXTIME=1000, MINTIME=0900, MAXDATE=78236, MINDATE=78230, the number of transactions from 9 to 10 for each day specified are added together and the output reports that result are based on that sum, as if one day were specified. Reports for each individual day are not produced.

OUTPUT=

specifies whether or not selected log data is to be saved. The OUTPUT option is used to capture the input log record data used by LOGANAL. This data can be used as input to LOGANAL for additional analysis without reading the entire log data set again. The data may be saved and collected over a period of time for cumulative analysis of log data. It is more compact than the original log data--only log data needed by LOGANAL is retained as 46-byte fixed length records.

If date and time selection criteria are used (MAXDATE, MINDATE, MAXTIME, MINTIME), then the LOGOUT output collects only the selected records. If YES is coded, the log data is written using ddname LOGOUT. If a different ddname is desired, code OUTPUT=ddname. If selected log data is not to be saved, code NO (default).

RESPONSE=

specifies whether or not response time reports are required. Code YES if they are required; code NO if they are not required. The default is YES.

SCALE=

specifies the scale (number of messages) that each horizontal line of the histogram will represent (within constraints defined by MAXPAGE). The default is 1.

Figure 12-5 illustrates JCL for execution. (Note data set names are user-specified and need not correspond to those given below.)

```
                //LOGANAL   JOB
                //STEP1 EXEC    PGM=LOGANAL,PARM='HISTOGRAM=YES,
                //          SCALE=100',REGION=200K
                //STEPLIB   DD  DSN=INT.MODLIB,DISP=SHR
always          //SYSPRINT  DD  SYSOUT=A
required        //PRINT DD  SYSOUT=A
                //ERROR DD  SYSOUT=A
                //LOGIN DD  DSN=INT.INTERLOG,UNIT=xxx,
                //          VOL=SER=xxxxxx,DCB=(RECFM=VB,
multiple        //          BLKSIZE=xxxx,LRECL=xxxx),DISP=OLD
volume          //      DD  DSN=INT.INTLOG2, (etc., as above)
logs            //      DD  DSN=INT.INTLOG3, (etc., as above)

                //SYSOUT    DD  SYSOUT=A
                //SORTLIB   DD  DSN=SYS1.SORTLIB,DISP=SHR
as              //SORTWK01  DD  UNIT=SYSDA,SPACE=(TRK,(200),,CONTIG)
required        //SORTWK02  DD  UNIT=SYSDA,SPACE=(TRK,(200),,CONTIG)
by                  .   .       .           .               .
sort                .   .       .           .               .
program             .   .       .           .               .
                //SORTWK06  DD  UNIT=SYSDA,SPACE=(TRK,(200),,CONTIG)

required        //ddname    DD  DSN=INT.SORTLOG,UNIT=xxx,
for             //          VOL=SER=xxxxxx,
OUTPUT=YES      //          DCB=BLKSIZE=(multiple of 46,
or                          default=920),
OUTPUT=         //          SPACE=(TRK,(50,10)),DISP=(NEW,KEEP)
 ddname

OUTPUT=YES implies ddname=LOGOUT
```

Figure 12-5. Sample JCL for Execution of LOGANAL

Multivolume INTERLOG data sets from one execution of Intercomm should be processed so that the volumes are read in increasing chronological order by LOGANAL. This ensures proper tracking of transactions spanning volumes. If volumes are out of order, statistics from those transactions will be lost.

Multiple INTERLOG data sets from separate executions of Intercomm can be read in any order as long as individual multivolume data sets, if any, are read in consecutive order, as above. When concatenating Multiregion INTERLOG data sets ensure that all data sets are for the same configuration. For satellite region data sets, the data sets must all be for the same satellite region. When control region log data sets are concatenated, all data sets must be from Intercomm execution using the same Region Description Table (RDT).

For documentation of LOGANAL messages and corrective actions, see the Intercomm Messages and Codes.

12.5 THE FILE LOAD PROGRAM (PMIEXLD)

The Intercomm utilities include a program (PMILOAD) which sequentially reads in load modules from a partitioned data set, and creates a BDAM data set. There is one record created on the BDAM data set for each load module (with specified first three characters) on the partitioned data set.

To create the load module for executing this program, use the JCL shown in Figure 12-6. The load module name to be created will then be PMIEXLD.

```
//              EXEC LKEDP,Q=LIB,LMOD=PMIEXLD
//LKED.SYSIN    DD   *
     INCLUDE    SYSLIB(BATCHPAK)
     INCLUDE    SYSLIB(PMIFILET)
     INCLUDE    SYSLIB(PMISERC3)
     INCLUDE    SYSLIB(IXFHND00)
     INCLUDE    SYSLIB(IXFHND01)
     INCLUDE    SYSLIB(PMILOAD)
     ENTRY      PMILOAD
     NAME       PMIEXLD (R)
```

Figure 12-6. JCL to Create PMIEXLD

If LMOD=PMIEXLD is coded on the EXEC statement, the NAME card is not needed. When both LMOD and the NAME card are used, the names must be the same.

This off-line utility is typically used for loading disk-resident table entries for Intercomm execution. However, it may be used for converting members of any partitioned data set to relative blocks (RBNs) of a BDAM data set, as long as the following naming conventions and table entries are met:

- There must be an entry in the File Table (PMIFILET Csect) created via the GENFTBLE macro for each BDAM data set to be loaded. Figure 12-7 illustrates the member PMIFILET defining Intercomm data sets for tables, and one user file. Note that a PMISTOP macro must follow the last entry.

- The member names of the partitioned data set must follow the convention xxx0nnnn where nnnn varies from 0001 to 9999, incremented by 1 with no unassigned entries. (See Section 12.5.1, "Partial File Load.")

- The BDAM data set must be named xxx000 on its DD statement. (See Figure 12-8.)

```
PMIFILET     CSECT
             ENTRY    PMIFILTB
PMIFILTB     EQU      *
             GENFTBLE FNAME=RCT000,BLKSIZE=1500,TYPE=BDAM
             GENFTBLE FNAME=DES000,BLKSIZE=750,TYPE=BDAM
             GENFTBLE FNAME=VRB000,BLKSIZE=750,TYPE=BDAM
             GENFTBLE FNAME=SEC000,BLKSIZE=100,TYPE=BDAM
* BLKSIZE FOR DES000,RCT000,VRB000 CORRESPOND TO INTERCOMM RELEASE
* SPECIFICATIONS.  USER MUST CHANGE FOR LARGER TABLE ENTRIES.
* ADD USER FILE DESCRIPTIONS HERE.
             GENFTBLE FNAME=USERFILE,BLKSIZE=xxxx,TYPE=ISAM,          *
                   DESNUM=7
             PMISTOP
             END
```

Figure 12-7. Sample File Table (PMIFILET)

Each member of the partitioned data set becomes the n-1 RBN of the BDAM data set. At the first "member not found" condition, the load program fills out the current extent of the BDAM file with records containing binary zeros, unless Partial File Load is used.

The JCL shown in Figure 12-8 may be used for execution to load the entire file; the SYSIN control card varies for Partial File Load.

```
//            EXEC    PGM=PMIEXLD,PARM='NOCHECK'
//STEPLIB     DD      DSN=INT.MODLIB,DISP=SHR
//xxx000      DD      DSNAME=xxx000,DISP=(,KEEP),
//                    SPACE=(      ),UNIT=xxxx,
//                    VOL=SER=xxxxxx,
//                    DCB=(DSORG=PS,BLKSIZE=xxxx,RECFM=F)
//xxxLOAD     DD      DSN=INT.MODxxx,DISP=SHR
//             DD     DSN=INT.MODREL,DISP=SHR
//SYSPRINT    DD      SYSOUT=A (default BLKSIZE is 605)
//SYSIN       DD      *
xxx000
/*
```

PARM='NOCHECK'
is used to indicate that each table member being loaded does not contain four bytes of asterisks at its end.

xxx000
should be changed to DES, RCT (or RPT), VRB, or SEC representing the table file being created. If an existing file is being recreated, change the DISP parameter to OLD.

xxxLOAD
is the input PDS containing the table members (xxx00001-xxx0nnnn) to be loaded. xxx must be the same as on xxx000.

Figure 12-8. JCL for File Load Program Execution

Multiple files can be created in the same execution if there is a SYSIN control card for each file, and the pair of xxx000 and xxxLOAD DD statements are defined for each associated output file and input PDS library.

Use of this program for loading table entries for the utilities and sample JCL is contained in the Utilities Users Guide. Figure 12-9 is a summary table reproduced from that document.

Requirements	Utility		
	Edit	Output	Change/Display
ddname of disk resident table entries in Intercomm execution JCL	VRB000	RCT000	DES000
PMIFILET blocksize specification at installation time	750	1500	750
Symbolic Table Entry Library	INT.SYMVRB	INT.SYMRCT	INT.SYMDES
Load Module Table Entry Library	INT.MODVRB	INT.MODRCT	INT.MODDES
Table Entry Library member name convention	VRB0nnnn	RPT0nnnn	DES0nnnn
Coding convention within disk resident entry	VERB macro, RBN=nnnn	REPORT macro, NUM=nnnn	none
Core-resident table requirements.	VERBTBL CSECT: VERBGEN macro plus in-line assembly of disk resident entries	PMIRCNTB CSECT: None (OFT no.-1 is used for RCT000rbn)	PMIFILET CSECT: GENFTBLE macro, DESNUM=DES000rbn or CHNGTB CSECT: DC A(DES000rbn)

Figure 12-9. Conventions for Disk-resident Tables for the Utilities

The control card is printed on SYSPRINT, followed by one or more of the following messages:

- PMILOAD PROCESSING COMPLETE
- PROCESSING HAS BEEN COMPLETED FOR FILE xxx000
- OF THE nnnnn BDAM BLOCKS WRITTEN, nnnnn CONTAINED DATA

Error messages that may appear on SYSPRINT during execution, which result in an abend (U100), are:

- xxxxxx IS AN INVALID FILE NAME
- LOAD MODULE xxx0nnnn IS TOO LARGE FOR RECORD SIZE
- FILE xxx000 CANNOT BE OPENED
- AN INVALID SELECT OCCURRED ON FILE xxx000
- THERE WAS A PERMANENT I/O ERROR ON FILE xxx000
- NO RCDS WERE FOUND FOR FILE xxx000

The following two error messages do not cause an abend:

- ERROR IN INPUT CONTROL CARD--NO PROCESSING DONE
- NO CORE AVAILABLE

A condition code of zero at end-of-job indicates all processing completed successfully. Unsuccessful processing (see the last two messages) results in a condition code of 12.

If an error message is printed, correct the error and rerun the job. See Messages and Codes listings of utility error messages for further explanation.

12.5.1 Partial File Load

The File Load Program allows the SYSIN data set to specify replacement of a specific member of the partitioned data set or creation of a BDAM data set by loading all members within a specified range of member names, starting with member xxx00001. Following are examples:

Example 1

To copy PDS member name xxx00007 to the existing BDAM data set xxx000, the SYSIN data set specifies:

```
// SYSIN DD *
xxx00007
```

For this processing, the DISP parameter for the file being updated (xxx000) must specify OLD or SHR.

Example 2

To create the BDAM data set xxx000 from PDS member names xxx00001 to xxx0nnnn, irrespective of the number of actual members on the PDS, the SYSIN data set specifies:

```
// SYSIN DD *
xxx000-nnnn
```

The File Load Program will copy PDS members in ascending sequence to the BDAM data set xxx000 beginning with xxx00001, which must be present. When a "member not found" condition arises, the File Load does not terminate, but the last member found will be copied to the BDAM data set until the next "member found" occurs, or the "upper limit" member xxx0nnnn is encountered. To illustrate, assume members RPT00001 to RPT00050, RPT00100 to RPT00106 exist on the library INT.MODRCT. The File Load Program specification

```
// SYSIN DD *
RCT000-0110
```

will cause creation of a BDAM data set (RCT000) with 110 RBNs. The member RPT00050 will be duplicated in RBNs 49 to 98 (once as the actual table entry RBN 49, repeated until RPT00100 is found and loaded to RBN 99). The member RPT00106 will be duplicated in RBNs 105 to 109. For Partial File Load, the DISP parameter of the data set being created should specify OLD if a recreate, or NEW if the file does not exist.

There is no limit to the number of control cards input via the SYSIN data set. Further, given the proper JCL, several BDAM data sets may be recreated and/or individually updated in one execution of the File Load Program.

12.6 BDAM FILE CREATION (CREATEGF)

The CREATEGF program is used to create formatted BDAM data sets. The blocks are all formatted with binary zeros. This program should be used to format the disk queue data sets which are defined by the DFLN parameter in the SYCTTBL macro, or LUNIT/LCOMP (VTAM) macros.

NOTE: When formatting disk queue data sets, the number of blocks must always be a multiple of eight.

An additional feature is the ability to place data (for testing or real data) into the file in the relative block number desired as indicated on the RECORD card. Five control cards can be used when executing CREATEGF. These are FILE, RECORD, FIXED, VARIABLE and END cards. If only formatting a BDAM data set without inserting data is desired, the only control card required is the FILE card.

The format of the control cards is:

- FILE card--to designate creation of a new file.

  Column 1--F

  Column 3-10--ddname

  Column 11-17--number of records to allow for in file; must be right-justified, blanks permitted on left.

  Column 19--ONLY option - if 0 or ONLY coded starting in column 19, CREATEGF will create only as many RBNs as requested. If omitted, CREATEGF will fill the last used extent with records, even if this causes more than the number of records requested to be produced.

  The ddname given must be used on a DD statement which describes the file by giving the DCB parameters BLKSIZE and DSORG=DA. This DD statement must be in the job stream when CREATEGF is executed.

- RECORD card--to define the record to be created in the following cards.

  Column 1-3--R1S

  Column 4-5--blank

  Column 6-8--RBN of record to be created (in EBCDIC).

- FIXED card(s)--to designate a fixed-length data field to be placed in the file record indicated by the RECORD card. These cards must be in the order of the data fields on the file record.

  Column 1--X

  Column 2-3--Size of field

  Column 4--1=Binary; 2=Packed Decimal; 3=Character Image

  Column 10-70--Data (EBCDIC)

  where the maximum characters for each field are:

  1. Binary fields--maximum nine characters becoming four bytes binary

  2. Packed fields--maximum 29 characters becoming fifteen bytes packed

  3. Character fields--maximum 60 characters

- VARIABLE Item Code card(s)--to place a field in record preceded by an item code, length, and (optionally) occurrence number. The maximum size of field defined for the FIXED card applies to this card as well. For this card, size of field must include one byte for occurrence number (if specified). Actual size of the field in record will include two bytes for item code and length.

  Column 1--I

  Column 2-3--size of field

  Column 4--1=Binary; 2=Packed Decimal; 3=Character Image

  Column 5-7--item code for data

  Column 8-9--line no. (or 0 or blank)

  Column 10--Data (EBCDIC)

- END Record card--to define end of a record (block).

  Column 1--E

When creating multiple data sets, any number may be created in one step. The Data Set Control Block (DSCB) for the data set created has an Option Code (OPTCD) indication. This can be overridden at execution time by coding an OPTCD subparameter on the DD card (for use with Intercomm File Handler).

Figure 12-10 illustrates CREATEGF JCL and control cards.

```
//         EXEC  PGM=CREATEGF
//STEPLIB  DD  DSN=INT.MODREL,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSSNAP  DD  SYSOUT=A
//SYSUDUMP DD  SYSOUT=A
//DDNAME01 DD  DCB=(DSORG=DA,BLKSIZE=blksize),DISP=(,KEEP),...
//DDNAME02 DD  DCB=(DSORG=DA,BLKSIZE=blksize),DISP=(,KEEP),...
//SYSIN    DD  *
F DDNAME010000080 This is all that is required for monitor disk queues
F DDNAME020000152 Format a user file:
R1S  001          The following data goes into this RBN
X021     1234     Fixed binary field
X032     12345    Fixed packed field
X053     FIELD    Fixed character field
I031001016789     Variable binary field with an item code of 001
                  and a line number of 01.  Note the length of
                  03 includes one byte for the line number.
E                 End of record
/*                End of job
```

Figure 12-10. Example of CREATEGF JCL and Control Cards

12.7 OPSCAN -- SCAN FOR PROGRAM OPERATION CODES

This program analyzes an Assembler Language source module and lists all those statements having significant operation codes. Among the selected operation codes are IBM macros GETMAIN, FREEMAIN, WAIT, POST, SPIE, STAE, CALL and data management functions, and Intercomm macros STORAGE, STORFREE, LINKAGE, PTNLINK, DISPATCH and other significant opcodes. The summary listing thus produced can be readily scanned for significant features of the source program.

OPSCAN is executed using the Intercomm-supplied JCL Procedure OPSCN, described in Chapter 2.

12.8 PRT1403 -- PRINT OUTPUT UTILITY BATCH REPORTS

The PRT1403 Utility is used to format output from the Output Utility to the RPT000 data set, that is, the Batch Report facility.

The PRT1403 Utility provides a line-by-line formatted output as opposed to the snap dump formats that appear in Test Mode normally. Thus, the Batch Report feature can be used to get hard copy formatted output from Test Mode. It can also be used during live execution of Intercomm to obtain formatted output of reports, which for one reason or another (perhaps length) were put out to tape or disk, rather than sent to remote terminals.

Before executing the PRT1403 Utility, a load module must be created. The JCL in Figure 12-11 is used to create the load module. The created load module is executed using the JCL in Figure 12-12.

```
//             JOB
//             EXEC   LKEDP,P=INT,Q=USR,LMOD=PRNTAPE
//LKED.SYSIN   DD     *
          INCLUDE     SYSLIB(PRT1403,BATCHPAK)
          INCLUDE     SYSLIB(IXFHND00,IXFHND01)
/*
```

Figure 12-11. JCL to Create Load Module for PRT1403 Utility

```
//             JOB
//             EXEC   PGM=PRNTAPE
//STEPLIB      DD     DSN=INT.MODUSR,DISP=SHR
//SYSIN        DD     DSN=RPT000,UNIT=__,VOL=SER=__,DISP=(OLD,DELETE),
//                    DCB=(RECFM=V,DSORG=PS,BLKSIZE=1004,LRECL=1000)
//SYSPRINT     DD     SYSOUT=A,
//                    DCB=(DSORG=PS,BLKSIZE=133,RECFM=F)
//SYSUDUMP     DD     SYSOUT=A
//SNAPDD       DD     SYSOUT=A
//SYSSNAP      DD     SYSOUT=A
//SYSSNAP2     DD     SYSOUT=A
/*
```

Figure 12-12. JCL to Execute PRT1403 Utility Load Module

Note that UNIT and VOL=SER parameters on the SYSIN DD statement must correspond with parameters indicated in JCL when creating RPT000. DISP parameter may be altered if user wishes to keep the RPT000 data set. Execution of PRT1403 produces output for the entire RPT000 data set.

12.9 LIBCOMPR -- SYMBOLIC LIBRARY COMPARE

The utility program LIBCOMPR compares two source data sets (or members of partitioned data sets). All statements that do not match are printed. See description of CHANGER program to produce a change deck from a source member comparison.

The statement sequence field (columns 73-80) is used to determine corresponding records. Records are printed if:

- A sequence number in one input data set is not matched in the other input data set.
- Data in correspondingly numbered statements differs.

Statements printed are identified as to which data set contains each statement. If any statements are printed, a summary follows the listing. This summary indicates the number of statements read and the number printed from each data set.

Two input DD statements (SYSUT1 and SYSUT2) and one output DD statement (SYSPRINT) are required. The input data sets must either have standard labels or the block size (multiple of 80) must be specified.

To bypass listing SYSUT2 statements having unmatched sequence numbers, code PARM='S' on the EXEC statement. This is useful when comparing updates (SYSUT1) to a complete existing program (SYSUT2).

A sample JCL stream to execute LIBCOMPR is shown in Figure 12-13. (Appropriate alteration should be made for particular cases.)

```
//             EXEC   PGM=LIBCOMPR,PARM='S'
//STEPLIB      DD     DSN=INT.MODREL,DISP=SHR
//SYSPRINT     DD     SYSOUT=A
//SYSUT1       DD     DSN=LIBR1(MEMBER1),DISP=SHR
//SYSUT2       DD     DSN=LIBR2(MEMBER2),DISP=SHR
```

Figure 12-13. Sample JCL to Execute LIBCOMPR

12.10 UTILITY PROGRAMS TO CREATE INPUT TEST DATA

There are two utility programs which can be used to create input data sets for batch testing:

- CREATSIM--create input messages for BTAM terminal simulator
- SIMCRTA--create input messages for Test Mode execution

12.10.1 CREATSIM Program

CREATSIM accepts only raw data, as from a terminal, and requires a separate execution for each message data set created.

To execute CREATSIM, use the following JCL:

```
//   EXEC  PGM=CREATSIM
//STEPLIB  DD    DSN=INT.MODREL,DISP=SHR
//SYSPRINT DD    SYSOUT=A
//SYSUT2   DD    DSN=message-data-set,DISP=(NEW,KEEP),
//         UNIT=SYSDA,SPACE=...,
//         DCB=BLKSIZE=maximum-message-length+4
//SYSIN    DD    *
message-cards
            .
            .
            .
/*
```

where SYSUT2 defines the sequential output data set containing the messages for one terminal which will be input to the BTAM simulator for an Intercomm execution. This data set is variable unblocked; the BLKSIZE must be at least as large as the largest message record to be created, plus 4. If no BLKSIZE is specified, the default is 304.

From the SYSIN card-image input, CREATSIM constructs a variable-length output record from successive cards until a card with an EOB (X'26') or ETX (X'03') is found. The output record is then written to SYSUT2 and a new record is built starting with the next card. If for some reason it is not desired to fill all 80 columns with data, an end-of-card character must be punched following the data. This character is X'FF'. (Use graphic control described below to assign a graphic to X'FF'.)

Special characters, such as EOB, NL, etc., may be multipunched or, for convenience, other punchable graphic characters may be used. These will be converted by the creation program to the hex value for the special character. Internally provided conversion graphics (identify map) are:

Input			Output	
Graphic	Hex Code	Name	Hex Code	Name
!	X'5A'	Exclamation point	X'26'	EOB,ETB
¬	X'5F'	Negation sign	X'03'	ETX
¢	X'4A'	Cent sign	X'37'	EOT
|	X'4F'	Vertical bar	X'11'	SBA
"	X'7F'	Double quote	X'15'	NL

There are several types of input control cards which are coded starting in column 1. The graphic card sets one-time-only changes to the input translate table to define graphics for special character codes. Formats are:

- GRAPHIC,CLR,ghh,ghh,...

  This clears the translate table for identify mapping and enters new graphic ghh where g is the input graphic and hh is the hex code for the character to be substituted.

- GRAPHIC,ADD,ghh,ghh,...

  This form adds new definitions to the current translate table.

- GRAPHIC,DEL,g,g,...

  This form deletes substitutions from the table for graphic g.

There are also 3270 SBA generation cards. These cards simplify entering of 3270 SBA addresses. First the model is set, if not Model 2 (default), by

SBA,Mn

where Mn is M1 or M2 for Model 1 or 2, respectively.

For each input field, SBA addresses are entered in the message stream by the following format:

SBA,rrcc

where rr is the row number (decimal) and cc the column (decimal) for the beginning of the following field. Row and column are relative to 1, and are defined in the ranges 01 to 24 and 01 to 80, respectively. The subsequent text card begins in column 1 and if it does not end in column 80, it must be delimited with an end-of-card character.

For 3270 simulation, if SIM3270 is included in the Intercomm linkedit, the message text stream must be as follows:

	AID	CURSOR	SBA	rrcc	verb	SBA	rrcc	text-field...	EOB
Value	g	ØØ	|	0102	vvvv	|	rrcc	data	!
Length	1	2	1	2	4	1	2	n	1

where text-field is one or more input text fields separated by SBA cards. The SBA sequence for the verb is optional and the verb itself may be omitted if the terminal is defined as locked to a verb in the Intercomm Front End Network Table.

For AID Values, see IBM 3270 documentation for graphic equivalents. A GRAPHIC card must be coded for CREATSIM to define a graphic equivalent for the Enter key, for example:

GRAPHIC,ADD,<7D

A sample input text stream to CREATSIM for verb CHEK to access account number 12345 from a formatted screen would be:

```
GRAPHIC,ADD,;FF          define end-of-card (field) character
GRAPHIC,ADD,<7D          define Enter key
<ØØ;                     AID value and cursor location
SBA,0102                 optional
CHEK;
SBA,0320
12345;
!
```

This could be followed by the AID value for the next message, etc. For positional (unformatted) input (such as an Intercomm control command), the text statement can be coded:

ØØFLSH$TPUABC01$ALL!

where $ represents the installation standard system separator character.

12.10.2 SIMCRTA Program

SIMCRTA creates input messages for an Intercomm Test Mode execution, and handles multiple data sets in one run. It creates a message data set for each input terminal-ID specified via a MSG card.

SIMCRTA accepts standard Back End test messages and will insert the correct end-of-line character (New Line or CR/LF) at the end of each data card, based upon the STATION and DEVICE tables. It also inserts EOB and EOT (X'2637') at the end of each message. Any special characters, such as HT, VT, etc., must be multipunched into the card. An EMS card must be used to indicate the end of each input message text. SIMCRTA will create as many terminal data sets as necessary at the same time. The message cards do not have to be in order of terminal-IDs. Figure 12-14 illustrates linkedit and execution JCL for SIMCRTA. Note that the ddnames for the corresponding terminal data sets must consist of the terminal-ID preceded by an A; that is, NYC01 must have a ddname of ANYC01. The different data sets may then be specified for SYSIN when executing Test Mode Intercomm.

```
//LKEDCRTA     EXEC       LKEDP,Q=xxx,LMOD=EXSIMCRT
//LKED.SYSIN   DD         *
               INCLUDE SYSLIB(SIMCRTA,TERMCONV,BINSRCH)
               INCLUDE SYSLIB(BATCHPAK,PMIEXTRM,PMISTATB,PMIDEVTB)
//EXECCRTA     EXEC       PGM=EXSIMCRT
//STEPLIB      DD         DSN=INT.MODxxx,DISP=SHR
//ANYC01       DD         DSN=INT.NYC01,DISP=(,CATLG),VOL=SER=yyyyy,
//                        UNIT=zzzz,SPACE=(500,(20,2),RLSE),
//                        DCB=(BLKSIZE=500,LRECL=500)
//ACNT01       DD         DSN=INT.CNT01,DISP=(,CATLG),VOL=SER=yyyyy,
//                        UNIT=zzzz,SPACE=(500,(20,2),RLSE),
//                        DCB=(BLKSIZE=500,LRECL=500)
//SYSPRINT     DD         SYSOUT=A
//SYSIN        DD         *
MSG    A000    NYC01              0001
DEMO
FLN DDNAME01
KEY ABCD
FDN FIELD
RPT 73
EMS
MSG    H000    CNT01              0002
DSPL
FLN DDNAME01
FDN FIELD
KEY ABCD
RPT 73
EMS
/*
```

Figure 12-14. SIMCRTA Linkedit and JCL

12.11 CREATE KEYED BDAM FILE (KEYCREAT)

KEYCREAT creates and preformats a keyed BDAM file of fixed-length unblocked dummy records. The key length and record size are determined by the DCB subparameters KEYLEN and BLKSIZE on the DD statement. The size of the file is determined by the number of records (blocks) indicated in the PARM field of the EXEC statement in the JCL; blocks will be written until the PARM value is reached. In this case, the number of blocks supplied must be a multiple of the number of blocks per track of the device defined by the UNIT parameter on the DD statement describing the file (INTKEYFL). However, if the PARM value is omitted or 0, records will be written under control of the program (not the user) until the primary space allocation is filled.

Figure 12-15 shows the JCL required to execute KEYCREAT.

```
//   EXEC   PGM=KEYCREAT[,PARM='number-of-records-to-create']
//STEPLIB   DD  DSN=INT.MODREL,DISP=SHR
//SYSPRINT  DD  usual-installation-parameters
//SYSUDUMP  DD  SYSOUT=A                              (optional)
//INTKEYFL  DD  DSN=data-set-name-to-be-created,
//          DISP=(NEW,{CATLG},DELETE),
                    {KEEP }
//          SPACE=allocation-parameter,
//          VOL=SER=volserid,
//          UNIT=dasd,
//          DCB=(KEYLEN=key-length,BLKSIZE=blocksize,
//              RECFM=F,DSORG=DA)
```

Figure 12-15. KEYCREAT Execution JCL

There is no restriction on the number of records to be created as supplied in the PARM field other than that the value must be numeric. The data portion of the records is initialized to binary zeros.

Should an I/O error occur, the utility abnormally ends. Should the KEYLEN subparameter be omitted, the utility will issue an appropriate WTO message and abend with a User Code 4. Should the input PARM field contain nonnumeric characters or be too long, the utility will issue an appropriate WTO message and abend with User Code 8. An unsuccessful open of INTKEYFL's DCB results in a related system abend, as no SYNAD exit is provided. The WTOs are documented in Messages and Codes.

12.12 ICOMFEOF - Recover From Missing End of File

ICOMFEOF recovers from a missing/invalid end of file condition on a sequential output file, such as can occur after an operating system or hardware failure in which the file was not closed. In particular, ICOMFEOF is designed to ensure that a valid end of file exists on INTERLOG, the Intercomm log, so that a restart is possible. ICOMFEOF may also be used against a TOTAL data base log file. Coding a PARM on the execute statement indicates a log file and the type of log. In this case, the name coded for the parm is used as the ddname of the log file to be processed.

To determine if a valid EOF exists, the file (disk or tape) is read until one of the following occurs:

1) A valid EOF is detected. In this case there is nothing to do.

2) A no-record-found occurs. This indicates an invalid EOF on disk.

3) A data check occurs. This indicates a missing EOF on tape.

4) If PARM=INT... is specified, then the subject file is assumed to be an Intercomm log. In this case, the log code is validated and the time stamp is checked for the first message in each block. An invalid log code or a descending date/time is treated as a missing end of file. If single region logging is used in a Multiregion Intercomm system, log records on the control region log may not be in ascending order, because the time is set in the satellite region. A local global &DTMARGN may be set to allow for a descending time variance in minutes; see the comments at the beginning of module ICOMFEOF.

5) If PARM=TOT... is specified, then the subject file is assumed to be a TOTAL log. In this case, each record is verified to have a monotonically ascending sequence number. If this check fails, a missing end of file is assumed.

When a missing EOF is diagnosed, the EOF is written by issuing a POINT to position to the last block read successfully. The block is then rewritten and a CLOSE is issued, writing the EOF.

Optionally a WTOR to the console operator can be issued requesting acceptance or refusal of the new EOF. Or, the operator may request an abend so that the System Manager can examine the cause of the missing EOF and the last valid record in the dump. See the description of message RL069R and abend 2222 in Messages and Codes.

Figure 12-16 shows the JCL required to execute ICOMFEOF. IF PARM=INT... or PARM=TOT... is specified, substitute the ddname coded in the parm field for NOEOF, because the latter is used only to process a sequential data set which is not either an Intercomm or TOTAL log file.

```
//          EXEC  PGM=ICOMFEOF(,PARM=....)
//STEPLIB   DD DSN=INT.MODREL,DISP=SHR
//SYSUDUMP  DD SYSOUT=A                                  (optional)
//NOEOF     DD DSN=name-of-sequential-file,
//          DISP=OLD,VOL=SER=volserid,...
```

Figure 12-16. ICOMFEOF Execution JCL.

NOTE: in order to recover a tape data set after an operating system failure, it is important that the tape either be pre-initialized with tape marks (if new - see User Contributed Program Descriptions for sample program) having at least a tape mark at the end of the tape, or previously written on until the end of the tape. This will prevent a runaway tape condition after the last block is read. In order to recover a disk data set after an operating system failure, the IBM utility IEBGENER must be executed to copy the entire data set (primary extent only) to another disk area. This will cause an end-of-file mark to be placed at the end of the extent. ICOMFEOF will then find and mark the real end of file (last valid record) within the extent. Any records created beyond the first extent will be lost as the DSCBs on disk are not updated until the file is closed. When copying the Intercomm log (INTERLOG), RECFM=U must be specified on both the input and output disk data sets for IEBGENER (also BLKSIZE=maximum block size; omit LRECL).

12.13 CHANGER--Produce Change Deck from Two PDS Members

This program compares two partitioned data set Assembler Language members (an original and a modified version of a module), and produces an IEBUPDTE change deck consisting of IEBUPDTE control statements (CHANGE, INSERT, DELETE) and data statements, as necessary. This change deck, if applied to the original version of a module, as defined by the OLDMEM DD statement, would produce the new version of the same module, as defined by the NEWMEM DD statement.

In this way, an original module may be copied to a user's private library and conveniently updated (online via TSO, for example) without disturbing the original, while at the same time keeping an accurate audit trail of modifications. Program output (the change deck) may be SYSOUT (printed or punched) or any desired card-image (LRECL=80) data set. The block size is of the user's choice.

The two versions of the module must have the same sequence numbers except for the changes (deletions).

JCL for executing the CHANGER program is:

```
//          EXEC PGM=CHANGER
//STEPLIB     DD DSN=INT.MODREL,DISP=SHR
//OLDMEM      DD DSN=pds1(original),DISP=SHR
//NEWMEM      DD DSN=pds2(modified),DISP=SHR
//CHANGEDK    DD {SYSOUT={A}  },DCB=BLKSIZE=multiple-of-80
               {       {B}  }
               {DSN=data-set}
```

Appendix A

INTERCOMM TABLE SUMMARY

NAME	MACROS	DSECTS	FILE	DESCRIPTION
AIDSECT	AIDDATA	AIDSECTS		3270 AID key replacement table
	AIDGRP			
BTAMSCTS	SYCTTBL	SCTLISTC	(BTAMQ)	Front End Terminal Queues
	PCENSCT			
BTVRBTB	BTVERB	PVRBTBLE		F.E. Transaction Codes--Verbs
	PMISTOP			
CHNGTB	(DC's)			C/D-Fixed Format Identifiers
COBPCBTB	ICOMPCB	DLIB		DL/I Data Base Interface
CPUIDTBL	(DC's)			3735 Terminal CPU-ID lists
DDQDSTBL	DDQDS	DDQSECTS		DDQ queues dd names
DDQENV	SET.			DDQ execution environment
FDPTABL	(DC's)		FDP000	3735 Terminal FDP lists
FDR	FDHDR	FDRLIST	DES000	C/D-File Record Description
	FDETL			
FENETWRK	BDEVICE	DEVTABL		F.E. Network Definitions-BTAM
	LINEGRP	LGDSECT		
	BLINE	PLNDSECT		
	BTERM	DIALTABL		
	POLLIST	PTRDSECT		
	DFTRMLST	PEXTABLE		
	GFE	GFEDSECT		--GFE/Extended TCAM
	PMISTOP			
FIXTABLE	ICOMFIX	FIXSECT		VS Page Fixing groups
FORMTBLE	(DC's)			FGEN verbs/OFT numbers
INTDEFMT	DRFORM			Data Entry Format Names/Numbers
INTSCT	SYCTTBL	SCTLISTC	(PMIQUE)	Subsystem Control Table
	RESOURCE			
	GENINDEX			
	PCENSCT			
INTSPA	SPALIST	SPALIST		System Parameter Area
				SPA Extension Area
IXFDSCT1	IXFDSCTA	IXFDSCTA		Data Set Control Table

USER-DEFINED TABLE LIST

NAME	MACROS	DSECTS	FILE	DESCRIPTION
KEYTABLE	(DC's)			C/D-Key conversion routines
LOGCHARS	DEFINE	MMUDSECT		MMU Device Processing Definitions
	DEFAULTS			--ASMLOGCH
	COMMAND			--COBLOGCH
	CNTLCHR			--PLILOGCH
	ATTRIB			
LOGSETGB	SET.			Log Analysis generation parms
LOGVRBTB	LOGVERB			Log Analysis utility verbs
LPINTFC	LPINTFC			Link Pack interface list
	LPVCON			
LPSPA	LPSPA			Link Pack resident modules
	LPENTRY			
LUT	VCT	VCT	(VTAMQ)	VTAM network definitions
	LUNIT	LUDSECTS		
	LCOMP			
	VTLSB			
	VTCSB			
	VTLVB			
	VTIDTAB			Terminal-id synonyms
MMU maps	MAPGROUP	MMUDSECT	INTSTORn	MMU map definitions
	MAP			
	SEGMENT			
	FIELD			
	ENDGROUP			
MMUVTBL	MMUVT	MMUVT		MMU vector table
MRMCT	REGCOM	MCTDSECT		MRS-region communications
NEWPOOLS	ICOMPOOL	RMDSECTS		Core pools descriptions
OVLYBTB	(DC's)	OVLYTBL		Overlay B verb table
PADDTBLE	PADD			Editing pad characters
PAGETBLE	PAGETBL	PGEDSECT	(PAGES)	Terminal/Page file lists
PMIALTRP	PMIALTRN	ALTREPRT		Alternate Terminal OFT Reports
PMIBROAD	BCGROUP	BRODSECT		Broadcast MSG. terminal groups
PMIDEVTB	DEVICE	DEVLISTC		B.E. device descriptions
PMIFILET	GENFTBLE	FTBLISTC		File tables (E/O, C/D)

USER-DEFINED TABLE LIST

NAME	MACROS	DSECTS	FILE	DESCRIPTION
PMIRDT..	REGION	RDTSECTS		MRS-Region Description Table
	SUBSYS			
	GENRDT			
	MRPASSWD	MPWDSECT		RAP processing
PMIRPTAB	(DC's)			OFT Terminal Restrictions
PMISECTB	STATION	STALIST	SEC000	Basic Security Processing Table
	SECVERBS	SECTB		
	GENSEC			
PMISTATB	STATION	STALIST		B.E. Terminal Table
	DVMODIFY	DVMODIFY		
	PMISTOP			
PMITIMTB	TMZONE	TIMETBL		Time-of-Day/Subsystem List
PMIVERBS	VERB	VERBTBL	VRB000	Verb Editing Control (ECT)
	PARM			
	PMIELIN			
	VERBGEN			
PTRNTBL	PATRN			C/D--Output Edit Patterns
REENTSBS	SUBMODS	DYNDSECT		Subroutine Codes/Entries
REPTAPE	(DC's)		RPT000	OFT Report Spooling to Tape
RPT.....	REPORT	RCTLISTC	RCT000	OFT Report Definitions
	LINE			
	ITEM			
SAMTABLE	MAPACCT	SAMCB		SAM Reporting Areas
SECURITY	(DC's)			Basic Security User Exit VCONs
SETENV	SET.			F.E. Network Environment
SETGLOBE	SET.			System Control Globals
TOTFILE	TOTFLGEN			TOTAL Data Base Files
TRANGEN	GENERTRN			MSG--Transaction Generation
TUNERTBL	(DC'S)	TUNRTBLC		Fine Tuner Commands SS names
USERSPA	(DC'S)			User Extention to INTSPA
USRBTVRB	BTVERB	PVRBTBLE		User additions to BTVRBTB
USRSCTS	SYCTTBL	SCTLISTC		User additions to INTSCT
USRSUBS	SUBMODS	DYNDSECT		User additions to REENTSBS
USRVERBS	VERB,etc.	VERBTBL		User additions to PMIVERBS

Appendix B

INTERCOMM MESSAGE HEADER

The following lists the names and formats of all fields in the Intercomm message header, and describes their contents and changeability.

Field Name	Length	Description	Alter Legend*
MSGHLEN	2	Length of message, including header (binary number)	Y
MSGHQPR	1	Teleprocessing segment I/O code: 02/F2=full message 00/F0=header segment 01/F1=intermediate segment 03/F3=final (trailer) segment	N
MSGHRSCH	1	Receiving subsystem code high order byte (binary zero if terminal output)	Y
MSGHRSC	1	Receiving subsystem code low order byte (binary zero if terminal output)	Y
MSGHSSC	1	Sending subsystem code low order byte (binary zero if terminal input)	M
MSGHMMN	3	Monitor message number assigned by Message Collection	N
MSGHDAT	6	Julian date (YY.DDD). The period is a one-byte message thread number (for resource management and/or message restart purposes).	N
MSGHTIM	8	Time stamp (HHMMSSTH)	N
MSGHTID	5	Terminal identification (originating terminal on input messages, destination terminal on output) or Broadcast Group Name	Y

Field Name	Length	Description	Alter Legend*
MSGHCON	2	Reserved area	N
MSGHCON+1 (MSGHRETN)	(1)	Subsystem Return Code (Log Code X'FA' entries only)	N
MSGHPID	5	Reserved area	N
MSGHSSCH	1	Sending subsystem code high order byte (binary zero if terminal input)	M
MSGHUSR	1	Reserved (see below)	L
MSGHBMN	2	Front End message number (binary)	N
MSGHLOG	1	Log Code (see Chapter 9)	L
MSGHBLK	1	Reserved area	N
MSGHVMI	1	Verb or Message Identifier interpreted by receiving subsystem as required, and by FESEND.	Y

*Alter Legend:

Y - must be filled in by application program for a message for Output Utility, a terminal, or another subsystem. For calls to FESEND(C), MSGHVMI should be set to X'57' or X'67' as appropriate for output messages, and MSGHRSCH/C must be binary zeros (low values).

M - Should be filled in for user's own information (required by Intercomm for restart)

N - DO NOT TOUCH (must be copied from input message header to output message header)

L - may be modified for user codes based on subsystem logic.

NOTE: Log records are blocked by LOGPUT with a Block Descriptor Word containing the block length at the beginning of the block. The individual message records within the block do not, however, contain Record Descriptor Words. Intercomm uses the length in the message header to increment to the next message in the block. Therefore, the blocks are written as RECFM=U. Do not use programming or JCL access to the log as RECFM=VB.

MSGHUSR is used for interface with Intercomm modules as follows:

1. If the input verb had HPRTY=YES coded for the BTVERB macro; MSGHUSR contains a C'P' to request priority queuing for the subsystem. The user may move a C'P' to this field to request priority queuing for output messages to a terminal (via FESEND) or to another subsystem (via Message Collection).

2. For messages to be processed by the Edit Utility; contains C'F' to indicate that the input message was from a 3270 CRT and contains SBA sequences.

3. For output messages to a switched async device (Teletype, Dataspeed 40, and 2740); a C'B' requests disconnect after transmitting the output message.

4. For output messages to a switched Teletype or Dataspeed 40 device; a C'X' requests using the alternate call-list for the next input message (as described in the BTAM Terminal Support Guide).

5. For output messages to a switched IBM 7770 device; this field must contain one of several optional values, as described in the BTAM Terminal Support Guide.

If none of the above considerations are applicable, the subsytem may use this field for messages queued to other user subsystems, or for special logging information, as desired. The LOGPRINT utility always prints the value coded in this field (in hexadecimal).

Appendix C

USER CODING OF THE SCT OVERLAY INDEX

As illustrated in Figure C-1, the following coding conventions must be utilized when the SCT Overlay Index is coded by the user:

- Code an ENTRY statement for SPA references, as follows:

  ENTRY SCXFSOG1,SCXESCX,SCTRES

- The first word must be labeled SCXESCX, as one of the fields in the SPA Csect is an address constant referencing this label (SPAPSCX).

- The header of the Subsystem Control Table Index consists of the three fields SCXESCX, SCXFRSS and SCXLRSS. SCXESCX is an address constant pointing to the last detail entry in the index. The end of the index has been given a label SCXEND. Since each detail entry is twelve bytes long, the constant A(SCXEND-12) will point to the first word of the last entry, even if new overlay groups are inserted in the index during future maintenance.

- The fields SCXFRSS and SCXLRSS contain the address of the first and last entries in the resident portion of the Subsystem Control Table. Code as A(0) if there are no resident or dynamic load SCTs. The value SCTSIZE has been subtracted from the address of the end of the group to establish the starting address of the last entry.

- At the beginning of the Subsystem Control Table, the label SCTRES must be placed on a DS statement that establishes a fullword boundary for the beginning of the first entry for a resident subsystem in the table. The end of the first entry has been flagged with the label SCTERES1. At the end of the table (following all of the overlay groups), these labels have been used in an EQU statement to establish the size of an individual entry (SCTSIZE). This size is an important figure in coding the address constants in the index.

- The first word following the entries for the resident portion of the Subsystem Control Table is indicated by the label SCTLRES. The first word following each subsequent overlay group is labeled SCTLOVn, where n identifies the overlay group. With these labels established, the coding of the index can proceed.

- The coding of detail entries for each subsystem group consists of the fields SCXFSOGn, SCXLSOGn and SCXOVNMn, where n is varied to distinguish between overlay subsystem groups.

- The OVLY number is the constant value coded at SCXOVNMn. (Overlay numbers 1, 2 and 3 are reserved for Overlay Regions B, C and D.) Numbers 4 through 62 are used for subsystem groups in Overlay Region A. SCXFSOGn is the address of the first entry in the Subsystem Control Table for each overlay group. SCXLSOGn is the address of the last entry in the group; the value SCTSIZE is used to calculate this address. Each detail entry is padded at the end with three bytes. Coded as shown in the example, the labels SCTLRES and SCTLOVn serve the dual purpose of defining the address of the last entry in the preceding group, and the starting point of the next group.

- When the overlay index is user-coded, the GENINDEX macro must specify OVLYNDX=NO.

```
*  THE THREE FOLLOWING FIELDS CONSTITUTE THE HEADER OF THE SYSTEM
*       CONTROL TABLE INDEX
SCXESCX   DC     A(SCXEND-12)          ADDR 1ST WRD, LAST DTL ENTRY.
SCXFRSS   DC     A(SCTRES)             ADDR 1ST SCT FOR RES S/S.
SCXLRSS   DC     A(SCTLRES-SCTSIZE)    ADDR LAST SCT FOR RES S/S.
*
*  THE FOLLOWING CODE ESTABLISHES A DETAIL ENTRY IN THE INDEX FOR
*       THE FIRST OVERLAY GROUP.
SCXFSOG1  DC     A(SCTLRES)            ADDR 1ST SCT FOR OVLY FOUR
SCXLSOG1  DC     A(SCTLOV1-SCTSIZE)    ADDR LAST SCT FOR OVLY FOUR
SCXOVNM1  DC     AL1(4)                OVLY NUMBER.
          DC     3B'0'                 PADDING.
*
*  THE FOLLOWING CODE ESTABLISHES A DETAIL ENTRY IN THE INDEX FOR
*       THE SECOND OVERLAY GROUP.
SCXFSOG2  DC     A(SCTLOV1)            ADDR 1ST SCT FOR OVLY FIVE
SCXLSOG2  DC     A(SCTLOV2-SCTSIZE)    ADDR LAST SCT FOR OVLY FIVE
SCXOVNM2  DC     AL1(5)                OVLY NUMBER.
          DC     3B'0'                 PADDING.
*
*  DETAIL ENTRIES FOR ADDITIONAL OVERLAY GROUPS MAY BE INSERTED HERE.
*
SCXEND    EQU    *                     ADDR OF END OF INDEX.
*
*  FOLLOWING IS THE SYSTEM CONTROL TABLE.
*
SCTRES    DS     0F       1ST SCT FOR RES S/S.
          SYCTTBL
SCTERES1  DS     0F
          SYCTTBL                      ADDITIONAL SCT'S
          SYCTTBL                        FOR RESIDENT OR DYNAMICALLY
          SYCTTBL                        LOADED SUBSYSTEMS.
SCTLRES   DS     0F
*  BEGINNING     0F       OVERLAY A GROUP ONE
          SYCTTBL                      SCT'S FOR
          SYCTTBL                        OVERLAY
          SYCTTBL                          GROUP 4
SCTLOV1   DS     0F
*
*  BEGINNING     0F       OVERLAY A GROUP TWO
          SYCTTBL                      SCT'S FOR
          SYCTTBL                        OVERLAY
          SYCTTBL                          GROUP 5
SCTLOV2   DS     0F
*
*  ADDITIONAL OVERLAY GROUPS 6 THROUGH 59 MAY BE INSERTED HERE.
*
SCTSIZE   EQU   SCTERES1-SCTRES
          GENINDEX OVLYNDX=NO
          PCENSCT
          END
```

Figure C-1 User-Coded Subsystem Control Table Index Structure

Appendix D

INTERCOMM USER EXITS

D.1 INTRODUCTION

Generally, user exits are conditionally called (CALLIF) for special processing, for example:

- Additional error recovery (terminals, files, etc.)
- Cancelling/modifying/routing of messages
- Additional security checking
- Statistics gathering
- Additional startup and closedown processing
- VTAM interface processing

D.2 CODING CONVENTIONS

Unless otherwise documented for the specific user exit, the following coding conventions for user-coded or user-modified exits from Intercomm processing routines must be observed:

- Written in Assembler Language only
- Reentrant (establish and chain save areas)
- Use standard linkage conventions; at entry

R15	contains address of user exit
R14	contains return address to caller
R13	points to caller's save area
R0	generally not used
R1	contains a parameter, or points to a parameter list which contains one or more addresses or values, as documented for each exit
R2-12	may contain additional parameter values, as applicable

- If STORAGE macro used to acquire save/work area, RENT=NO must be coded
- SPALIST address can be acquired via GETSPA macro
- Do not give up control to the Dispatcher either directly (dispatcher macros) or indirectly (call to Intercomm service routine or user subroutine) if documented as prohibited
- Most of the exits are called in thread 0 (system thread)
- Intercomm macros for application programming documented in Basic System Macros and the Assembler Language Programmers Guide may be used; be aware of putting the caller in a wait state if an INTENQ macro is issued
- Intercomm control commands may be issued (format message and queue via FESEND, or via FESENDC if message storage area to be copied)
- At exit, if documented, user must pass back a return code, either via register 15 or in a status word
- Exit must return to caller.

D.3 LIST OF USER EXITS

Note the following:

Source: YES indicates sample source code provided on SYMREL

Doc: the Intercomm manual in which the user exit is described, as follows:

BTG = BTAM Terminal Support Guide
DBMS = Data Base Management Users Guide
DEIG = Data Entry Installation Guide
ESS = Extended Security System
M&C = Messages and Codes
ORM = Operating Reference Manual
SNA = SNA Terminal Support Guide
TSG = TCAM Support Users Guide
UUG = Utilities Users Guide

Name	Caller	Source	DOC	Comments
CHNGEXIT	CHANGE	-	UUG	called before updating a file
COPYEXIT	COPYSS	-	BTG	allow/cancel COPY processing
DDQEXIT	DDQINTFC	-	-	segmented input messages processing
DEUSEXIT	INTENTRY	-	DEIG	Data Entry input data editing etc.
DEUSEXTR	INTDEXTR	-	DEIG	Data Entry extracted record processing
INQEXIT	BMH000	-	BTG	modify header/free msg. before queuing
IOEXIT	RMNADISA	-	ORM	for thread or file hung-I/O time-out
SECUEXIT	INTSEC02	-	ESS	ADD/SIGNON/SIGNOFF processing
SNAPEXIT	PMISNAP1	-	ORM	determine whether to take a snap
SPINEXIT	SPINOFF	-	ORM	use to generate job to print SNAPDD
SPSNEXIT	SPIESNAP	-	ORM	determine whether to take snap 126
TCAMSTRT	STARTUP3	-	TSG	Basic TCAM startup after Qs open
TCAMCLSE	CLOSDWN3	-	TSG	Basic TCAM closedown processing
USERINIT	STARTUP3	-	TSG	Basic TCAM - check TCAM MCP active
USERLOGE	LOGPUT	-	ORM	log statistics gathering
USERPDBE	PDATBASE	-	DBMS	DATBAS calls statistics gathering (TOTAL)
USRBSCEX	BSCLEASE	-	BTG	leased CPU optional error recovery
USRBTLOG	BTSEARCH VTRECVE	YES YES	BTG SNA	after verb verification-log input msg (F1) after input msg. received, header formatted
USRCANC	SYCT400 IJKTLOOP MRCSAMOD	YES	ORM	See PMICANC-issue 'message cancelled'
USRCHKPT	CHECKPT3	-	ORM	called prior to write checkpoint record
USRCLOSE	CLOSDWN3	YES	ORM	issue 'Intercomm closed' message
USRCLSE1	USRCLOSE	-	ORM	additional closedown processing

Name	Caller	Source	DOC	Comments
USRCONVE	CONVERSE	-	-	called when entered from subsystem
USRCONV1	CONVERSE	-	-	called when entered from Subsystem Controller (next message received)
USRECRY	BLHIN BLHOT	YES	BTG	error recovery message handling
USRER129	BLHIN BLHOT	YES	BTG	129 Card Read/Punch hardware error recovery
USRESTRT	LOGPROC	-	ORM	restart message option
USROTEDT	PMIOUTPT FESEND	-	ORM	formatted output message changes
USROUTCK	PMIOUTPT	-	ORM	cancel message formatting by Output
USRPRMPT	INTSEC00	-	ESS	suppress sign-on prompt message at startup
USRSAM01	SAMSECT	-	ORM	SAM user function routines (01-10)
USRSEC00	INTVRB00 INTSEC00 INTSEC02	-	ESS	security statistics gathering, etc.
USRSEREX	BLMSGCOL	YES	ORM	Serial Restart - Front End input message queuing exit
USRSGNOF	PMISIGN	-	ORM	Basic Security sign-off checking
USRSGNON	PMISIGN	-	ORM	Basic Security sign-on checking
USRSTART	STARTUP3	YES	ORM	issue 'Intercomm started' message
USRSTRT1	USRSTART	-	ORM	additional user startup processing
USRTDWN	BDIAL TPUMSG	- -	BTG BTG	terminal disconnected processing terminal down (TDWN) processing
USRTPUP	TPUMSG	-	BTG	terminal up (TPUP) processing
USRTRAP	TRAP	-	M&C	user TRAP debugging
USRWTO	WTOMOD	-	M&C	additional system message output routing
USRXIN	BLHIN	-	BTG	input msg. modification before queuing

Name	Caller	Source	DOC	Comments
HALT	VTLUCMD	-	SNA	SPLU$TPUxxxxx$HALT processing
INQUEUE	VTRECVE	-	SNA	before msg. queued for Back End
LOGON	VTEXITS	-	SNA	after OPNDST completed
LUS	VTRECVE	-	SNA	LUS (sense data) received
OTQUEUE	VTQMOD	-	SNA	before msg. put on component queue
OUTSEG	VTSEND	-	SNA	if VTLSB specifies SOUTSEG=USER
RCVEXCD	VTRECVE	-	SNA	invalid input message
SHUTD	VTLUCMD	-	SNA	SPLU$TPUxxxxx$SHUTD processing
SIGNAL	VTEXITS	-	SNA	Signal Expedited Flow command received
SNDABT	VTSEND	-	SNA	SEND error recovery processing
SNDEXR	VTRESP	-	SNA	negative response received
SNDNRM	VTRESP	-	SNA	positive response received
VTURLRX1	VTEXITS	YES	SNA	Intercomm-supplied version of VTUSRLRX exit
VTUROTX1	VTQMOD	YES	SNA	Intercomm-supplied version of OTQUEUE exit
VTURSDX1	VTREGP	YES	SNA	Intercomm-supplied version of SNDNRM exit
VTUSLGNX	VTEXITS	-	SNA	final validation of LOGON request
VTUSRLRX	VTEXITS	-	SNA	VTAM RELREQ exit scheduled
VTUSVSDX	VTLUCMD	-	SNA	VTCN$SHUTD command processing

INDEX