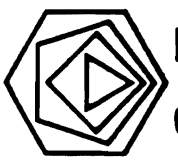
INTERCOMM

CONCEPTS AND FACILITIES



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

LICENSE: INTERCOMM TELEPROCESSING MONITOR

Publishing History

Publication	Date	Remarks
First Edition	September 1973	This manual corresponds to Intercomm Release 6.0.
Second Edition	July 1975	Revisions, corresponding to Intercomm Release 6.2.
Third Edition	October 1977	Revisions and updates, corresponding to Intercomm Release 7.0.
Fourth Edition	July 1980	Revisions corresponding to Intercomm Release 8.0.
Fifth Edition	January 1987	Revisions corresponding to Intercomm Release 9.0.

PREFACE

Intercomm is a state-of-the-art teleprocessing monitor, operating under the control of IBM System/370 operating systems (MFT, MVT, VS, MVS, or XA). Intercomm monitors the transmission of messages from and to 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.

Intercomm meets the need for a well-designed and functional real-time monitor system, minimizing cost and time required in implementation. Intercomm imposes no restrictions on access methods, number of terminals or source languages for the user's programs, and is capable of fast and complete recovery of message and file status of a failed system. The result of such centralized control of multiple, varied on-line customer applications in a system with broad capabilities is maximum utilization of computer resources.

This document presents the concepts of operation and describes the facilities provided in the Intercomm on-line teleprocessing monitor system. Familiarity with operational concepts of an on-line environment is assumed on the part of the reader. For further details, please consult the various Intercomm technical publications listed on the Intercomm Publications page of this manual.

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 Descriptions

FEATURE IMPLEMENTATION MANUALS

- 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
- Store/Fetch Facility
- SNA Terminal Support Guide
- TCAM Support Users Guide
- Utilities Users Guide

TABLE OF CONTENTS

Chapter 1

THE NEED FOR INTERCOMM

1.1 THE DEVELOPMENT OF INTERCOMM

Since its introduction in the spring of 1969, Intercomm has matured through nine major releases. The capabilities and facilities of Release 9.0, the current release, are described in this manual. Intercomm has demonstrated adaptability and viability throughout this period of volatile technology. From the IBM 360 to the System/370 range of computers, from OS MFT to MVT to VS1 to MVS, and now to XA, it has continued to provide the advantages and to accomodate the disadvantages of the newer technologies.

The Intercomm teleprocessing monitor is a state-of-the-art software system which supplies those services required in a comprehensive transaction processing environment. These services are efficiently provided to maximize throughput and minimize response time. Hundreds of users have utilized Intercomm, each over many years of production. Such extensive use guarantees the highest level of code itegrity within the Intercomm-provided software and ensures that these services have been rigorously field tested.

Intercomm's features include:

- Communications Functions for Devices and Line Control

  Provided through the user's selection of BTAM, TCAM, and/or VTAM for physical device management, and through other Intercomm Front End components, for logical network management.

- Full Resource Management

  Main storage, files, CPU cycles

- Job Management

  Scheduling, loading

- Task (Program) Management

  Queueing, concurrent processing, swapping, multithreading

- System Control

  Security, message and file restart/recovery, logging

- Application Program Services

  Message formatting/editing, scratch pad management, data storage and retrieval, conversational control

- Preprogrammed Applications

  Message switching/broadcasting, data entry, record retrieval

The Intercomm design philosophy has resulted in a system which meets the following needs:

- High Programmer Productivity

  Intercomm includes a wealth of preprogrammed functions which are table-driven, not requiring coding, testing and debugging by the application programmer. An Intercomm program is written as much like a batch program in the programmer's native language (Assembler, COBOL, PL/1 or FORTRAN) as possible in an on-line environment. The programmer is shielded from the communications devices by either the Edit/Output Utilities or Message Mapping Utilities. The created program is device-independent. It deals with fixed-format, fixed-field messages (both input and output). It calls the monitor for preprogrammed services. The programmer can concentrate on the logic of the application rather than the bits and bytes of the teleprocessing environment.

- Low System Overhead

  Intercomm makes low system overhead possible in both CPU cycle and main storage utilization. This is particularly significant in the area of data management. File I/O is centralized in Intercomm's generalized File Handler, which allows for true dynamic buffering, a single DCB or ACB for each file, exclusive control at a physical record level, and dynamic file allocation. A file can thus be opened and closed once per day without high overhead. Another area where low system overhead is highly significant is that of obtaining and freeing areas of main storage on an as-required basis. Intercomm's Resource Management facilities contain an option for linkediting predefined core pools (areas of user-specified size and number) with the system. Intercomm services storage requests from these pools, utilizing a best fit technique. This prevents storage fragmentation and reduces CPU utilization by bypasssing GETMAIN/FREEMAIN overhead. Via tuning, the user can reduce the pool sizes to the lowest possible storage requirements consistent with the desired level of performance.

- High System Throughput

  Intercomm is a system without any built-in roadblocks. Given a high enough priority message volume on a standalone computer, Intercomm will use 100 percent of the available CPU cycles on a machine while continuing to process messages. High system throughput is obtained by efficient message processing management within the Intercomm job. This job management capability, unique to Intercomm, is primarily implemented through the Subsystem Controller, which controls the scheduling of applications within an Intercomm region or partition. Scheduling is based on resource availability and demands, and an algorithmic combination of the priority of the input terminal, the message and the application program. The Subsystem Controller, in conjunction with the Dispatcher (Task Manager) and other components, totally controls the sequence of execution of all messages, such that maximum processing overlap occurs (through concurrent processing and multitasking) whenever possible.

- High System Integrity

  Intercomm will detect application program loops (CPU and non-CPU bound), intercept application program checks and ABENDs (SPIE and STAE protection), and handle terminal and file I/O error situations. In the event of system failure, it provides for message and file restart/recovery. In addition, the multiregion version of Intercomm (MRS) allows for protect-key separation of application systems (groups of application programs) into their own independent regions, partitions or address spaces, thus providing the highest possible level of system integrity.

- Future Growth

  The Intercomm system contains no restrictions on the number of terminals or application programs which it can support and manage efficiently. The additional storage requirement for adding new terminals and/or applications is minimal. Intercomm is essentially operating system-independent, permitting an easy migration from one operating system to another without application program impact. Data Base Management System (DBMS) support has grown from TOTAL and DL/1 (IMS) to include ADABAS, IDMS, Model 204, and System 2000, and can be expected to increase as new packages are introduced. Intercomm's MRS includes the unique provision for concurrent support of multiple DBMS. Intercomm's terminal support has continued to expand over the years to include the newest devices in a timely manner. It went from the 2260 under GAM to the 3270 under BTAM or TCAM, and the 3600 and 3790 (Inquiry Mode) to the 3270 (BSC and SDLC) under VTAM, and can be expected to support additional future device

protocols as SNA becomes more widely accepted. Intercomm's modularity facilitates the inclusion of existing (and future) new features without changes to the current production application programs. This generally involves a relinkedit of the load module for the Intercomm system with additional INCLUDE cards for the new features and requires table changes at the system level. Intercomm is indeed a software system with the flexibility to meet current needs as well as future requirements.

- Vendor Support

  We offer on-site and telephone support by thoroughly trained and experienced Systems Engineers (SEs). Product maintenance and improvements are provided via updates and/or new releases on tape. Automated facilities for their application are described in the ASMF Users Guide and Installation Guide. Intercomm Education is available in both public and on-site courses. The courses are described in the Intercomm Customer Course Catalog. Additionally, Technical Information Bulletins are issued periodically to describe vendor or Intercomm fixes for hardware problems, suggested operating system or environment dependent user mods, etc.

- User Contributed Library

  The Intercomm User Group Contributed Library (IUGCL) is a repository of vendor-field-developed and user-developed Intercomm software. It is distributed in source form on the Intercomm release tape. The programs are provided without a warranty, however, support may be provided on a Time and Materials basis dependent on the availability of an Intercomm System Engineer (SE) who has familiarity with the particular programs. Documentation for the programs is contained in the Intercomm User Contributed Program Descriptions manual.

1.2 THE INTERCOMM ENVIRONMENT AND FACILTIES

Intercomm executes under IBM System/370 operating systems on any System/370 machine (43xx,30xx) including the new 3090, and compatible machines. It is IBM operating system-independent and functions under OS/MFT, OS/MVT, VS1, MVS and XA. It executes as a single job (or multiple independent jobs with the Multiregion Support Facility), normally concurrent with the user's regular workload in other partitions or regions.

Within its region, Intercomm may be thought of as an operating system within the operating system. The computer's operating system was designed to efficiently handle a batch workload consisting of multiple programs performing iterative functions over a relatively long time period. Intercomm was designed to supervise an on-line workload consisting of multiple programs performing processing functions once per input message. The batch program reads and processes a series (batch) of sequenced transactions from a single input source. The on-line program receives a message (transaction) from a communications device, processes the message, transmits one or more responses (output), and then is idle until another transaction is entered. If a user had only one program, which processed messages from one terminal, this situation might be acceptable. However, the usual situation is that the user has more than one terminal and more than one application program. With multiple terminals submitting multiple messages to multiple application programs, a control program is necessary; hence Intercomm.

The five main Intercomm components that perform this control over the on-line environment are as follows:

- Teleprocessing Interface (Network Management)

  Controls communication between the CPU and all teleprocessing terminals connected with Intercomm.

- Message Collection/Retriever (Queue Management)

  Controls queuing and dequeuing of queues of messages awaiting input processing or output transmission.

- Subsystem Controller (Job Management)

  Schedules and manages the sequence of initiating application subsystems (message processing programs) within Intercomm.

- File Handler (Data Management)

  Manages all file I/O activities in a centralized manner.

- Dispatcher (Execution Management)

  Manages the allocation and overlap of CPU time among multiple tasks operating concurrently, and establishes multiple real time clocks.

Intercomm's efficiency is achieved by processing parallel messages under the control of the Subsystem Controller (message processing scheduler), the Dispatcher (CPU scheduler), the teleprocessing interface (Front End message handler) and the File Handler (data management interface).

All incoming and outgoing messages are concurrently routed to the various processing programs via message queues managed by Message Collection and Retrieval. Intercomm's capabilities extend not only to scheduling and dispatching tasks for high volume throughput, but also to providing the necessary linkage for any operating system file access method. All I/O processing is centralized via a special Intercomm facility called the File Handler. Because of the multitasking Dispatcher, the CPU is active during file access procedures, and can be scheduled for other processing functions.

Intercomm is designed for processing both small and large volume multiapplication teleprocessing systems with many types of input messages. There is no restriction, aside from the amount of main storage available for the partition or region, as to the size or maximum number of nonreentrant or reentrant modules/programs that can be controlled by the monitor. Nonreentrant programs may either process one message at a time in parallel with other programs, or multiple copies of the program may be used. Within each reentrant Assembler, PL/1 or COBOL application program under control of the monitor, many messages can be operated upon concurrently by multithread processing.

A special load facility allows for dynamic loading of user-specified subsystems or subroutines. A linkedit is not required; a system interface resolves the external references. This facility allows revision and/or correction of user subsystems or subroutines while Intercomm is executing.

In the event of a program check within an application subsystem, the Intercomm job does not abend. The program responsible for the error is terminated for the message being processed, but the rest of the system continues execution. Intercomm will snap the program's registers, debugging information, and the related areas of main storage, then will free files, and cancel the message in progress. The operator at the originating terminal is notified of the message cancelled condition, and the master control terminal receives a detailed message describing the program check condition. Whether or not the program with the error continues to process additional messages is the user's option. A feature is available to spin off snap output to a separate data set which may be printed concurrently with on-line execution, rather than after job termination. Under MVS or XA, snaps may be routed to a SYSOUT data set which is deallocated and queued for printing immediately on completion of the snap.

With its unique fast message and file restart/recovery capabilities, Intercomm ensures safe recovery of all messages and files and coordinated restart in the event of computer failure or system abend. The Restart/Recovery facility encompasses queued messages, messages in process, and completed messages. Queues and files are automatically reconstructed from data contained on the system log (journal). Completed messages are discarded; duplicated messages are eliminated; processing is resumed from the point of failure, in most cases within minutes of the start of execution.

The system log used in conjunction with Restart/Recovery may be either disk or tape-resident. The log is also used off-line for generating the performance and accounting statistics discussed in detail in subsequent sections. (Additional statistics are maintained on-line. These are available via input of preprogrammed inquiry commands).

With Release 9.0, a disk logging protection feature is available to ensure that system operation continues after an out of space condition (B37, D37 abend) for the log data set. A facility is also provided to ensure the log can be recovered in the event of an operating system crash.

Intercomm also provides extensive Resource Management facilities. The Storage Management option offers an optimal environment within which the user may effectively obtain and audit/purge resources associated with an application program, including files, enqueued system facilities or subroutines, and main storage. In the event of system/application failure, full diagnostic capabilities are provided. This is realized with minimum OS SVC usage. Furthermore, user-specified storage pools which section the available space into predetermined blocksizes result in efficient storage allocation. This helps eliminate fragmentation of main storage and increases the speed with which storage may be obtained and freed. Core-use particulars, in the form of global and/or detail statistics, are collected at a user-specified interval and printed off-line. These statistics can be used for tuning the system.

Another Resource Management option is a storage cushion feature. Of user-selected size, this block of main storage protects against the degrading effects of temporary core shortage.

1.3 INTERCOMM MESSAGE FLOW

The various capabilites and functions of the Intercomm on-line system monitor are described in the following sections. The Intercomm components and their interaction are described with respect to the processing of a single message. Two schemes are presented, showing use of Message Mapping Utilities (MMU) and the Edit/Output Utilities. (The utilities are described in detail in Chapter 4, "On-Line Utilities.")

1.3.1 Message Flow Using Message Mapping Utilities (MMU)

In Figure 1, the numbered arrows in the diagram correspond to the numbered paragraphs below:

1 The Front End (teleprocessing interface) reads an input message and prefixes it with a 42-byte control header, which is based upon user-supplied data in the Front End Verb (transaction control) Table. The header contains routing information, time, date, originating terminal and message length. The message is then queued for subsystem processing by Message Collection.

2 The Subsystem Controller schedules the application program and retrieves the message based upon Subsystem (program) Control Table (SCT) scheduling criteria.

3 The message is passed to the application.

4 Input in terminal-dependent format is transformed to a terminal-independent form by a call to MMU.

5 The application program performs message processing logic, requesting I/O service functions from the File Handler or DBMS interface via CALL statements.

6 The application program creates a terminal-independent output message and transforms it into terminal-dependent form by calling MMU.

7 The application subsystem passes the output message to the Front End by a call to FESEND.

8 The application program returns control to the Subsystem Controller, passing a return code which indicates normal completion or an error condition.

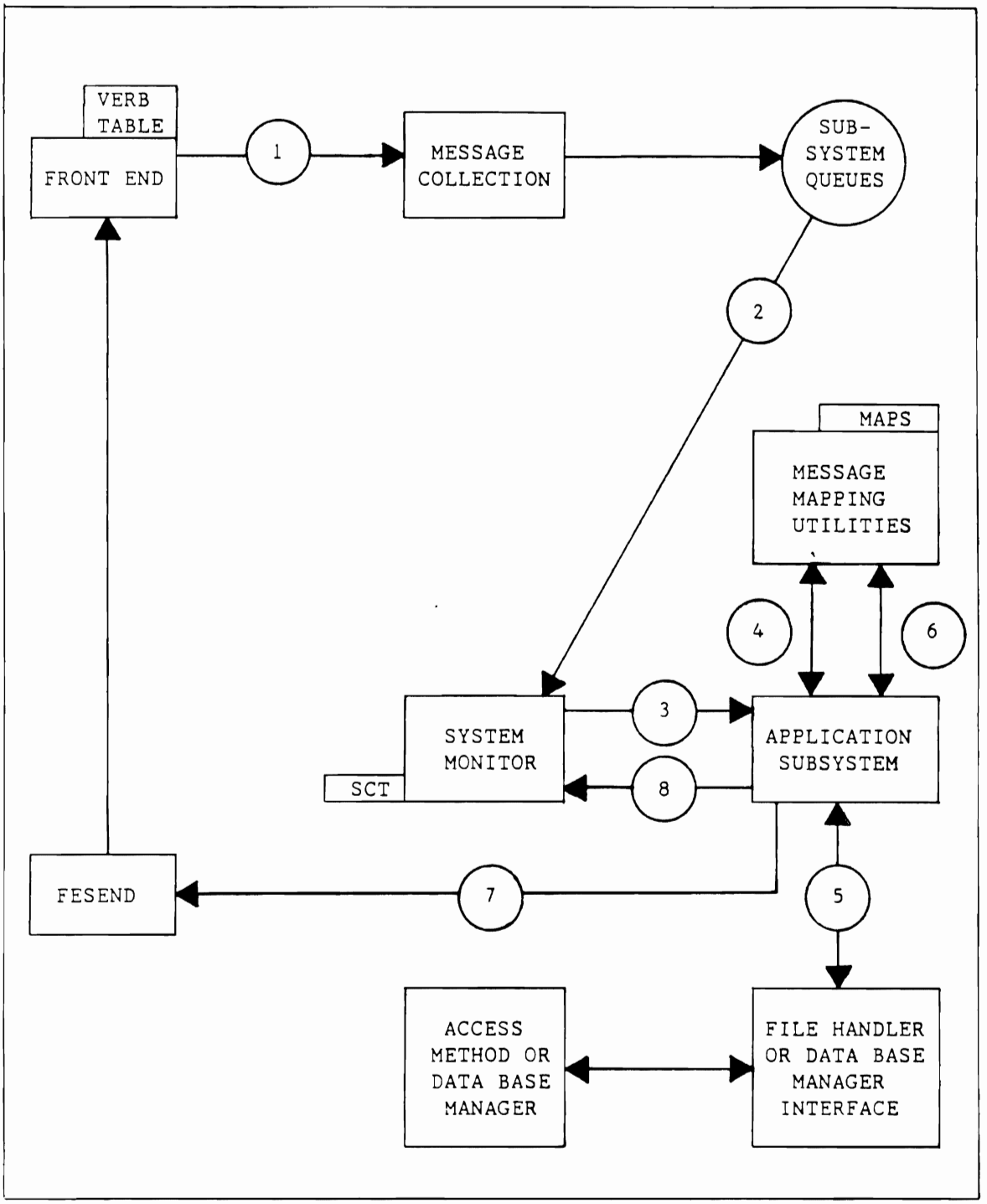


Figure 1. Intercomm Message Flow Using Message Mapping Utilities

1.3.2 Intercomm Message Flow Using the Edit/Output Utilities

In Figure 2, the numbered arrows in the diagram correspond to the numbered paragraphs below:

1 The Front End reads an input message and prefixes a 42-byte control header to it. The header contains routing information, time, date, originating terminal, and message length. The message is then queued for application program processing by Message Collection.

2 The Subsystem Controller schedules the application program and retrieves the message based upon the Subsystem Control Table scheduling criteria.

3 The Edit Utility is called by the Subsystem Controller and the input message is edited according to the Edit Control Table (ECT).

4 If editing is not successful, that is, invalid input data, the Edit Utility creates an error message for the originating terminal and queues it for the Output Utility by calling Message Collection. The input message is cancelled.

5 If editing is successful, the edited message is passed to the application program.

6 The application program performs message processing logic, requesting I/O service functions from the File Handler or DBMS interface.

7 The application program creates an output message and queues it for the Output Utility by calling Message Collection.

8 The application program returns control to the Subsystem Controller, passing a return code which indicates normal completion or an error condition.

9 The Subsystem Controller schedules the Output Utility and passes the output message to it for terminal-dependent processing.

10 The Output Utility performs formatting, if specified in the header, according to entries in the Output Format Table (OFT), then passes the message to the teleprocessing interface program (FESEND). FESEND passes the output message to the Front End (places the message on the terminal output queue).

11 FESEND returns to the Output Utility.

12 The Output Utility returns to the Subsystem Controller.

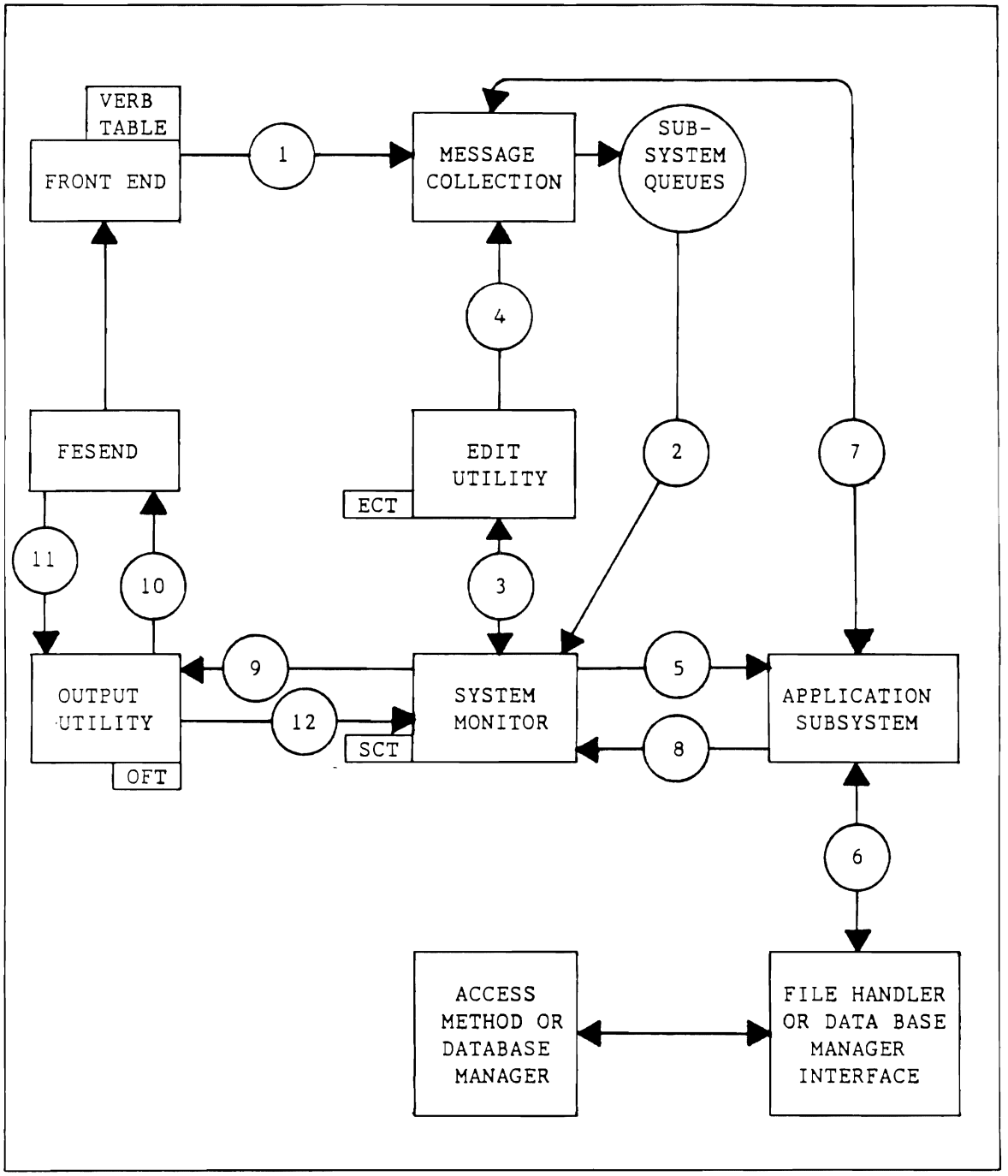


Figure 2. Intercomm Message Flow Using the Edit/Output Utilities

1.4 SINGLE THREAD VS. MULTITHREAD PROCESSING

In a multithread environment, the previously described message flow is carried out concurrently for many messages. For all practical purposes, the maximum number of messages in process concurrently is limited only by the region or partition size. During I/O, a reentrant application program can begin processing another message, or another program may process still another message. A message is processed as completely as possible until the application subsystem voluntarily gives up control via a file I/O or data base access request, for example. Therefore, at any one time, many messages may reside within Intercomm at various stages of execution.

1.4.1 Single Threading

Basically, a single thread system processes only one message at a time, that is, each function is performed sequentially on a message, from the first function to the last function, before another message begins processing. Under single thread processing, one task must be completed before another is started.

Figure 3 shows the execution of tasks in a typical teleprocessing application in a single thread environment. Each part of the processing of every message is dependent upon the completion of the prior step for that one message. Therefore, each time I/O operations are performed, the entire on-line application is essentially nonproductive. The user can be processing a card-to-tape routine, an assembly, a batch processing job in the background, or listing on the printer, but in the critical area of on-line response, the computer is contributing nothing. If the computer facilities could be directed to the processing of other on-line messages during the I/O operation, a higher level of efficiency would be attained. Multithread processing allows the computer facilities to accomplish this.

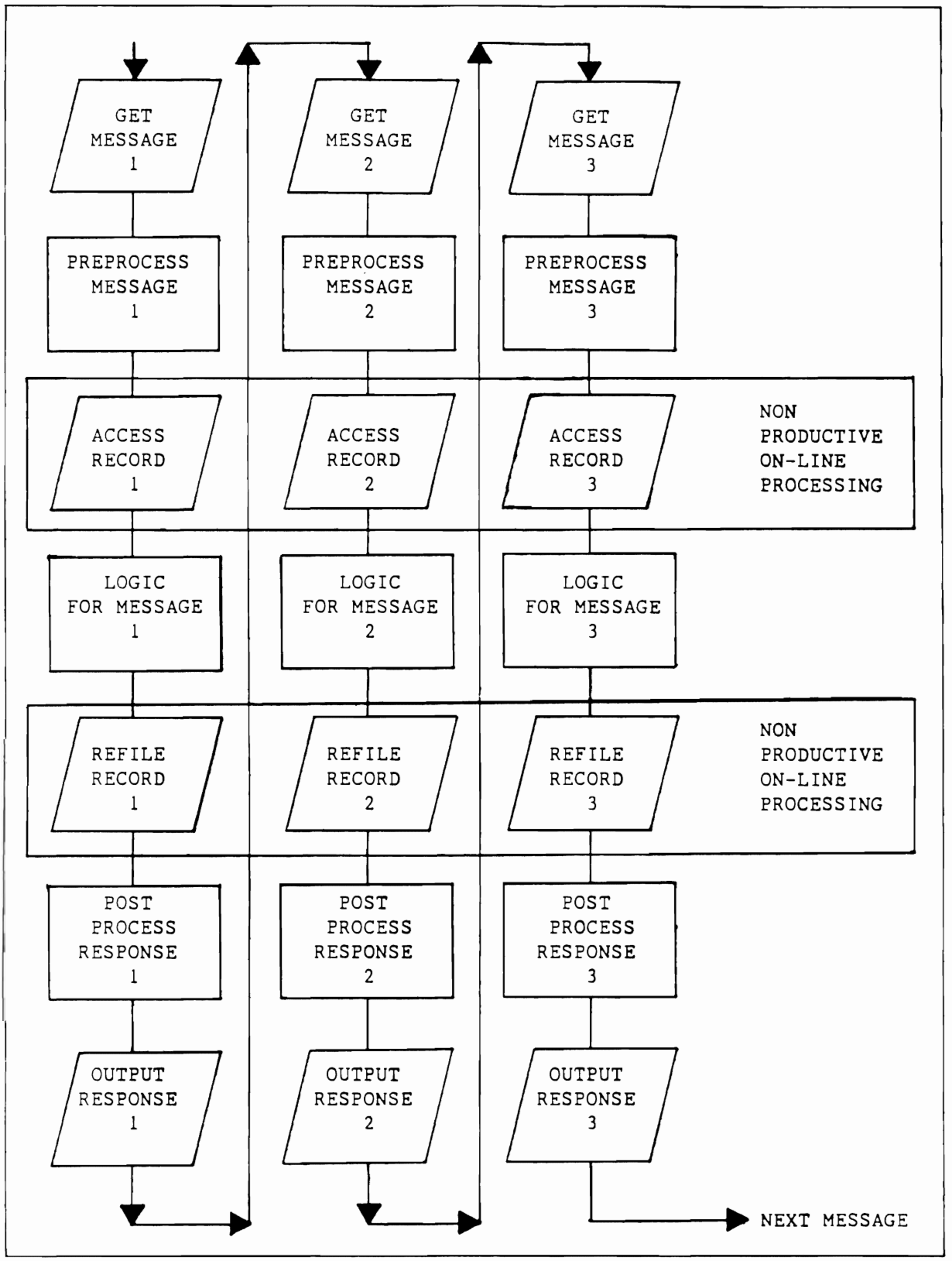


Figure 3. Processing On-Line Messages in a Single Thread Environment

1.4.2 Multithreading

Multithreading means that more than one message thread or path of program logic is concurrently active. Thus, in multithread processing, different messages can be processed in parallel, and many tasks can be performed concurrently. Each task involves a series of functions to be performed. Multithread processing takes place by starting one task, and performing the functions in that task. This continues until a function is reached that ordinarily would put the program (or partition or region) into a wait state, that is, a function such as file I/O. During the time that the I/O function responsible for the program wait state is being performed, another task can be started and executed by the same application program (if coded reentrantly) or by another application program, until that second task requests I/O or other wait-type functions. At this point, yet another task can be started. As Intercomm is scheduling the processing of all these tasks, it is able to pick up processing on the first task when its I/O or wait-type function is completed, finish processing that task, and then return to the second task, and so on.

Intercomm treats those portions of a program between I/O operations as separate program segments. This division of the application module into segments allows Intercomm to begin and end processing of any one segment without completing the processing of the entire message response.

The time required to communicate with on-line terminals, and the time required for the execution of application logic for each message processed, is relatively small in relation to the time required for I/O operations. Without multithread processing, a large amount of processing time would be wasted. This is due to the fact that while the on-line program is awaiting completion of I/O associated with each message, teleprocessing throughput is essentially zero.

Figure 4 depicts message processing in a multithread environment. Multithread processing is characterized by concurrent execution of application programs, and by overlapping the execution of segments or logical steps of different programs. For example, whereas the application partition was idle during the retrieval of record 1 in Figure 3, here it is possible to continue and overlap processing by getting message 2, preprocessing it, and initializing a read for on-line record 2. The user can have many messages executing within Intercomm, each at various stages of processing. Each message is capable of having an additional processing step performed on it as the CPU time or on-line data becomes available. But the next segemnt in a cycle of processing for one particular thread is not performed until all necessary prerequisites for its execution have been met.

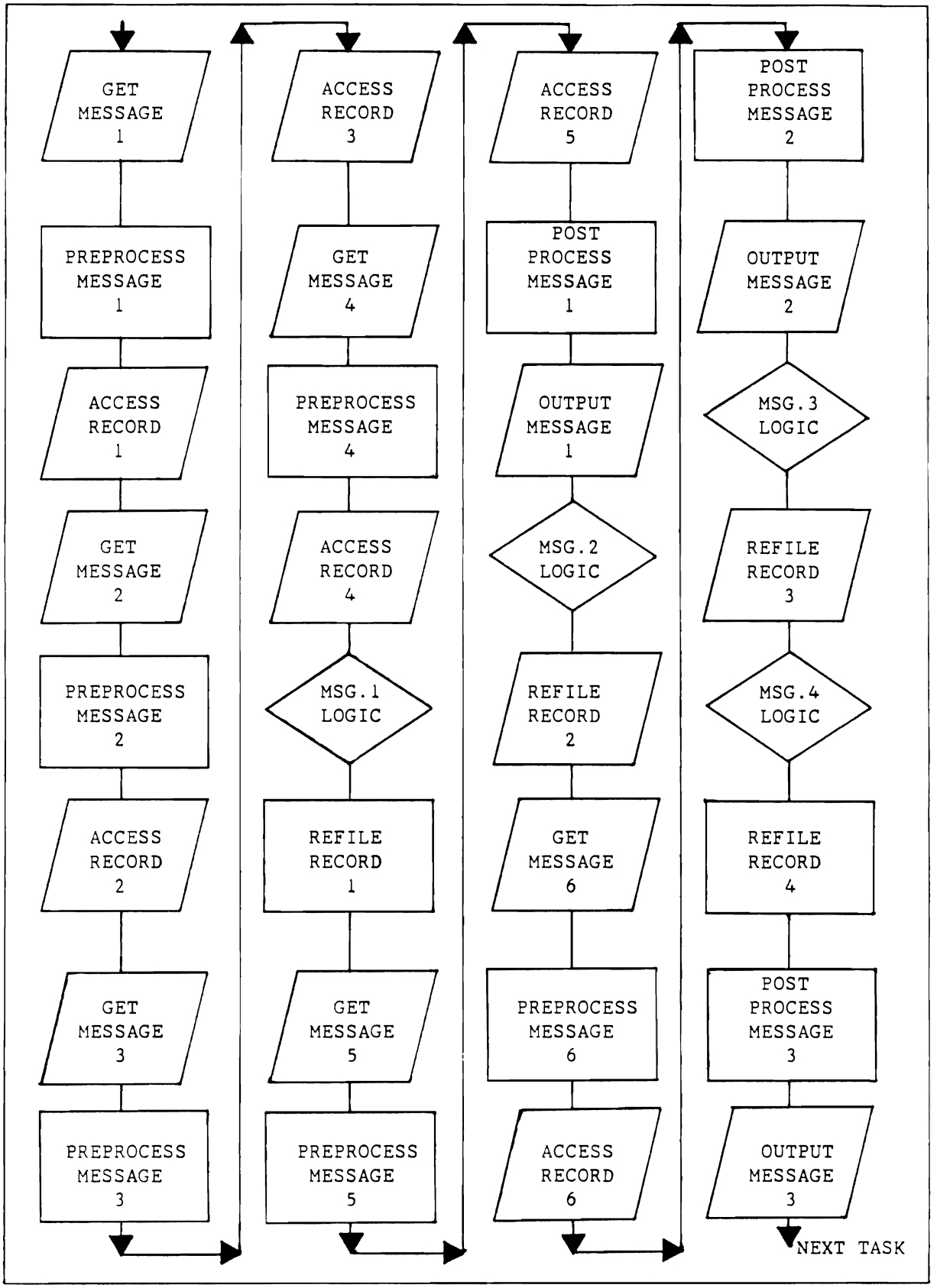


Figure 4. Processing On-Line Messages in a Multithread Environment

In Figure 4, messages are handled on the multithread segment basis. Each message is processed for as long as the computer can dedicate itself to that message without waiting for the completion of I/O operations. When processing for any message must be suspended for I/O, Intercomm initiates processing for any other message that is not in the similar situation of requiring I/O operation completion before continuing. In this manner, many messages are processed concurrently. Many logical threads of various message processing subsystems are operative at any one time.

Figure 4 represents the same period of processing time as in Figure 3. In the single thread environment of Figure 3, three messages are fully processed in a certain time interval. In the same time interval under a multithread environment, those same messages have been completed; but there are three additional messages within the computer, each at some point in the required processing cycle. These additional messages are started during the time that operations in the teleprocessing region are suspended for the I/O operations on the first three messages.

It is this ability to handle messages by multithread processing, utilizing the CPU during I/O operations, which provides the Intercomm user with fast response time and high volume throughput.

1.5 SPECIAL FEATURES

The basic Intercomm system consists of all the components necessary to implement an efficient on-line teleprocessing environment. These include the utilities (Message Mapping, Edit/Output, Change/Display), the BTAM Front End (line and device control), the Subsystem Controller, Dispatcher, File Handler, Resource Management, Message Collection and Retrieval, and the Store/Fetch facility which allows storage of data in core or on disk (a scratch pad facility).

In addition, a great many other features which are extensions to the basic system are available to the user. These are termed Special Features and are available as extra cost options. They are listed in Figure 5 and described in later chapters of this document. Each Special Feature is documented in a separate publication. Intercomm may be purchaced as a bundled system which includes the basic system and the Special Features.

Feature	Description
Autogen Facility	Generates MMU macros on-line from sample screens input from a 3270 CRT.
Data Entry System	Provides a preprogrammed general purpose data entry/verification capability.
DBMS: ADABAS DL/I IDMS Model 204 System 2000 TOTAL	Permits use of the relevant DBMS by Intercomm application programs and/or the DBMS query language. In most cases, coordinated restart/recovery is provided by the DBMS vendor and/or Intercomm vendor.
Dynamic Data Queuing Facility (DDQ)	Allows applications to dynamically create, retrieve, and delete logical data sets and/or queues of messages on a single BDAM data set.
Dynamic File Allocation (DFA)	Permits an application to dynamically create and retrieve sequential data sets.
Extended Security System (ESS)	Comprehensive on-line implemented security based on sign-on user-id and password.
File Recovery	Provides for recovery of on-line data sets in the event of system failure.
Generalized Front End Facility (GFE)	Provides the basic structure for interfacing to nonstandard communications devices, for example, minicomputers.
Model System Generator (MSG)	Provides a working system based on user specs to model the eventual system.
Multiregion Support Facility (MRS)	Allows groups of application subsystems to execute in separate regions while the Intercomm Front End resides in a Control Region.
Page Facility	Allows terminal operators to browse through a multiscreen output message.
SNA/VTAM	Provides Front End SNA support via the VTAM access method.
Extended TCAM Support	Provides Front End terminal interface via the TCAM access method.

Figure 5. Intercomm Special Features

1.6 SUMMARY

In summary, the advantages of Intercomm are:

- High Programmer Productivity

  Due to the many preprogrammed functions and the ease of application programming

- Low System Overhead

  Achieved in both storage and CPU cycle utilization through tuning the system to meet the user's needs

- High System Throughput

  Accomplished by this extremely efficient system

- High System Integrity

  Due to the maturity of the product and the many functions provided in this area

- Future Growth

  Easily accommodated because the system is modular; the product's past record is indicative of future expectations

- Vendor Support

  Assured by the vendor.

Chapter 2

INTERCOMM SYSTEM COMPONENTS

2.1 GENERAL DESCRIPTION

Intercomm appears to the operating system as a single batch job. It is executed through standard Job Control Language (JCL) conventions in a step which specifies "...EXEC PGM=INTERCOMM..." and contains the requisite DD statements for the BTAM devices and data files which are to be used by the Intercomm region or partition. Intercomm is not, however, a single program. It is actually a group of programs which include Intercomm system level programs, vendor-written application programs, and user-written application programs.

Intercomm components are logically divided into Front End and Back End categories. The Front End components are concerned with the communications functions of the system, while the Back End components are concerned with message processing. The Intercomm system and application-level software provided by the vendor is configured and used based on a individual installation's requirements which are specified in a series of tables. (See Section 2.8, "Intercomm System Tables.")

The user writes application programs (called subsystems) executed within the Intercomm region (or partition) to process individual messages. These subsystems are described to Intercomm by an individual Subsystem Control Table (SCT) entry for each program. The subsystems may be made resident (included in the Intercomm linkedit) or they can be dynamically loadable (loaded into the Intercomm region when there is a message for them to process). The user decides whether to make a subsystem resident or dynamically loadable based on considerations of response time and available main storage. (See Chapter 5, "Main Storage Organization.") The Intercomm Front End and system-level Back End components must be resident.

The major Intercomm components and their corresponding functions are listed in Figure 6. Detailed dexcriptions of component function and features are provided in the Operating Reference Manual.

Component	Function
Front End	Line and terminal control and interface to Back End
Subsystem Controller	Job management--schedules the start of a message processing subsystem
Dispatcher	Task management--CPU cycle allocation
File Handler	File I/O handling
Message Mapping Utilities	Input message editing and output message formatting
Edit Utility	Input message editing
Output Utility	Output message formatting
Log	Writes system and application log entries
Message Collection/Retriever	Queuing and dequeuing messages
Store/Fetch	Temporary (scratch pad) storage support
FECM	Front End Control Message processing
Language Interface	Support for high-level program languages
System Tuner	Provides statistics for tuning the system
Resource Management	Manages main storage, audits resource usage
Page	CRT page browsing capabilities
GPSS	General purpose subsystem which processes system control commands
Security	Table driven (Basic Security) or dynamically defined (Extended Security System)

Figure 6. Intercomm System Components

2.2 FRONT END

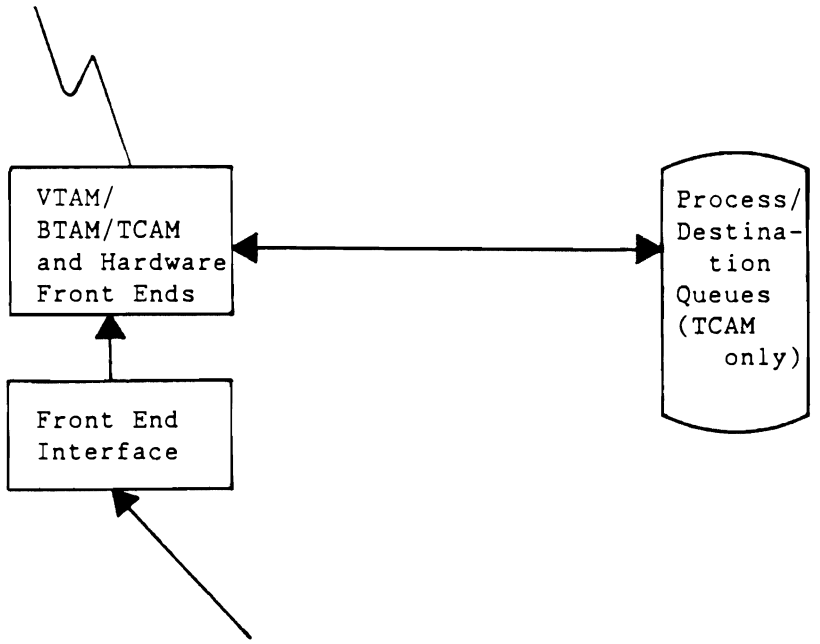
The Front End is responsible for the logical and/or physical management of the user's network. This includes monitoring and communicating with the terminals, receiving and sending messages, accomplishing code translations, providing queuing functions, checking message validity, and performing security checking. The physical line control is provided by the teleprocessing interface which uses BTAM or VTAM directly (in the Intercomm region), TCAM indirectly (in a separate region) via the user's TCAM Message Control Program (MCP), or directly by a hardware line control computer interface (GFE).

Regardless of the access method used, Intercomm is always logically responsible for the transmission, receipt, and control of data from on-line local or remote teleprocessing terminals. The Intercomm teleprocessing interface, therefore, consists of the line and terminal definitions related to the user's specific network configuration. It also contains all other program logic required for internal processing, error handling, control and routing of messages. The following teleprocessing areas are supported by the teleprocessing interface:

- Polling and receiving messages from terminals
- Addressing and sending messages to terminals
- Message translation--input and output
- Process and destination queuing
- Initiating corrective action in teleprocessing error situations
- Rerouting of messages for non-operational terminals

When using Intercomm's teleprocessing interface, all teleprocessing services such as polling, reading and writing are not logically connected to the application program. This transparency of the interface to the user's application program facilitates changes or additions of new terminals. Any required changes are made to the teleprocessing interface of the monitor with no impact on the user's program. Likewise, it is much easier to add new programs and applications. An on-line application program can be developed in any language, with the type of input or output device being transparent to the programmer. Figure 7 depicts the Intercomm Front End and its functions.

If BTAM or VTAM is used as the teleprocessing interface, all Intercomm functions and message processing programs under monitor control are located in a single region/partition. This facilitates minimal storage and operating system overhead to achieve a given throughput capacity. Intercomm totally controls the flow of work within that partition or region. The Intercomm teleprocessing interface is generated by conditional assembly for each user based upon the specific terminal network configuration defined via table entries.



- Communications Support:

  Software: BTAM, VTAM, TCAM
  Hardware: Generalized Front End (GFE) Special Feature has been used to support CDC/M1000, PDP/11, Periphonics, SUPER NOVA, Varian, and other Front End communications processors.

- Terminal Support:

  Nearly all IBM asynchronous, synchronous and SDLC devices, such as 2260, 2741, 2780, 3270, 3600, 3790, CPU-CPU, and all plug-to-plug compatible devices.

  Also many non-IBM terminals, such as Teletype and Dataspeed 40

- Transaction validation--verb verification

- Security Support:

  Operator/Terminal/Transaction, or Extended Security System (ESS) which controls access to system resources: regions, subsystems, verbs, files and terminals as defined per user

- Terminal Simulation

Figure 7. Front End

When Intercomm is implemented in conjunction with a user-supplied and generated TCAM Message Control Program, all of Intercomm operates in the standard manner. The support for TCAM depends on Intercomm's ability to accept input from and route ouput to TCAM process and destination queues via its standard Front End. This feature is especially important to Intercomm users who also use IBM's Time Sharing Option (TSO) with TCAM as its Front End. The use of TCAM allows the sharing of terminals between the time-sharing function and on-line applications controlled by Intercomm. VTAM also provides terminal sharing among many TP systems.

Intercomm's TCAM support is also designed to provide the Intercomm user with access to terminals controlled by TCAM concurrently with terminals controlled by Intercomm's BTAM or VTAM Front End. All features available in the standard Intercomm Front End are available for terminals accessed by TCAM. Such features as 3270 AID processing, terminal up/down commands, terminal lock and unlock commands, start/stop line commands, etc., normally unavailable to TCAM users, are supported. Additional user code in the TCAM MCP is minimal. The user does not have to edit the messages to the process queues or from the destination queues handled by Intercomm. This substantially reduces the code required in the Message Control Program, which otherwise would be required to add and delete Intercomm headers.

In addition to BTAM, VTAM and TCAM, an Intercomm user may also opt for a hardware Front End system to handle the teleprocessing. In this case, any one of the following can occur:

- The BTAM portions of the Intercomm interface can be directed to interface with a high-speed line that would connect the two systems (the Front End system and the actual message processing system).

- The BTAM logic can be replaced by unique channel program logic related to a direct channel-to-channel interface between the two systems.

- An interface program can be developed to read from and write to the Front End as if it were a tape drive (several hardware Front Ends are currently designed to look like a tape control unit in their System/370 interface logic).

The interface with a hardware Front End can also be accommodated via the Generalized Front End (GFE) Special Feature of Intercomm.

More extensive descriptions of the BTAM and VTAM Front End facilities are included later in this document. For a detailed description of Intercomm's TCAM support, refer to the TCAM Support Users Guide.

2.3 QUEUE MANAGEMENT

The Intercomm Message Collection/Retriever routines are responsible for queuing and dequeuing of input messages, internal messages, and output messages. (See Figure 8.) The queues constitute waiting lists of messages in the Intercomm system. The queues allow for the separation of teleprocessing functions and message processing functions, that is, asynchronous processing, absolutely requisite to the efficiency demanded by the Intercomm system.

Message Collection collects, logs, validates and queues all messages for the user. When an input message is passed from the Front End, or an internal message is sent from one application program to another, Message Collection is invoked to queue the message onto the appropriate subsystem queue via a core and disk queuing capability. A system table (Subsystem Control Table; see Section 2.8, "Intercomm System Tables") is accessed to determine whether a message can optionally be core queued or must be written out to disk.

The size of both the core and disk queues are variable by program, depending on transaction volume. Once the maximum number of messages are core queued for a given application program (waiting their turn to be processed), the overflow is automatically written out to the disk queue. The disk queue is a logical entity by subsystem. However, it can exist on one or more BDAM data sets, with wraparound reusable blocks, and provides as well for variable blocked records.

Message Retriever is invoked by the Subsystem Controller to retrieve or select a message for processing by the appropriate subsystem. All input sources, that is core and disk queues, are checked by the Retriever for messages. When the Retriever accesses (dequeues) a message, it passes the address to the Subsystem Controller for processing by the application program.

Message Collection is also called by the Back End to Front End Interface Program (FESEND) to place an output message on a terminal queue (core and/or disk). The Retriever is called by the Front End message transmission routines to dequeue a message for output.

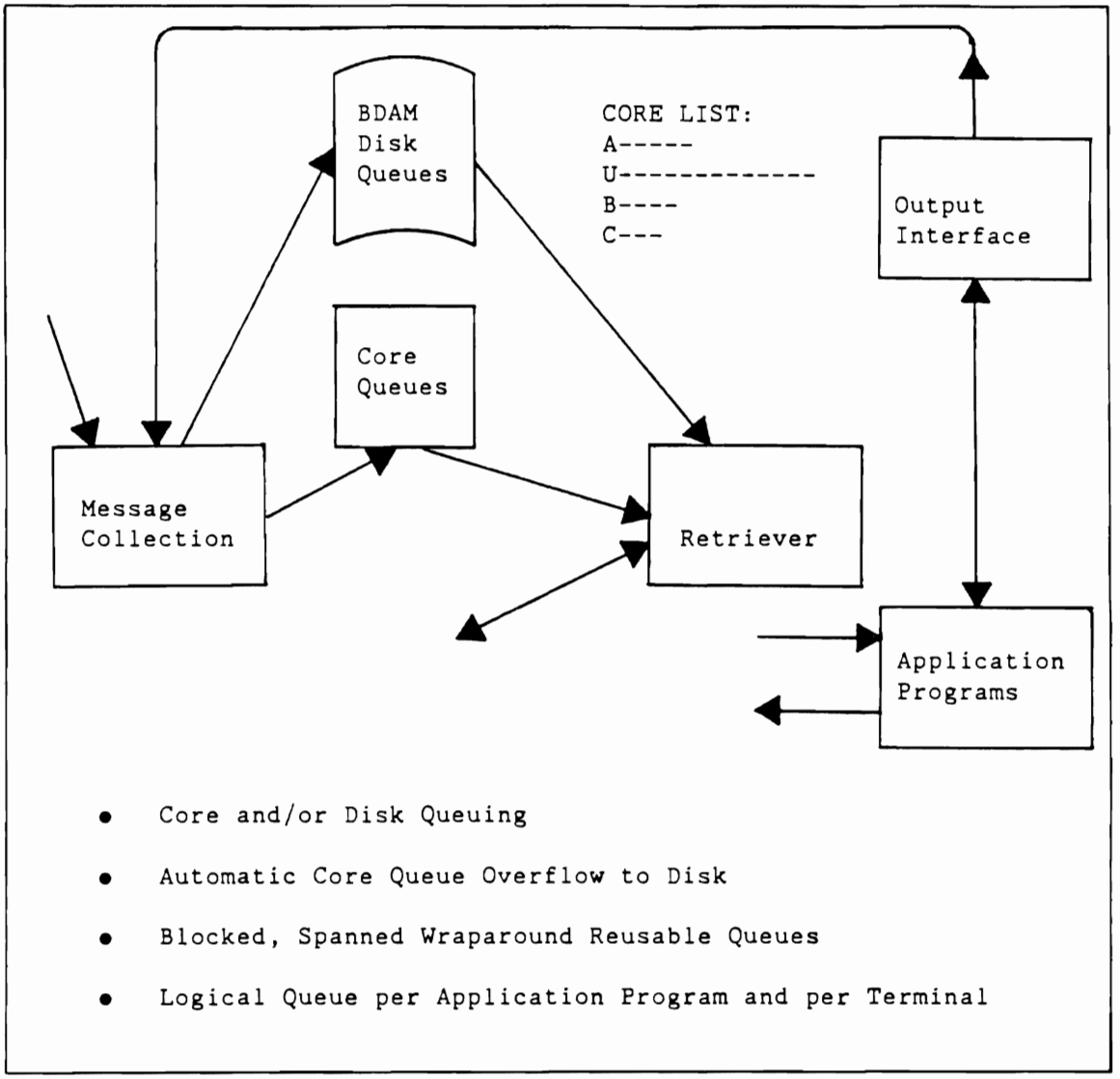


Figure 8. Queue Management

2.4 SUBSYSTEM CONTROLLER

The Subsystem Controller provides the job management capabilities of Intercomm. Figure 9 shows the component functions and their relationship to the applications.

In general, the Subsystem Controller performs the following:

- Interacts with the teleprocessing interface to control internal message traffic and to schedule, load, and activate application programs (subsystems) based on resource availability/demands, priorities and optional queue aging.

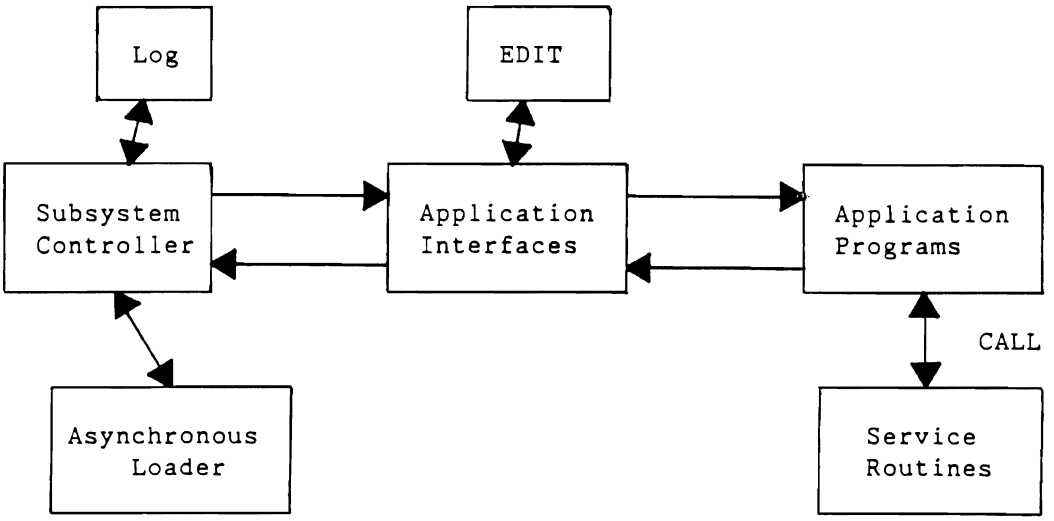
- Provides maximum efficiency by bypassing the OS/VS Control Program in regard to message/task management. The Subsystem Controller, in conjunction with the Dispatcher, supervises scheduling and loading of all application subsystems. These may be resident, dynamically loaded into a reserved area of the dynamic subpool managed by the Subsystem Controller, or loaded according to a planned overlay structure. Repetitive loading of subsystems is minimized by queue analysis and is overlapped with concurrent message processing.

- Selects incoming messages from queues (based on the priority assigned to the corresponding program) and directs them to the proper application program for processing. It also provides for the switching of original messages received by the system, and internally developed messages from within the system. This program-to-program passing of messages is accomplished through both core and reusable disk queuing techniques automatically provided by the Message Collection/Retriever facility.

- Maintains a system log for statistics and restart/recovery.

- Dynamically loads and deletes nonresident subsystems as required during execution of Intercomm. This function is performed by Asynchronous Loader routines. Once loaded, reusable subsystems will remain resident until the concurrent user-specified message processing limits are reached or message traffic for the subsystem ceases.

The Subsystem Controller references the user-generated Subsystem Control Table (SCT) entry for each program/subsystem under its control. (The SCT entry for each subsystem details its characteristics, queue specifications, and scheduling considerations to the Subsystem Controller.) Since the Subsystem Controller uses these SCT entries for its job management functions, it is considered a table-driven Intercomm component.



SUBSYSTEM CONTROLLER

- Optimization/Throughput
- Table-Driven
- All Logging and Message Restart
- Asynchronous Program Loading
- High-Level Language Interface
- Resource Enqueuing

APPLICATION INTERFACES

- Optional Call to EDIT for Message Formatting
- Formats Entry Parameters
- Obtains Dynamic Storage

APPLICATION PROGRAMS

- Multithread Operation
- Standard OS/VS Compilers
- Conversational Mode
- Transparent Operations
- Use Standard Call Interface

Figure 9. Subsystem Controller, Application Interface, Application Program Functional Relationships

2.4.1 Logic Overview

Essentially, the Subsystem Controller acts as the traffic coordinator for all work executing under the monitor in the multithread environment. (See the description of Intercomm message flow in Chapter 1.) All program execution is provided under control of the Subsystem Controller.

The Subsystem Controller utilizes the Intercomm Message Queuing facility during message processing. Message Retriever is invoked by the Subsystem Controller to determine if a message is available for processing (on the queue) by the appropriate subsystem. If a message is present, the corresponding program status indicator is checked to determine whether that program is resident. If not resident, it is loaded into main storage by the Subsystem Controller's Asynchronous Loader routines. Loading of nonresident subsystems may be dynamic, that is, into an area of dynamic core, or according to a preplanned overlay structure, into an area linkedited as an overlay segment. The SCT is checked for the maximum number of messages that may be processed concurrently by the program and, the number of messages already being so processed. If the current message causes the maximum to be reached, the Subsystem Controller prevents the initiation of any further messages for concurrent processing by the program until the number of concurrent messages in progress is decreased.

Prior to passing a message to an application program, the Subsystem Controller creates a system log entry indicating that the message has started processing through that subsystem. Then the Subsystem Controller issues a call to the program for which the message is intended and passes the address of the message to the program. This transfer of control initiates message processing. The application program will then have control of the CPU until an I/O operation, a time interval delay, or any other call causing a processing delay is executed.

A RETURN to the Subsystem Controller is issued when message processing is complete. The SCT entries are then updated and, to indicate completion of the message that was in process, the Subsystem Controller writes a completion entry on the system log.

Whether the message processing program is resident or dynamically loaded (it can be reusable or non-reusable), as specified in the SCT, the Subsystem Controller allows for the startup of a new thread or for an already existing one to regain control. Both events occur when control is passed to the CPU scheduler, the Dispatcher. If the program type was an overlay, the Subsystem Controller calls in the next overlay according to conditions concerning the current overlay. (See Figure 10 for Subsystem Controller logic.)

(For coordination of Subsystem Controller and Dispatcher functions, see Section 2.5, "Dispatcher--Thread Management.")

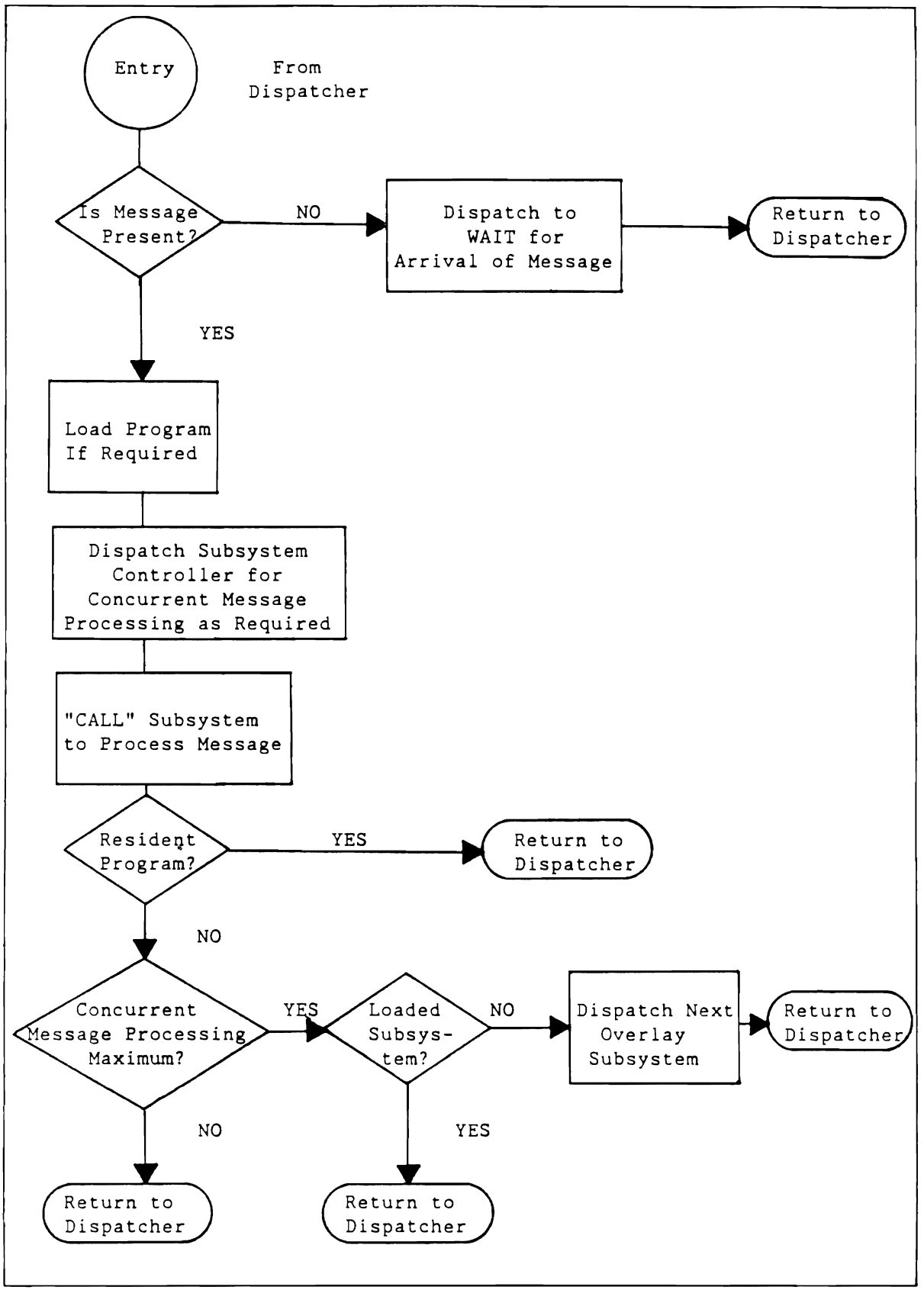


Figure 10. Subsystem Controller Logic

2.4.2 Dynamically Loaded Subsystems and Overlay Management

With Intercomm's dynamic load facility, nonresident subsystems can be loaded on demand into the dynamic subpool. Once loaded, a subsystem remains resident until a maximum number of messages have been processed, or until message traffic has ended. The message limit is specified in the SCT. Alternately, loading of subsystems may be controlled by the Intercomm Overlay Management scheduling facility. In this case, subsystems are linkedited as overlay region segments and loaded according to a preplanned structure and sequence.

In both cases, the sequence of subsystem loading is optimized by the Subsystem Controller. Once loaded, priority for CPU use during message processing by an application subsystem is determined by the Dispatcher based on SCT entries.

The dynamic load facility allows the user to correct subsystems and utilize a new copy of the program without terminating the Intercomm job. If load list specifications by BLDL are included, the system must be notified of the existence of this new load module by a LOAD command. Replacing an overlay region subsystem load module entails a linkedit of the entire system.

In summary, the Subsystem Controller exercises control in conjunction with:

- Subsystem Control Table entries for each application program or Intercomm message processing program
- Message Collection routine which collects, logs, validates, and queues all messages
- Retriever routine which acquires messages from Message Collection's core or disk queues for processing by the appropriate module
- Restart/recovery routines

2.4.3 Resource Enqueuing

Resource enqueuing is a technique that provides an additional scheduling dimension to the Subsystem Controller. Selected subsystems can be specified as users of certain selected resources. A separate specification denotes the maximum number of concurrent users of that resource. The Subsystem Controller uses that resource limit as a system-wide thread limit against that resource. For example, assume that tuning information indicated that it was counterproductive to start more than three messages that use a certain file. Ten programs are users of the file and each has an individual thread limit of three. However, to insure a system-wide thread limit of three, each subsystem is denoted as a user of that file (resource) and the resource limit is specified as three. Thus, individual subsystem thread limitation may be overridden. Another significant use of resource enqueuing is to separately specify subsystem residency from multithreading. For example, each Subsystem Control Table is specified as a user of a unique resource (a different resource per SCT). If the resource limit is one but the subsystem thread limit is five, then the subsystem becomes single threaded, yet it will stay loaded (dynamically or in an overlay) to process up to five queued messages. Thus, use of this feature may substantially reduce overlay or dynamic program loading and significantly improve system performance.

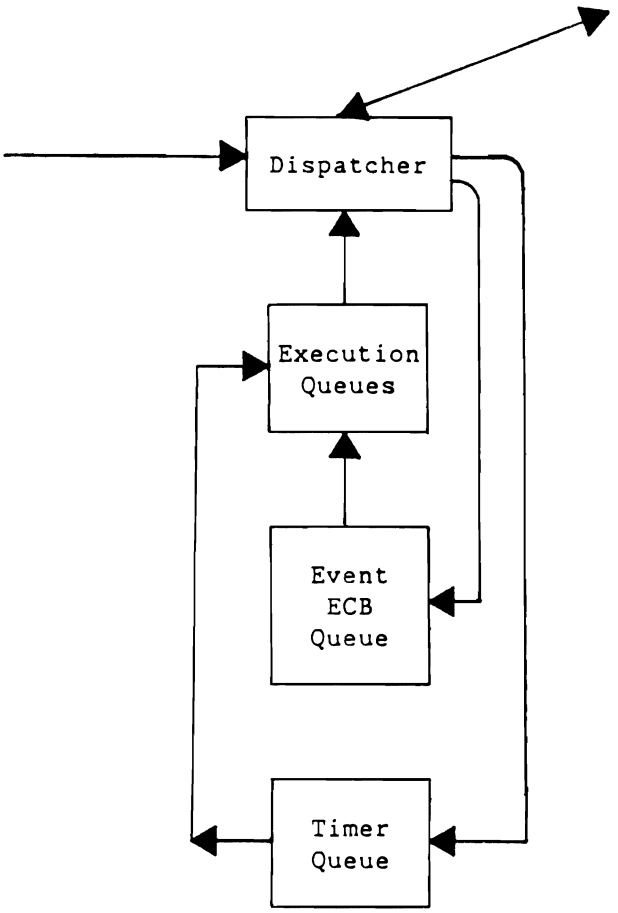
2.5 DISPATCHER--THREAD MANAGEMENT

The Dispatcher is the system component that provides thread management. This is accomplished by supervising the asynchronous parallel execution of any number of related or unrelated programs within the Intercomm region or partition. That is, the Dispatcher allocates available CPU time efficiently among all active threads under Intercomm control.

Multithreading occurs when more than one thread is processed in parallel. The Dispatcher coordinates multithreading or the processing of several messages concurrently. This is done in conjunction with other Intercomm components, such as the Subsystem Controller, Message Collection/Retriever, File Handler, DBMS Interface, and the Front End.

The Dispatcher schedules CPU time based upon the entries on its queues (lists) of segments awaiting execution. A thread is a set of functions which completely process a message, and a segment is a logical group of those functions. As previously discussed in Chapter 1, multithread processing means that more than one thread of program logic is active. The queues of thread segments are the means of scheduling multithreaded use of the CPU and represent the separation of functions performed on a message. This separation allows a segment of another thread to gain use of the CPU while the first segment of the current thread is awaiting completion of an operation that otherwise would suspend further processing. That is, the Dispatcher schedules the CPU for another segment when a program is in the wait state. Many messages, therefore, may reside in the system at various stages of execution. Each message awaits completion of the next segment in its cycle of processing. With the Intercomm multithreading facility, processing time is efficiently utilized during waits and throughput is thereby greatly enhanced.

Thread segments are placed on the Dispatcher queues according to whether they are presently executable, event-dependent, or time-dependent. (See Figure 11.) There is one event queue, one timer queue, and four execution queues. All units of work placed on an event or timer queue remain queued until the event transpires or the duration expires. They are then, depending upon assigned priority, transferred to the execution queues. Threads are dispatched from an execution queue according to first-in/first-out sequence within each priority level. The four execution queues correspond to the highest-lowest subsystem priority codes of 0, 1, 2, 3.



RESOURCE MANAGEMENT:

- Timer Queue:
  - Delayed Processing
  - Time-Dependent Facilities
- Event Queue:
  - Wait for I/O Completion
  - Intertask Switching

THREAD MANAGEMENT:

- Execution Queues:
  - Maintained by Priority
  - Contain Completed Entries from Other Queues

Figure 11. Dispatcher--Thread Management

The following requests for use of the CPU are serviced by the Multithreading Priority Dispatcher queuing facilities:

- A request for a unit of work to be placed on a specific priority execution queue and executed as soon as priority permits, for example, the Subsystem Controller may request such queuing to start up another message for a program; Front End modules may request line servicing operations
- A request for a unit of work to be placed on a timer queue and executed upon a specified duration of elapsed time
- A request for a unit of work to be placed on an event queue and executed upon the completion of a specified event, that is, I/O operations initiated by the File Handler waiting for completion
- A request to delete a previously queued request
- A request to terminate control and initiate execution of the highest priority unit of work awaiting execution

A thread is placed onto one of the Dispatcher queues to await execution. Execution is dependent upon a DISPATCH macro instruction specifying the priority of the thread, and the address of the routine to be dispatched. Following the execution of the DISPATCH macro, that is, a thread segment is dispatched, the Dispatcher returns to the program that issued the request, or exits from the program and dispatches from its work queues the next thread to be executed. Determining which route to take is based upon a parameter coded on the macro. (See Figure 12).

A special queue is provided for events completed within Intercomm; that is, events completed by an Intercomm routine, rather than by an operating system routine, such as I/O completion. Upon completion of these internal events, a work unit can be transferred immediately from the internal queue to the execute queue. The result of the use of this special queue is that a significant amount of CPU time is saved by the Dispatcher, because it no longer has to scan a long list of waiting events to see which ones can be scheduled. They are transferred directly to the execute queue when the event completes. This is accomplished by having the waiting task specify in its Dispatcher call that the event will be completed internally and should be placed on the special internal wait queue. The task that detects the event completion must call a special Dispatcher entry point which will accomplish the transfer of the waiting task to the appropriate execute queue immediately.

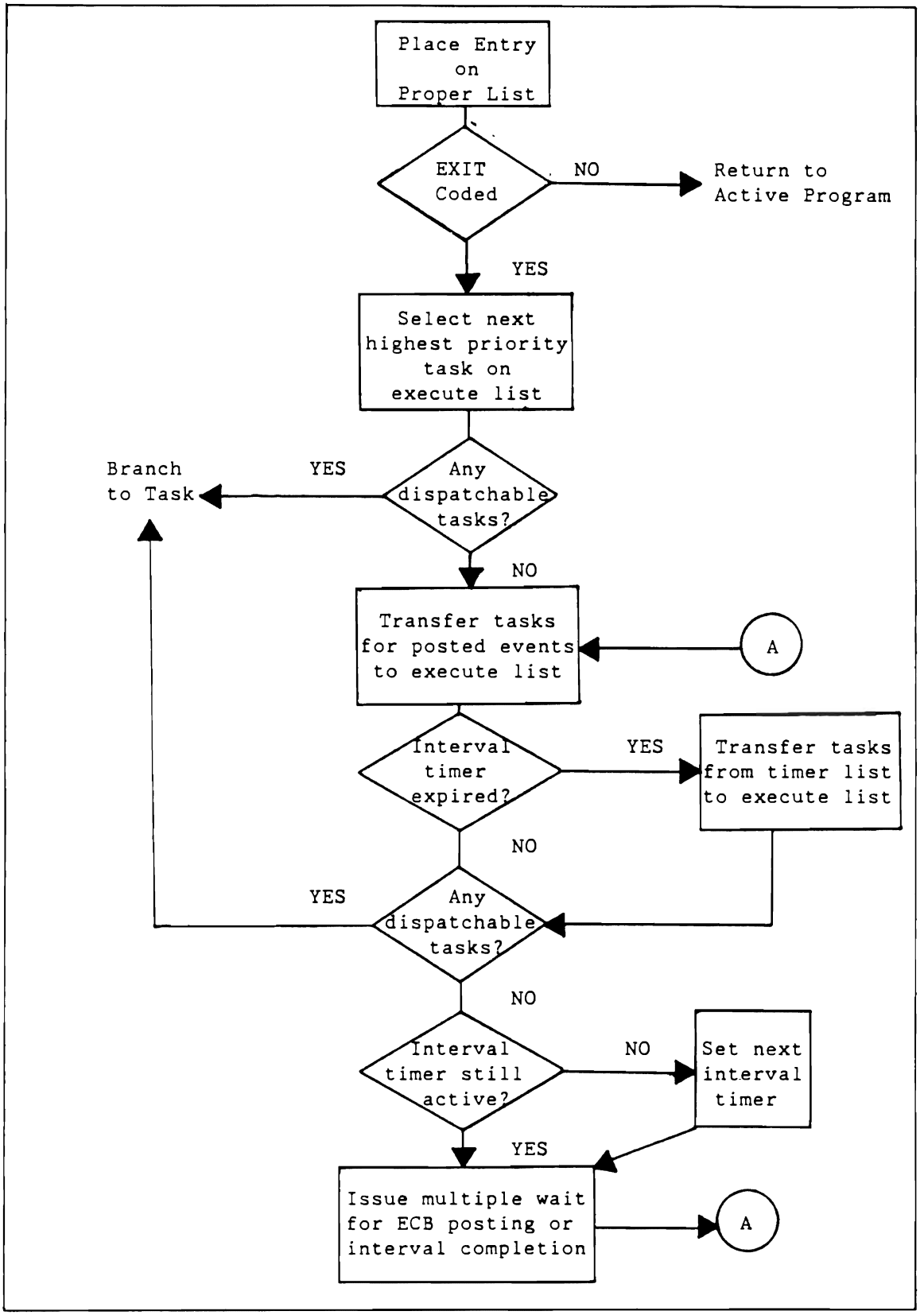


Figure 12. Dispatcher Logic

2.5.1 Dispatcher and File Handler

The Dispatcher is utilized by the Subsystem Controller and the File Handler to automatically achieve, as an example, dynamic program management. For instance, when the File Handler initiates an I/O operation, it places an entry on the event queue via a DISPATCH macro and relinquishes control to the Dispatcher. The Dispatcher is consequently free to schedule any other thread in the system, depending upon the status of its execution queues. Eventually, when the File Handler I/O completes, the Dispatcher recognizes completion; the File Handler entry becomes an executable thread segment when it reaches the top of its queue for the particular priority involved. When the entry is removed from the execution queue, the Dispatcher returns control to the File Handler which then returns control to the program requesting the I/O operation.

Therefore, during I/O operations, CPU time may be utilized to continue program execution within the Intercomm system.

2.5.2 Dispatcher and Subsystem Controller

Scheduling of message processing by the Subsystem Controller is based upon arrival of messages, an event within the Intercomm system. Thus, the Subsystem Controller dispatches itself to wait for the arrival of messages and is given control by the Dispatcher when executable. Similarly, the Subsystem Controller waits for the overlapped loading of nonresident subsystems via the Dispatcher.

2.5.3 Dispatcher and Time Control

The Dispatcher also provides for any number of timer requests to be outstanding at any time. As a result of this capability, user programs can be started based on time of day; routines can be awakened after the expiration of a requested time interval; or loops can be detected in a bad thread of an application program. If a loop is detected, the message in progress is cancelled and both the original terminal operator and the system or master control terminal operator are advised of the condition; Intercomm also snaps the program's registers and related areas of storage, and frees any physical record being held with exclusive control by the program.

2.6 FILE HANDLER--DATA MANAGEMENT

The Intercomm File Handler provides the data management facilities of the IBM 370 Operating System to all user application programs. All IBM access methods are supported with the exception of BPAM. Non-IBM access methods such as IAM are also supported. Furthermore, the File Handler provides those available facilities for high-level languages, and requires much less programming effort than direct coding in Assembler Language. File Handler services are requested by a standard subroutine CALL statement.

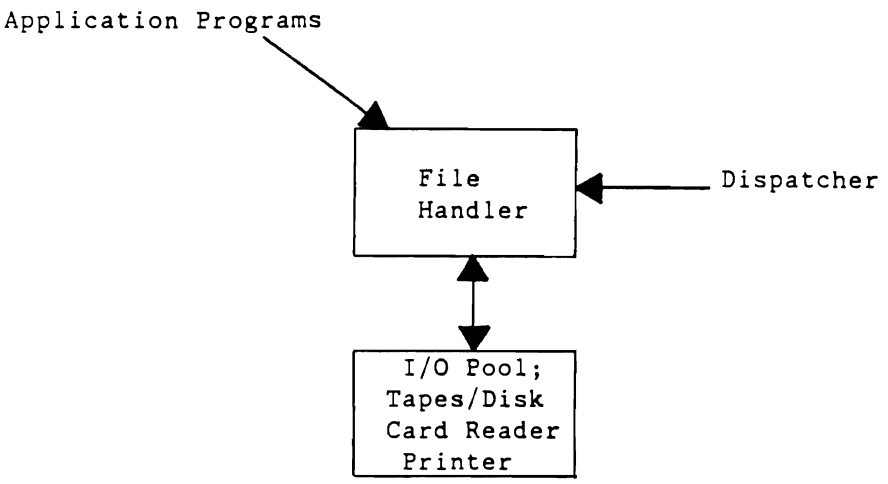
The following special facilities are supported with Release 9.0:

- VSAM Local Shared Resources Support--enables the user to specify a common shared buffer pool for selected VSAM files. This facility can cut down paging requests and wastage of virtual storage in systems where the use of VSAM files is extensive.

- B37 Abend Protection--Critical sequential output files on disk (including the Intercomm log) can be protected against an out-of-space condition by having an alternate data set specified for them. The File Handler will intercept the error and switch to the alternate data set. This both prevents the loss of critical data and guarantees continuous system execution because a B37 or D37 abend need no longer bring Intercomm down.

- Dynamic Deallocation/Reallocation Facility-- Under MVS and XA, Intercomm supplies on-line commands to deallocate a data set from Intercomm and to reallocate it at a later time. This enables the user to make an on-line data set available for off-line use without interrupting monitor execution. Any data set accessed via the File Handler is eligible for this facility.

All data access is centralized through the File Handler. Intercomm removes all I/O programming responsiblities from the programmer and places them under the control of the File Handler. The user is only required to perform external data management (data set organization and processing techniques); internals are handled entirely by Intercomm. (See Figure 13.)



- Supports: BSAM, QSAM, BDAM, BISAM, QISAM, VSAM. Also supports IAM, supplied by an independent vendor.
- Utilizes any I/O device supported by the OS/VS operating system
- Allows application programs running under Intercomm to output to JES SYSOUT data sets
- Generates ISAM control blocks based upon the specific use for the data set; that is, read-only, read with update only, or read, update, and add
- Provides for opening of data sets either at startup time or when actually required

Figure 13. File Handler--Data Management

As a result of centralizing all processing of on-line files in a single program, the File Handler effects the following:

- Eliminates duplication of I/O routines, control blocks, and buffers in application programs, thereby considerably reducing main storage requirements for control blocks, buffers and I/O areas.

- Substantially reduces program execution time by eliminating wasteful opening and closing (approxmimately 1 1/2 seconds) of data sets that would normally be required for each message processed. Data sets are normally opened only once per day.

- Reduces the amount of detailed understanding required of each programmer in utilizing the various access methods.

The same file records can be accessed concurrently by many different programs. Obviously, with centralized data access, simultaneous file updating could destroy file integrity. Therefore, the File Handler provides for exclusive control with direct and random access files.

With both centralized control and a dynamic file structure, files can be added or deleted from the on-line system on a day-to-day basis. This can be accomplished because the files are defined to the system entirely through the standard data definition (DD) card conventions. The File Handler constructs one central set of control blocks for each file, thus reducing storage requirements in individual message processing subsystems. The control blocks related to the DD statement defined data sets are created dynamically by the File Handler at Intercomm startup.

File Handler startup also checks to ensure that all disk files defined in the JCL actually exist on disk. An abend does not occur at the first access if a disk file does not exist; rather, this error situation is treated in the same manner as missing JCL; an error code is passed to the user who attempted to utilize that data set.

Comprehensive diagnostics for on-line security are provided as well as write-protection of master files. Security can be derived from operator sign-on under ESS or application program restriction of certain transactions to a terminal location. To obtain write-protection of master files, the DDname of the data set, as coded in the JCL, is specified on a control statement processed at startup. The File Handler does not permit updates to that file unless a system control command is entered to allow updates.

As an option, the File Handler will perform automatic error checking if special error recovery processing is not required by the application logic. In this case, the File Handler returns to the subsystem only if the requested operation completes successfully. If not, the File Handler cancels the related message and returns to the monitor. The originating terminal operator is then notified with a corresponding error message.

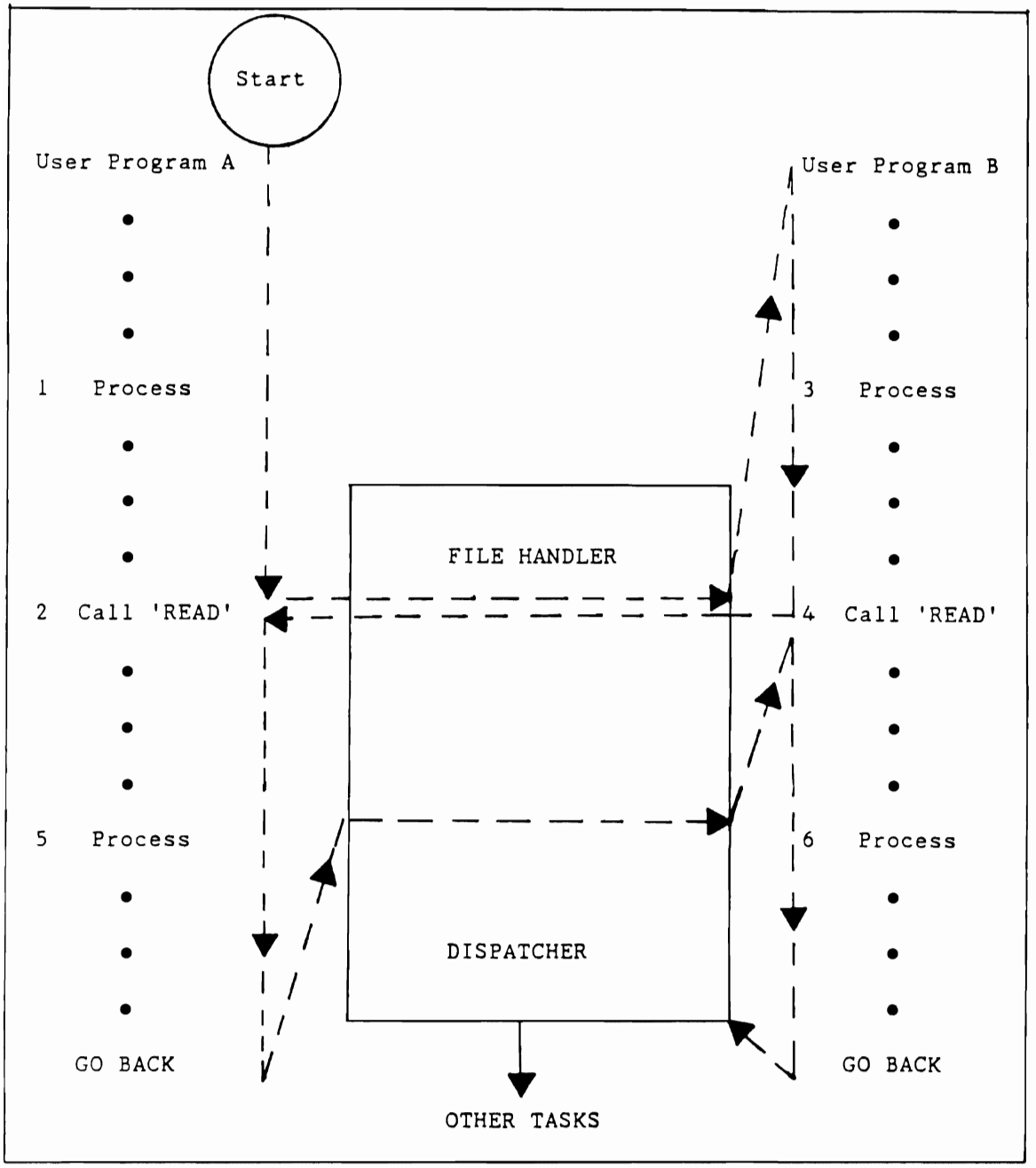


Figure 14. I/O Operations Using the Intercomm File Handler

The ultimate purpose of Intercomm is to control the concurrent execution of multiple on-line applications, based on user-defined priorities. Thus, Intercomm must have final control of which application or task is currently being executed, including I/O functions.

For an illustration of multithreading during an I/O operation, refer to Figure 14 where two programs, A and B, are both in operation under Intercomm in the same region. Program A has control and is processed at point 1. At point 2, Program A requests that an I/O operation be performed. Under normal circumstances, this operation would be initiated directly by the application program. Until completed, Intercomm would be placed in the wait state, and a lower priority region would gain control.

Under Intercomm, as depicted in Figure 14, the request for I/O is given by Program A, not to the operating system, but rather to the Intercomm File Handler. The File Handler makes certain determinations to insure that the I/O request on the file for Program A can be accommodated. It then initiates the I/O operation on the file. Instead of waiting for completion of the operation before continuing with Program A, however, Intercomm directs control to Program B, where processing begins at point 3. At point 4, Program B requests an I/O operation on its file. This operation is also received by the File Handler, as was the previous request for I/O from Program A. The File Handler initiates the desired operation on the file. If the I/O request for Program A has not yet been completed, and there were only two programs operating under control of Intercomm, processing would resume in a lower priority region. If three or more applications were executing under Intercomm, control would be passed to the processing phases of the additional modules.

Assuming I/O has been completed on the file for program A, Intercomm will determine that control can now be returned to Program A at point 5 to resume processing. When Program A completes processing, Intercomm directs control to Program B to resume processing. If either program wants to initiate additional I/O, the File Handler is given control once more. Intercomm determines when the requested I/O on the file for program B has been completed, and returns to Program B, where processing resumes.

The File Handler is a unique series of service routines. The control and scheduling of the various threads as they switch back and forth between I/O and processing are handled by the Dispatcher.

2.7 DATA BASE MANAGEMENT SYSTEM SUPPORT

Intercomm provides Special Features which allow users to access information maintained by the following data base management systems:

- ADABAS--a product of Software A.G. of North America, Inc.
- DL/1--a product of IBM Corporation
- IDMS--a product of Cullinet Corporation
- Model 204--a product of Computer Corporation of America
- System 2000--a product of S.A.S.
- TOTAL--a product of Cincom Systems

When recovery is coordinated with Intercomm, the data base is fully recoverable from any failure situation. (Refer to Chapter 11.)

Also available as a Special Feature is a Generalized Data Base Management System Interface whereby a series of programs supplies all but the specific data base logic necessary to provide data base access from Intercomm as well as data integrity against program and system failure. Although most application programs will be run under control of the Intercomm monitor, programs not under Intercomm's control, that is, batch programs, may require concurrent and/or overlapping use of the DBMS. Intercomm's interfaces are generalized to provide for utilization of the user's DBMS by batch off-line (non-Intercomm) programs as well.

2.8 INTERCOMM SYSTEM TABLES

Intercomm is a generalized on-line system and as such requires operating specifications tailored for each installation. This information is provided to the system in the form of tables which are defined using Intercomm macros which are described in Basic System Macros and/or facility descriptive manuals. An application programmer is usually not involved in defining the Intercomm tables except for application-oriented specifications for the utilities. Tables (by which the user specifies his unique requirements) exist for each of the following Intercomm functions:

- Front End Control

  -- Network configuration
  -- Valid transaction identifications

- Message Processing Control

  -- Subsystem specifications

- System Control

  -- Security functions
  -- Checkpoint/restart specifications
  -- Logging requirements

- Utility Control

  -- Message Mapping requirements
  -- Edit Utility requirements
  -- Output Utility formatting specifications
  -- Change/Display Utility file descriptions

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

As noted above, specifications of message processing control functions are tabled, for example, 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.

System control functions are table-driven: tables produce specifications indicating 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, some application program services operate according to user-specified table entries.

Major functions in Intercomm are controlled by the following tables:

- Verb Table

  A table listing all valid transaction identifiers or verbs and relating them to the subsystem required for message processing. There is one entry per transaction or message type.

- Network Configuration Table

  A table describing the terminal network access method(s) and hardware operating characteristics and relating individual terminals to five-character station identifications.

- Station Table and Device Table

  Tables describing terminal device-dependent characteristics to the Output Utility and the Message Mapping Utilities.

- System Parameter Area (SPA)

  A table describing system-wide operating characteristics. This table may be extended to include installation-defined table parameters, accessible to all subsystems.

- 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 at Intercomm startup.

- Subsystem Control Table (SCT)

  A table listing the characteristics (reentrancy, language, entry point, etc.), queue specifications (core and/or disk queues), scheduling specifications (resident or loadable, concurrent message processing limits, etc.) for each subsystem.

The Intercomm system components are individual routines coded in a generalized form where applicable. Each system component receives specifications for how it should function via table entries coded using Intercomm macros. Table entries may describe a hardware configuration, that is, the communications network, or software specification, such as MMU control functions.

By adjusting variable table entries, the user effectively tailors Intercomm routines to his installation without modifying any program logic. It can be seen that Intercomm is a flexible system via the implementation of user-selected options specified by tables or JCL.

Chapter 3

APPLICATION PROGRAMS AND SERVICE ROUTINES

3.1 APPLICATION PROGRAMS

In the Intercomm environment, an application program is a subsystem which executes under the control of the Subsystem Controller. A subsystem is executed after the arrival of a message, and therefore is message-driven. All scheduling, activating, and loading of the application programs is done by the Subsystem Controller by referencing the appropriate scheduling criteria in the SCT. The subsystem, in the capacity of a subroutine, is called by the Subsystem Controller which passes the address of the particular input message which the subsystem is to process.

Selection of an incoming message for processing is based on the priority assigned to the corresponding program and the ordinal position of the program in the Subsystem Controller message processing algorithm. Queued messages are processed on a first-in/first-out (FIFO) basis. A retrieved message is directed by the Subsystem Controller to the appropriate subsystem.

The subsystem performs the following functions (* indicates use of Intercomm-supplied service routines):

- Edit the Input Message

  -- Format conversion*
  -- Content checking

- Perform Logic of the Application

  -- Message text analysis
  -- Data base access*
  -- File access*

- Optionally switch the message, or a modified form of it, to another application program for further processing or for transmission to a terminal.*

- Format the output message(s)*

- Queue the output message(s)*

Application programs invoke Intercomm service routines via CALL statements. A preprocessor is not required to produce compiled programs executable in the Intercomm environment. No macros are used in high-level language programs.

Application programs can be written in any of the following languages:

- Assembler Language
- ANS COBOL, OS/VS COBOL
- PL/1-Optimized
- FORTRAN

COBOL programs compiled by the CAPEX Optimizer are also accepted. Details on application programming interfaces are provided in the Programmer's Guide for each language.

3.2 MESSAGE PROCESSING LOGIC

The character string keyed in at the terminal may be converted to an appropriate format for the application program by the Utilities. Thus, the program logic can always operate on an edited input message of exact format, irrespective of the originating terminal type. Content checking (such as value range editing) is performed by the application logic. The logic of the application program proceeds by analyzing the message text, determining file records to access from a particular data base, performing updates, or perhaps verifying information sent in and requesting correction of information. The actual file access is performed by calling a File Handler or DBMS interface service routine. The application program may create one or more response messages to be sent to the originating terminal and/or other terminals.

Intercomm provides several service routines which process terminal-dependent input messages to convert them to terminal-independent form for application processing. This processing includes removal of terminal-dependent control characters and conversion of data fields to the desired form. Similarly, for output messages, service routines provide transformation from terminal-independent results of application subsystem processing to terminal-dependent messages for transmission. This includes insertion of terminal-dependent control characters, conversion of data fields to character format, if required, and inclusion of header and title information, if specified. Each of these routines function via user-specified descriptions (tables) of input and output message formats. These service routines are:

- Message Mapping Utilities

  This is a set of service routines which perform device-dependent transformations. It handles both input and output messages.

- Edit Utility

  This is a service routine to process input messages, performing device-dependent transformations (and editing).

- Output Utility

  This is a service routine to process output messages and pass them to the Front End, performing device-dependent transformations.

Subsystem logic for input message text analysis and output message text creation varies depending upon whether or not Message Mapping or the Edit and Output Utilities are used. Figures 15 and 16 illustrate subsystem processing logic for both cases.

The application program is not responsible for actual transmission of a message to a terminal; messages are queued (put in a waiting list) by the Intercomm-supplied terminal queuing routines for subsequent processing. Intercomm line control functions perform the appropriate program logic to accomplish transmission for the particular type of terminal. (See Chapter 4, "On-Line Utilities.")

3.3 NONREENTRANT AND REENTRANT SUBSYSTEMS

A nonreentrant subsystem is a single threaded subsystem which can process only one message at a time. A reentrant subsystem is capable of processing several messages concurrently (multithreading) in order to provide a high volume capability. Additional messages can be passed to a reentrant subsystem without waiting for the processing of predecessor messages to complete. The subsystem can be operative, for example, during the time a previous message is being serviced by a File Handler routine. The user prescribes in the Subsystem Control Table the maximum number of messages to be concurrently processed. The actual program logic remains the same regardless of the number of messages processed concurrently.

However, certain coding conventions are requisite for reentrancy:

- Only application programs coded in Assembler Language, PL/1 or COBOL are eligible for reentrant scheduling.

- An application program must contain serially reusable logic between I/O operations, that is, the program must not modify itself.

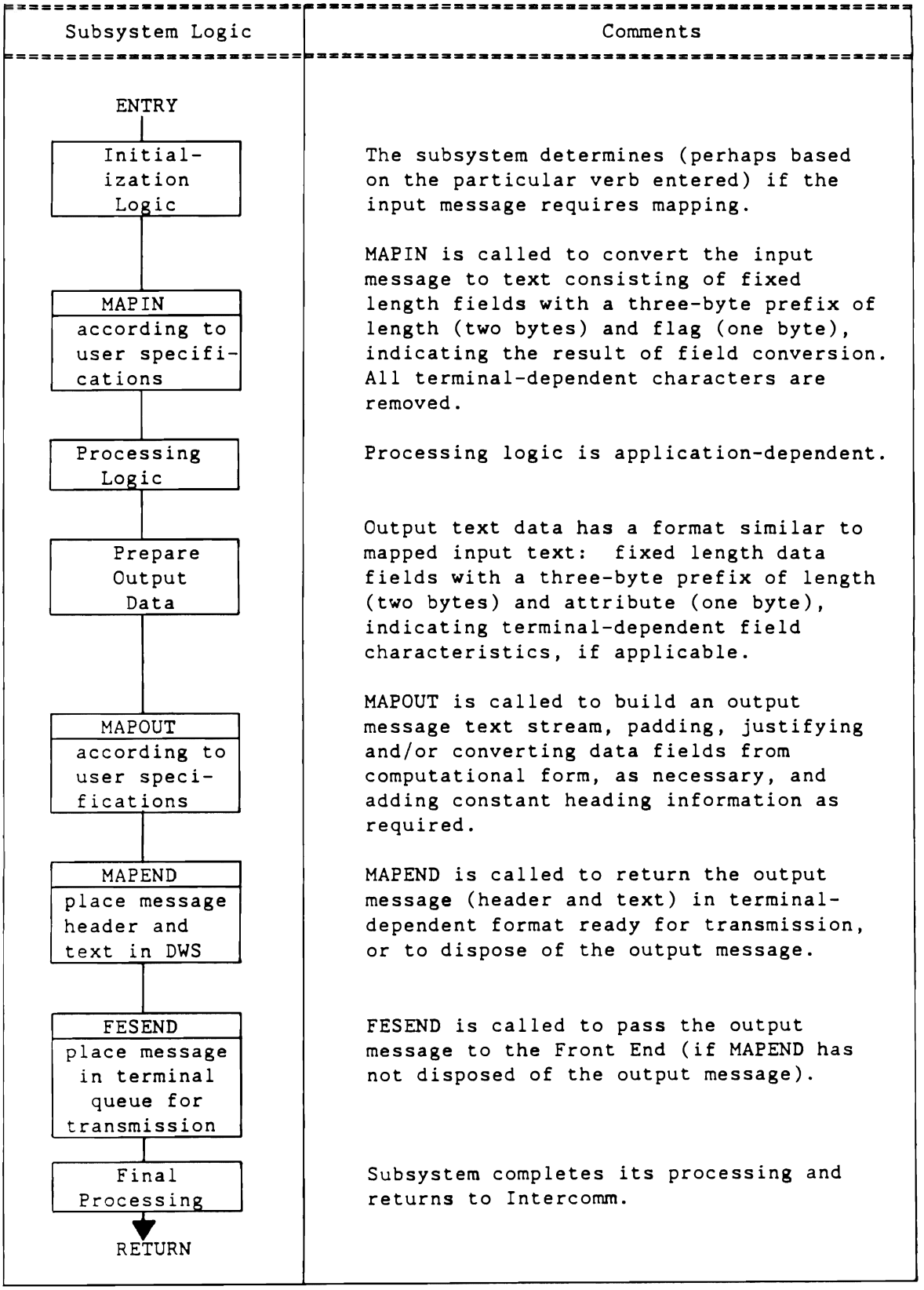


Figure 15. Subsystem Logic Using Message Mapping Utilities

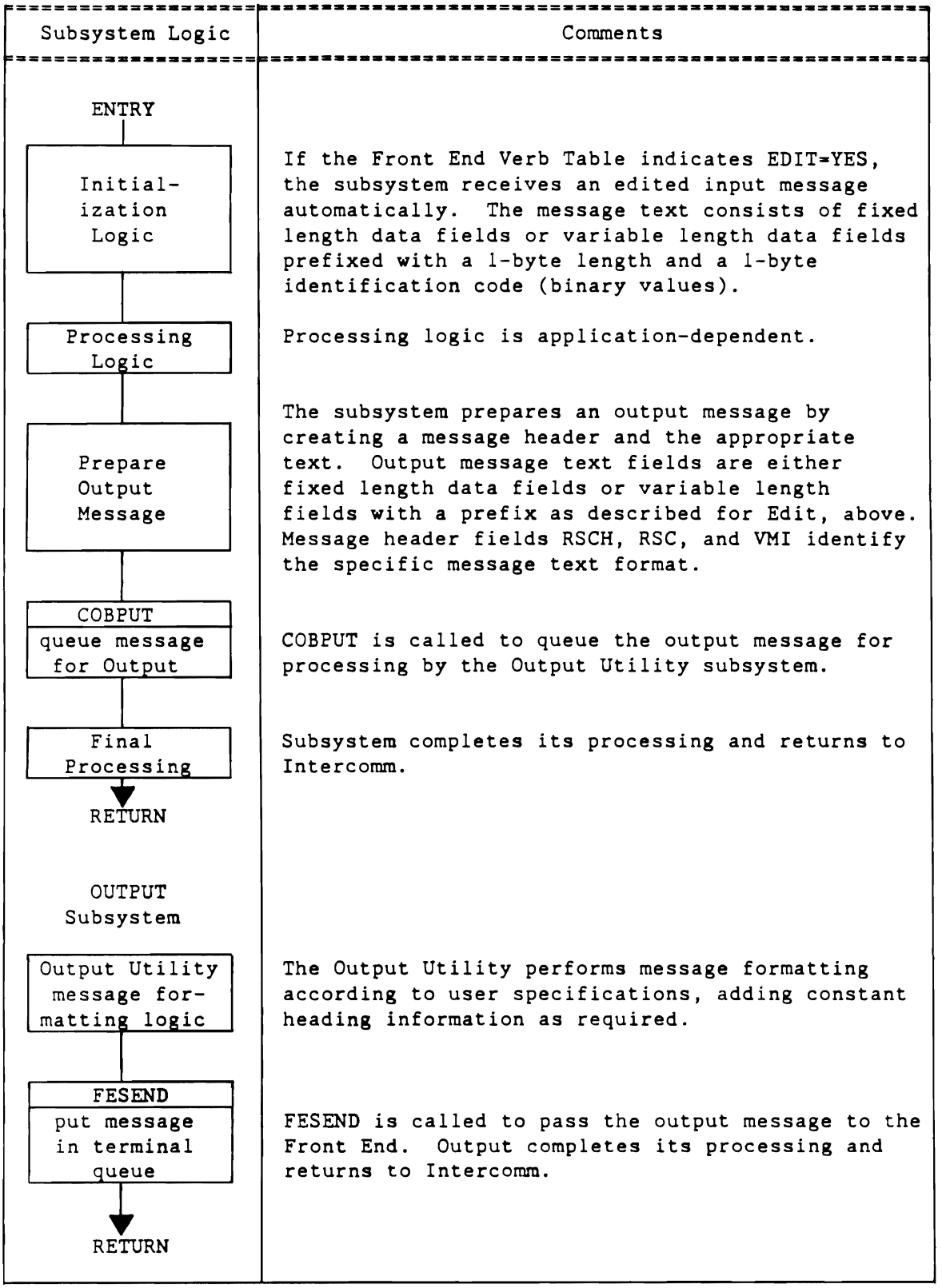


Figure 16. Subsystem Logic Using Edit and Output Utilities

- Working storage modified during subsystem execution must be unique to each message. Work space is required for application program I/O areas, switches and output message areas. This work space is main storage allocated dynamically by Intercomm from its pool area on an as-required basis, that is, dynamic working storage. (The pool is dynamic in that the composition varies; that is, areas are assigned, released, or made available for reuse as soon as a program indicates the area is no longer needed.)

- A reentrant Assembler Language subsystem utilizes Intercomm macros to get and free dynamic working storage; PL/1 utilizes its own automatic storage facility; and for a reentrant COBOL subsystem, Intercomm obtains dynamic working storage and passes the address to the subsystem via the LINKAGE SECTION. The reentrant application program performs its functions on a separate area of dynamic working storage, as opposed to the normal working storage areas coded within an application program.

3.4 SUBSYSTEM ENTRY PARAMETERS

The following entry parameters are passed to a subsystem:

- Input Message--COBOL, PL/1, FORTRAN, Assembler Language

  The logic of a subsystem is usually designed to process one input message. When called by the Subsystem Controller, the address of the input message to process is passed in this parameter. The address of the edited input message (an option) is passed to the high-level language subsystem, and the unedited message address is passed to the Assembler Language subsystems.

- System Parameter Area--COBOL, PL/1, FORTRAN, Assembler Language

  The System Parameter Area is a system-wide table containing control values for Intercomm, that is, those which may not be modified in the course of program execution. The System Parameters Area can be extended to include user fields or an area of main storage common to all application programs.

- Subsystem Control Table--COBOL, PL/1, FORTRAN, Assembler Language

  The Subsystem Control Table has some adjustable values, but, more significantly, it describes the application programs with one entry per subsystem according to:

  -- Characteristics (reentrancy, language, entry point)

-- Queue specifications (core and/or disk queues)

-- Scheduling (resident or loadable; concurrent message processing limits)

- Return Code--COBOL, PL/1, FORTRAN

  The fourth parameter, the return code, is defined in control blocks of the Subsystem Controller and indicates if message processing successfully completes. For Assembler Language programs, it is returned in register 15.

- Dynamic Working Storage--Reentrant COBOL Subsystem

  The remaining entry parameter, that of dynamic working storage, is required only in reentrant COBOL subsystems. Dynamic working storage comes from the Intercomm main storage pool and is acquired or assigned dynamically for use by a particular program on an as-required basis.

3.5 INPUT MESSAGE FORMATS

The message received by the application program can be of a nonedited format, comprised of a 42-byte message header and the message text character string as entered by the operator at the terminal. Alternatively, Message Mapping Utilities or the Edit Utility may be used to provide an input message format of either fixed-length fields in a predefined format or variable-length fields in a variable sequence, both prefixed by a message header. High-level language application programs utilize the fixed-field format. The variable format of an edited message is most useful in Assembler Language programming, storage being a consideration. In this format, the final message length is dependent upon the amount of information entered by the operator.

3.6 OUTPUT MESSAGE FORMATS

A message for transmission to a terminal may be constructed in two formats by the application program. A preformatted message may be constructed consisting of the message header, and the message character string ready for transmission to the terminal, including all positioning characters (blanks, tabs, etc.), headings, and control characters. Alternatively, formatting for display may be performed by either Message Mapping Utilities or the Output Utility. Every message is prefixed with the message header. Fields in the message header (created by copying the input message header and adjusting certain fields) designate the output terminal and other specifications.

Messages created by an application subsystem are passed directly to the Front End for transmission in the case where preformatted messages (or fully formatted results of Message Mapping Utilities processing) are utilized. This is accomplished by a service routine CALL. When the Output Utility is utilized, messages from the user subsystem are routed to the Output Utility by a service routine CALL for intersubsystem message switching.

3.7 INTERSUBSYSTEM MESSAGE SWITCHING

A flexible environment is provided for the passing of messages or data between user application programs. This facility is available to all subsystems, and is accomplished via a direct interface to the Intercomm queuing routine through a simple two-parameter CALL.

Message switching may be required as in the instance of an analyzing subsystem which, upon receiving a transaction from a terminal, analyzes and queues a message for another program. A different subsystem performs the actual processing of that message. Utilization of message switching can take advantage of the Intercomm priority structure, that is, once the original input message is analyzed, it can be sent to another lower priority subsystem for the processing of that particular message.

3.8 SCREEN GENERATION

Screen generation subsystems are provided to enable users to display predefined screen formats at a CRT terminal. This facility requires no user programming although the screen obviously must be defined to the system via OFT (Output Utility) or Map (MMU). This feature is useful where input is entered at the terminal by filling in the blanks in a screen template. The terminal operator merely indicates which format is desired and the system automatically sends that particular screen to the terminal for data entry.

3.9 MESSAGE SWITCHING BETWEEN TERMINALS

Message switching and broadcasting between terminals is provided by an Intercomm-supplied subsystem. If the receiving terminal is unable to receive the output message, Intercomm reroutes the message to an alternate device, if specified, or queues the message until the receiving terminal is available, at which time Intercomm then transmits the queued message(s) to the terminal. The messages that are on queues waiting for transmission are preserved across system failures by the Message Restart facility.

3.10 SERVICE ROUTINES FOR APPLICATION PROGRAMS

In the Intercomm system, many program functions which are normally application-oriented are accomplished by utilizing service routines. Therefore, the application program logic need not be responsible for the following:

- Data file access
- Message queuing
- Message logging
- Generation of Front End control messages

In addition, Special Features applicable to application program use are generally invoked by calls to service routines. This includes features for CRT page browsing, DBMS interface, and temporary storage of data (Store/Fetch, Dynamic Data Queuing).

Any Intercomm service routine can be called by any application program. (Some of the corresponding functions are alluded to in Chapter 2, "Intercomm System Components.") Intercomm service routines include the following for:

- Data File Access

  A File Handler service routine is called to read and write records accessed directly or sequentially. All OS/VS data set organizations and processing techniques, except BPAM, are available to programs in the Intercomm system. The application program does not contain any data management instructions itself; the required input/output processing for all Intercomm subsystems is performed by the File Handler.

- Message Queuing

  Queuing routines are called to accomplish queuing of messages in the Intercomm system. The efficient core and disk queuing capability facilitates message processing and contributes to high volume throughput.

  The intersubsystem Message Queuing facility is a general purpose system component used for message segment queuing, report and output segment queuing, external storage buffering, and general application program use. It is used by Intercomm to perform queuing of internal messages and output messages generated by applications.

  When Intercomm's BTAM or VTAM Front Ends are used, the Message Queuing facility is the means by which message traffic to and from the communications network and the message processing subsystems is monitored.

- Message Logging

  A call to the message logging service routine creates a special entry on the system log (INTERLOG) for the application program, or for a particular terminal, or both. Message status information or exceptional conditions may also be logged. All incoming and outgoing messages and message processing phases (when message processing starts and completes) are recorded automatically by Intercomm. Not only highly useful for accounting purposes, the log also renders statistics for system preformance analysis. (See Section 6.5 Intercomm Statistics.) The Intercomm system recovery function is implemented through use of the log to rebuild message queues, recover messages not completely processed at the point of system failure, and reset files. Logging and/or message restart can optionally be eliminated on a subsystem or terminal basis.

- Generation of Front End Control Messages

  A service routine is invoked to create a special message which is subsequently routed to the Front End to accomplish one of the following functions:

  -- Request feedback or notification that a message has completed transmission.

  -- Override normal Front End logic and cause immediate output to a CRT rather than wait for operator input.

  -- Notify the Front End of the existence of a Dynamic Data Queue (DDQ) of messages waiting for transmission.

3.11 CONVERSATIONAL SUBSYSTEMS

Conversational subsystems are defined as one or more subsystems designed to process more than one input message to complete a transaction. They effectively carry on a dialogue with the terminal operator, receiving an input message, retaining it and/or associated results of processing, issuing a response (perhaps a prompt for additional information), receiving another input message, retaining it, etc., until the transaction is complete. At the end of the conversation, appropriate files may be updated.

Conversational transactions involve the sending and receiving of more than one message in a terminal session. Each input message may be processed by related subsystems or by the same subsystem. A two-part conversational transaction is illustrated in Figure 17.

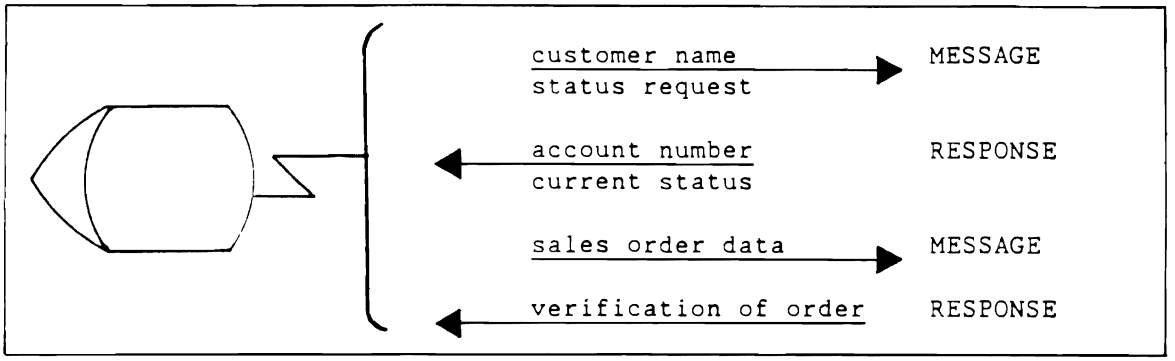


Figure 17. Typical Conversational Transactions

Assume a conversation in which three input messages and three responses are necessary to complete the transaction. A terminal, a subsystem and a storage medium on which to save the intermediate input message data is required. When the final input message is received and processed, appropriate files are updated and a final response is issued. Figure 18 illustrates the steps in this conversational process example.

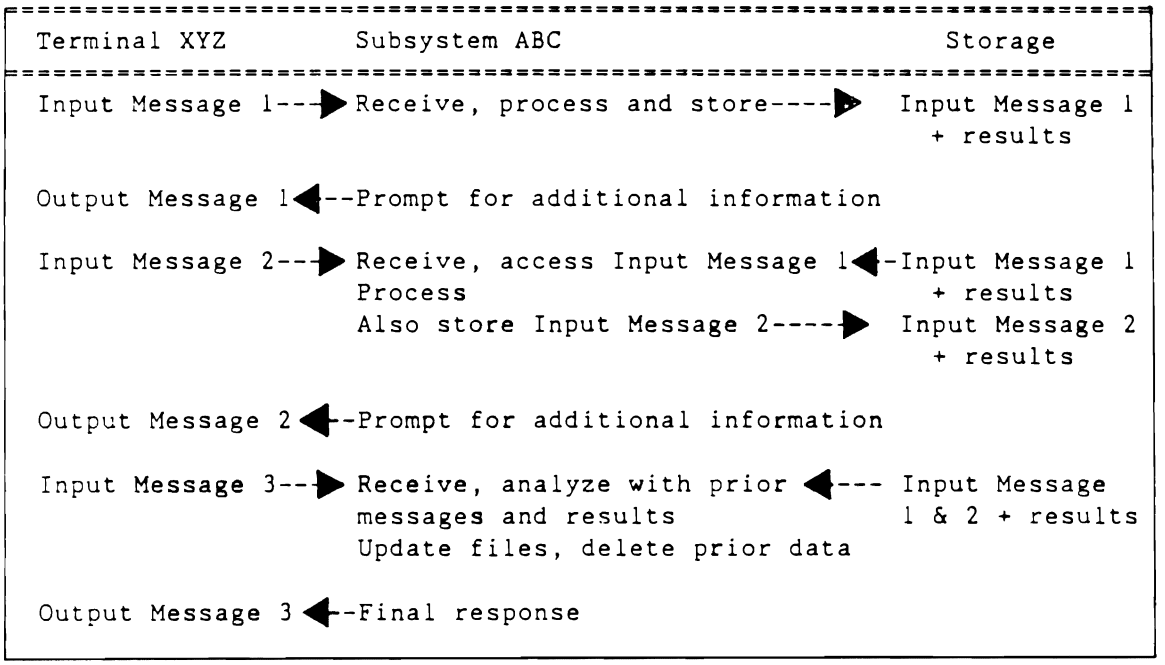


Figure 18. Retention of Input Messages During a Conversation

Conversational subsystems require a storage medium for saving input data and intermediate results until all processing of the transaction has been completed. Intercomm provides various techniques for saving this data. These include the following:

- User SPA (User Extension to the System Parameter Area)

  This is useful for storing system-wide information, that is, data that is accessed and used by most programs which run under Intercomm. An example of this is operator error statistics.

- Store/Fetch Facility

  This can be used to store data with a unique key (up to 48 characters) for short time periods when response time is critical. The data is retained in main storage (if available) with roll-out to disk automatically invoked for the least recently used data if main storage is exhausted.

- Dynamic Data Queuing Facility

  The Dynamic Data Queuing Facility provides a pool of disk space for saving large amounts of data which would otherwise consume too much main storage. It provides an excellent means of batching transactions for later update and spooling of messages for a printer, as well as for saving intermediate results in a conversation.

- CONVERSE Service Routine

  This provides the easiest (but most costly) method of handling conversational processing. It is the easiest method since all processing is done in the normal manner, except that following the queuing of the message for transmission, the program calls CONVERSE. This results in the program remaining in main storage (unless it is part of an overlay group) until a response is received from the terminal. When this occurs, the program is reactivated at the next instruction following CALL CONVERSE. It is the most costly retention method because storage is unavailable for use by other subsystems which would have been able to process while the conversational subsystem was waiting for the next message. A CONVERSE time-out may be requested to prevent an endless wait.

The choice of technique used for storage retention by conversational subsystems is a design decision which the user makes based on response time requirements, storage limitations, amount of data, and so forth. The important factor to consider is that Intercomm provides the necessary alternatives from which the user can choose the best solution to the unique requirements of each application.

Chapter 4

ON-LINE UTILITIES

4.1 PROGRAMMER PRODUCTIVITY

The Intercomm On-Line Utilities include Message Mapping, Edit, Output and Change/Display. Flexibility is provided by these utilities since they are controlled by tables. The tables, which are created by the application programmer but which are independent of the application program, define the utility program logic for each type of transaction. Since the tables are independent of the application program, they can be changed to alter the message format as it appears on the terminal (or the terminal type may even be changed) without alteration of the application program. An Intercomm subsystem need not be concerned with terminal specifics such as control characters, attribute bytes and addressing. It is relieved by the Utilities from the responsiblity of analyzing formats of incoming messages and for formatting output messages.

The Change/Display Utility is a table-driven application program provided with Intercomm. It can be used to satisfy simple file (single record) update and/or inquiry requirements without the need to write a specific application program.

The net effect of the Utilities is to enhance programmer productivity both in initial system implementation and in subsequent maintenance and modification phases.

4.2 MESSAGE MAPPING UTILITIES

The Message Mapping Utilities (MMU) provide an interface between the application subsystem and terminal-dependent message processing logic for both input and output messages. MMU is invoked by calls to service routines which perform mapping functions based upon user-specified tables. Mapping includes conversion of character data to computational or binary format and vice versa, and stripping/insertion of dollar signs ($) and decimal points. MMU input mapping produces fixed-length data fields prefixed by a two-byte length and a one-byte flag (indicating errors or omissions) field, unless the data fields are defined in a named segment (group of fields). In this case, the three-byte prefix occurs for the entire segment, rather than the individual fields. Fields are padded or truncated and justified, as necessary, according to user definitions.

MMU output mapping operates upon data in the same format, but the flag byte becomes the field (or segment) attribute character. The mapped input text area and the unmapped output text area are defined in the application program's dynamic work space. The application program references data fields and associated prefix fields by symbolic name. For example, a customer name of 25 characters would appear as an MMU symbolic COBOL definition as follows:

```
04  CUSTOMERL   PIC XX.      (length)
04  CUSTOMERT   PIC X.       (flag/attribute)
04  CUSTOMER    PIC X(25).   (data)
```

Message Mapping is offered as an alternative to the Edit and Output Utilities. MMU is particularly effective for dynamic manipulation of CRT screen data. For example, MMU is highly effective for a design where:

- A screen format with a fill-in-the-blanks template is displayed.
- The operator enters data into the unprotected fields and submits the message.
- The application subsystem highlights erroneous fields on the screen and adds error messages to the existing display on the screen.

MMU is one set of service routines for both input and output message formatting. Where both input and output formats are similar, then a single format descriptor (called a "map") can be used. With MMU, subsystems receive messages unchanged in format from original receipt from the terminal. Mapping is accomplished by calling service routines. A subsystem passes an input message to MAPIN for reformatting into device-independent form. Output messages are passed to MAPOUT for conversion from device-independent form to device-specific form, including all necessary control characters and sequences. When the subsystem is ready to transmit the mapped output message(s), a MAPEND routine is called which can be requested to automatically queue the output via the TP interface routine (FESEND). Control of MAPIN and MAPOUT for specific devices is provided by Device-Dependent Modules and the Device Descriptor Table.

4.2.1 Terminal Input Format Options

Message Mapping Utilities accept four different types of input message formats:

- Keyword Format
- Positional Format
- Formatted Screen Format
- Fixed Format

Each input option provides different advantages to consider in determining which input format to use for each message type entering Intercomm. The design approach should consider operator convenience, terminal characteristics and transmission requirements.

In this section, the conventions used for input message format notation are:

#	The field separator character--a delimiter for individual data fields entered in positional format
=	The keyword field start character--a delimiter signalling the end of a field-identifying keyword
%	The keyword field end character--a delimiter signalling the end of a keyword-identified data field (usually the same as the field separator)
&	The end of line--new line (NL), carriage return (CR) or carriage return/line feed (CR/LF) character(s) of the terminal
@	The end of message character sequence of the terminal--EOT, EOB, ETX, etc.

The delimiters are defined as system-wide values in the MMU Vector Table. They may also be defined in the Device Description Modules to specify override values for a particular device. Any map definition may also specify override values for a particular input message or portion of an input message.

The end of line characters are interpreted as field delimiters by MMU. End of message naturally signals end of input, and MMU processing of the message completes. The end of line and end of message character(s) for each device are an internal definition in the MMU Device-Dependent Modules.

When data is to be supplied in the Keyword Format, the message is entered in the following manner:

```
TRNS&                          (verb or transaction identifier)
CUST=JOHN R. WILLIAMS JR.&     (customer name)
ADDR=727 E. 43RD ST.&          (customer address)
C/S=WEST HEMPSTEAD L.I.&       (customer address)
ACCT=7432710&                  (customer account number)
DEBIT=$27.42&                  (debit amount)
CREDIT=$1.27&                  (credit amount)
END@
```

The four-character verb is the first field of each message. Each data element in the message is identified by a unique one-to-eight character field identification. The field identification is immediately followed by the data for that field. (Any number of separating blanks can be inserted between the keyword end character and the data; the blanks become part of the data field.) The keyword must be unique within any one transaction type; it may be reused in other transaction types. For example, the keyword CUST may be entered in all transaction types that require a customer name to be entered.

For data elements that can be entered more than once in a particular input message, the terminal operator simply enters the given keyword more than once in the message; each use of the keyword is followed by the appropriate data. Thus, in the example above, if it were possible to have three debit amounts, the data entered would be as follows:

```
DEBIT=$27.42&   (debit amount 1)
DEBIT=$ 7.93&   (debit amount 2)
DEBIT=$ 8.47&   (debit amount 3)
```

The terminal operator must be taught the proper keywords to use on input. Any field not applicable on the current entry can be omitted by not entering that keyword (and data) in the message. In addition, fields may be entered in any order.

When entering data in the Positional Format, only data fields are entered; no field identifications are used by he teminal operator. The data fields are separated by the field separator character (#). The example given in the section describing the Keyword Format would be entered in the following manner using positional format:

```
TRANS#JOHN R. WILLIAMS JR.#727 E. 43RD ST.#WEST HEMPSTEAD L.I.&
7432710#$27.42#$1.27
```

As the example illustrates, the transaction identification (verb) is also required in this format. Every possible field for the particular message type must either be supplied by the terminal operator or noted as omitted by the insertion of an extra separator character to indicate the absence of the field.

Some CRTs have the facility to display template screens for the operator's convenience. For example:

```
.... TRANS
.......................... CUSTOMER NAME
.............................. ADDRESS
.............................. CITY/STATE
........................... ACCOUNT NUMBER
..... ..... ..... ..... DEBITS
..... ..... ..... ..... CREDITS
```

The periods indicate those screen positions where an operator may enter data. In the case of the IBM 3270, only the data entered is transmitted as an input message, along with control characters indicating the screen position of the data fields and other terminal-dependent control characters.

Formatted Screen is the term used to identify this input option; the data fields are specified to MMU using row and column notation indicating position relative to the start of a particular screen area.

Some applications may require processing of messages in a format similar to batch mode fixed-length records. In this case, every input message contains fixed-length data fields in a fixed position within each message type. This situation might occur with CPU-to-CPU transmission of data fields.

For some applications, format options may be combined in one input message. A typical use of this facility might be to allow the operator to enter a mix of keyword and positional fields:

```
TRNS#JOHN R. WILLIAMS JR.&
727 E 43RD ST#WEST HEMPSTEAD L.I.&
ACCT=7432710&
CREDIT=$1.27%$48.26#$9.95
```

This approach combines efficiency of Positional Format and convenience of Keyword Format.

4.2.2 Mapping an Output Display

Figure 19 illustrates a typical screen image which can be produced using MMU. The MMU screen definition (map) defines field position and constant (title) data. One screen may be defined by one or more maps. For example, the BILL TO and SHIP TO sections might each consist of separately defined maps. The ITEM section (list of items) might be defined as one map with one segment (group of related fields) occurring three times (repeating down the screen).

```
        ON-LINE SHIPPING                          04/07/85

B  XYZ CO., INC.                    S  XYZ CO., INC.
I  1 ANYWHERE ST.                   H  1 ANYWHERE ST.
L  SOMEWHERE U.S.A. 11111           I  SOMEWHERE U.S.A. 11111
L  212-121-1111                     P  212-121-1111

BILL DATE:  04/04/85                SHIP DATE:  04/04/85

ITEM   DESCRIPTION          QTY  AMT

X102Y  WIDGET-1/2 INCH       10   20.00
Z423A  SPROCKET-.05 INCH     15    3.00
A158X  GADGET-1 INCH          8   12.00

       TOTAL                      35.00
```

Figure 19. Sample Output for Mapping

The output mapping process, invoked by subsystem calls to MAPOUT, would take constant information such as the titles ON-LINE SHIPPING, BILL DATE, SHIP DATE, etc., from the map definition and combine it with specific data for the response supplied by the application subsystem.

4.2.3 Mapping A Template Screen

As illustrated by Figure 20, one display screen and its corresponding set of maps can be used for both input and output mapping functions. The output mapping function displays the template for the operator's convenience in entering data. The input mapping function transforms and edits the data entered by the operator. This assumes use of a terminal with formatting capabilities, such as the IBM 3270. Output mapping could also be used to produce messages to the terminal operator for error correction procedures or to signal that a new input message may be entered.

The entire screen in this example is a map group (group of related maps). Each individual line may be defined as a map containing one or more fields, or, the entire screen may be defined as one map specifying all fields on the screen. The decision should be based on application programming convenience when invoking MMU service routines. In this instance, defining the map group with one map simplifies the application program logic.

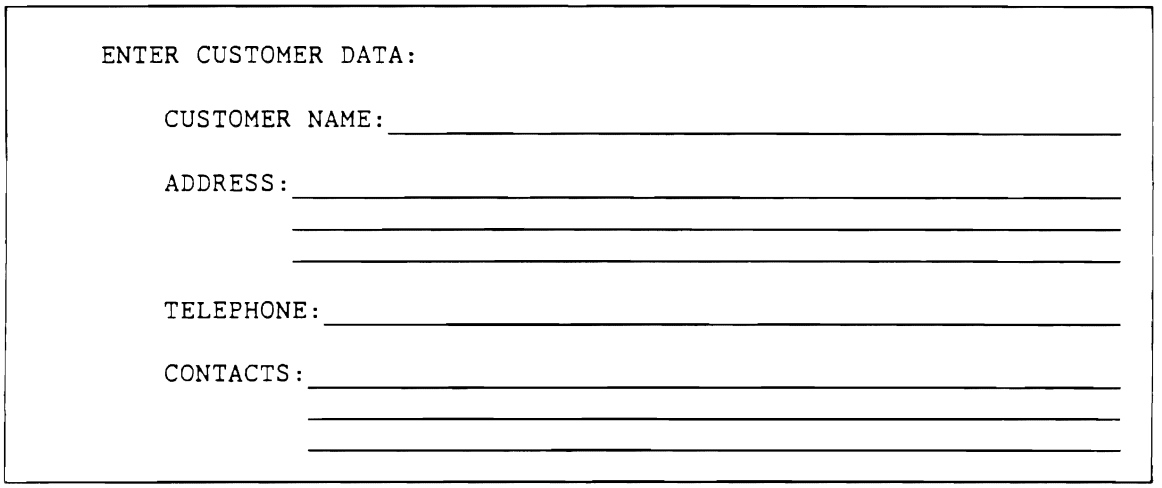


Figure 20. Sample Template Screen for Mapping

4.2.4 Message Processing Logic Using MMU

A summary of message processing logic using MMU is shown in Figure 21. For a complete description of MMU and its use by application subsystems, refer to the Intercomm Message Mapping Utilities.

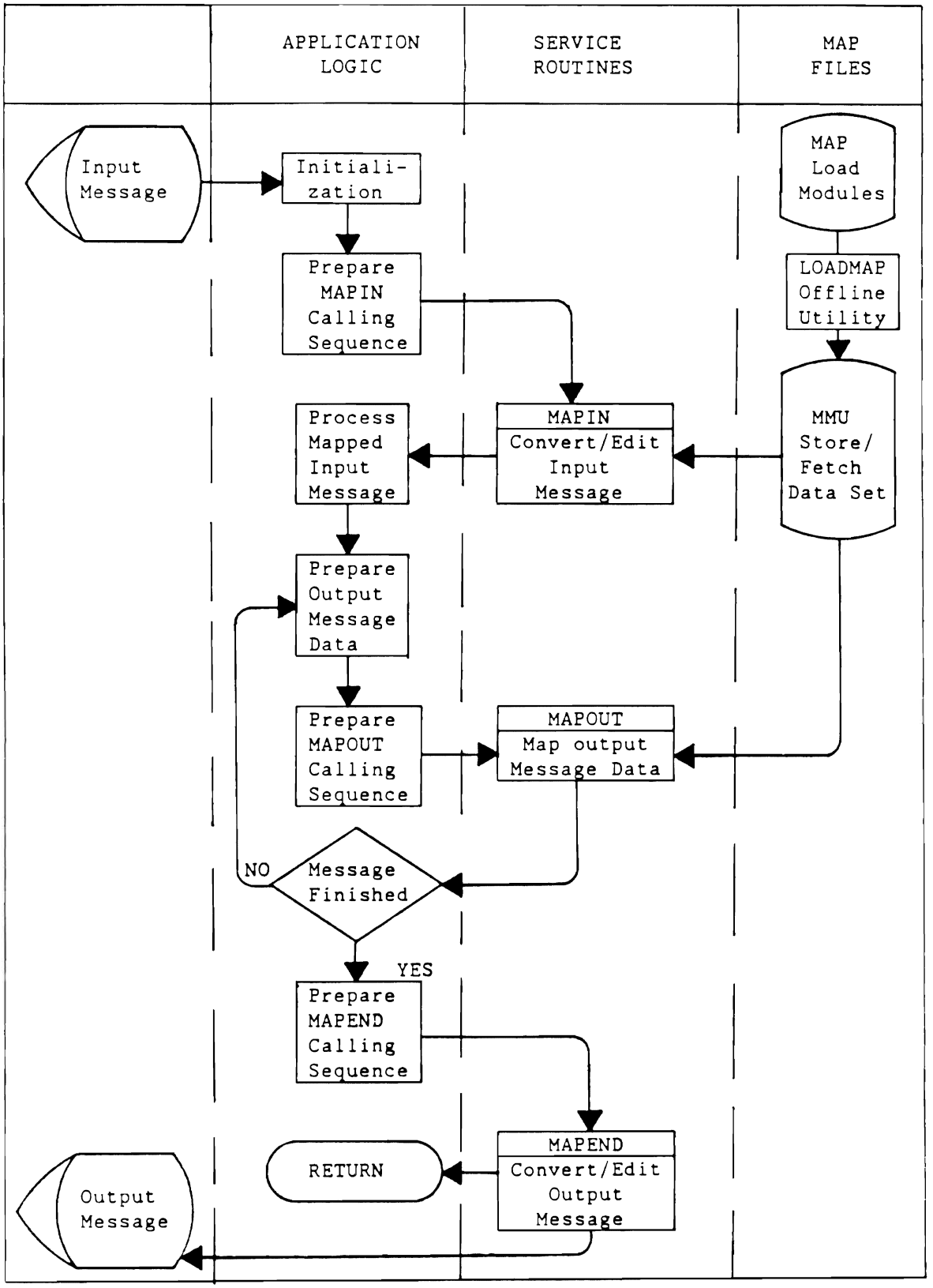


Figure 21. Message Processing Using MMU

4.3 EDIT UTILITY

The Edit Utility is a generalized utility program used to prepare input messages for processing by application programs. The utility is controlled by an Edit Control Table (ECT), which defines the characteristics of each input message, and the type of editing to be performed. One table entry exists for each message type to be edited.

The Edit Utility is called if indicated by control table specifications to edit the message according to edit specifications defined in the ECT. It is possible to have the input message edited by Message Collection prior to being queued, edited immediately prior to activating the message processing program (high-level language), or edited via a call from an Assembler Language program. (See Figure 22.) The Edit Utility performs the following:

- Eliminates all teleprocessing control characters.
- Pads/truncates/converts/packs/strips each field as needed; fields can be padded or truncated, converted to binary or packed decimal or stripped of dollar signs or decimal points
- Identifies required and omitted fields; if the operator omits fields specified as required, the message is rejected.
- Provides error messages; notifies operator of exact type of error.

The terminal operator has three format options available; input can be in a keyword, positional, or positional within keyword format, as described previously.

Specifications to the Edit Utility indicate how each field is to be formatted. The Edit Utility will place fields in the message into a predefined fixed format or a variable format. A fixed format consists of the standard message header, followed by fixed-length fields in predefined positions, a format most commonly used for high-level language application programs, as follows:

	NUMBER	QUANTITY	UNITS	OMITTED
42 BYTE HEADER	XXXXX	XXXXX	XXXXX	PAD

Alternately, the Edit Utility can produce a variable-field format. The standard message header is followed by data fields prefixed by item code (IC) and length (LN) fields, or the data may be prefixed by IC, LN, and an occurrence number (OC) field. The OC field is used only for fields defined as repetitive in the ECT. The field STATUS area indicates whether each parameter was provided and if any were provided in error:

42 BYTE HEADER	IC	LN	DATA	IC	LN	OC	DATA	STATUS

Variable format is useful in Assembler Language programming where main storage is a concern because the final message length depends entirely on how much information is keyed in by the operator.

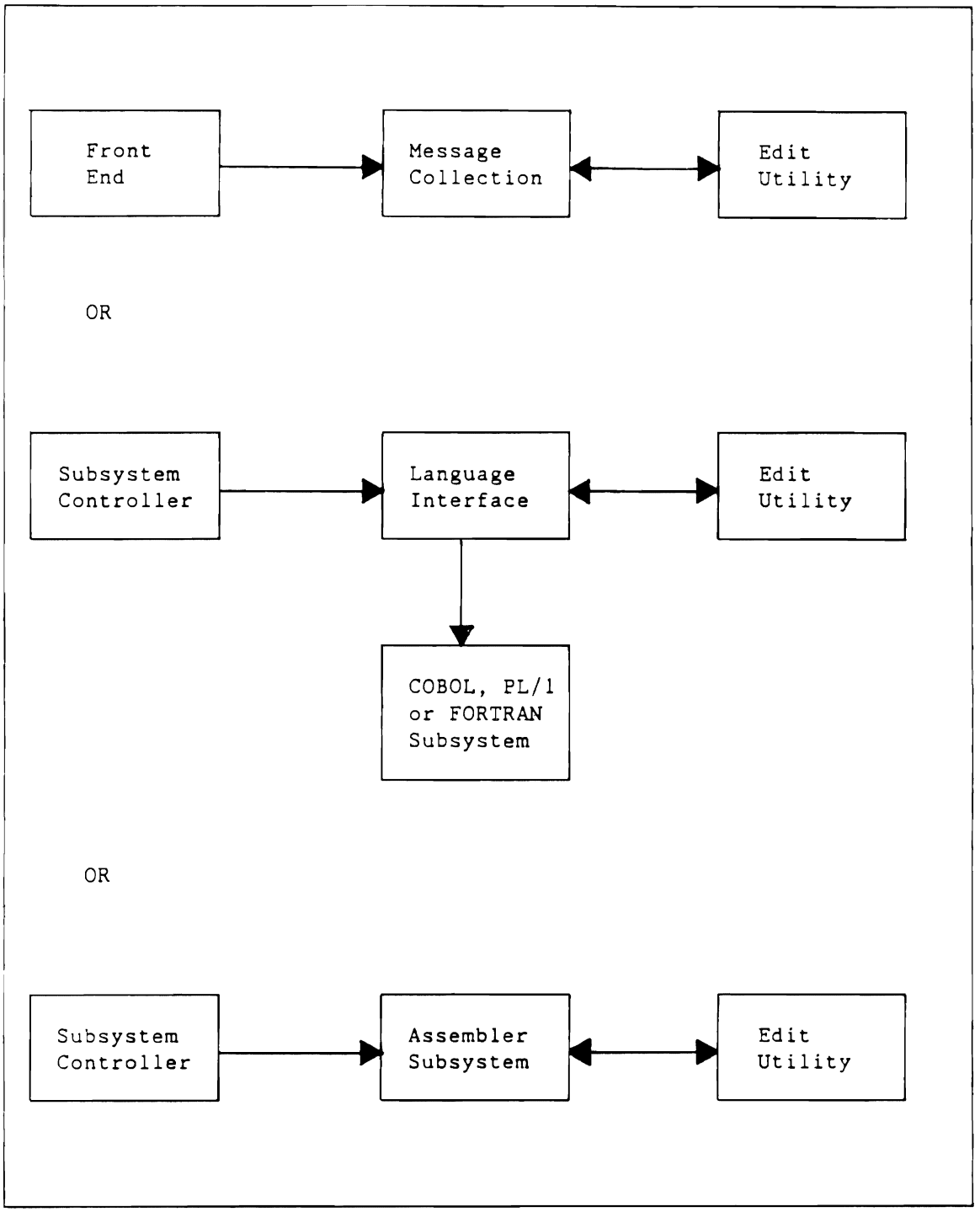


Figure 22. Edit Utility

4.4 OUTPUT UTILITY

The Output Utility operates on messages ready to be formatted and then transmitted to a terminal(s), providing simplified generation and revision of output formats for the telecommunications network.

An application program passes to OUTPUT the data fields necessary for a report. OUTPUT inserts the data fields into a predefined format. The reports (display formats) can subsequently be altered without any program modification. Messages may be displayed on different types of terminals without concern to the subsystem creating the message. This utility is table driven, report and display formats being specified in the Output Format Table. (See Figure 23.)

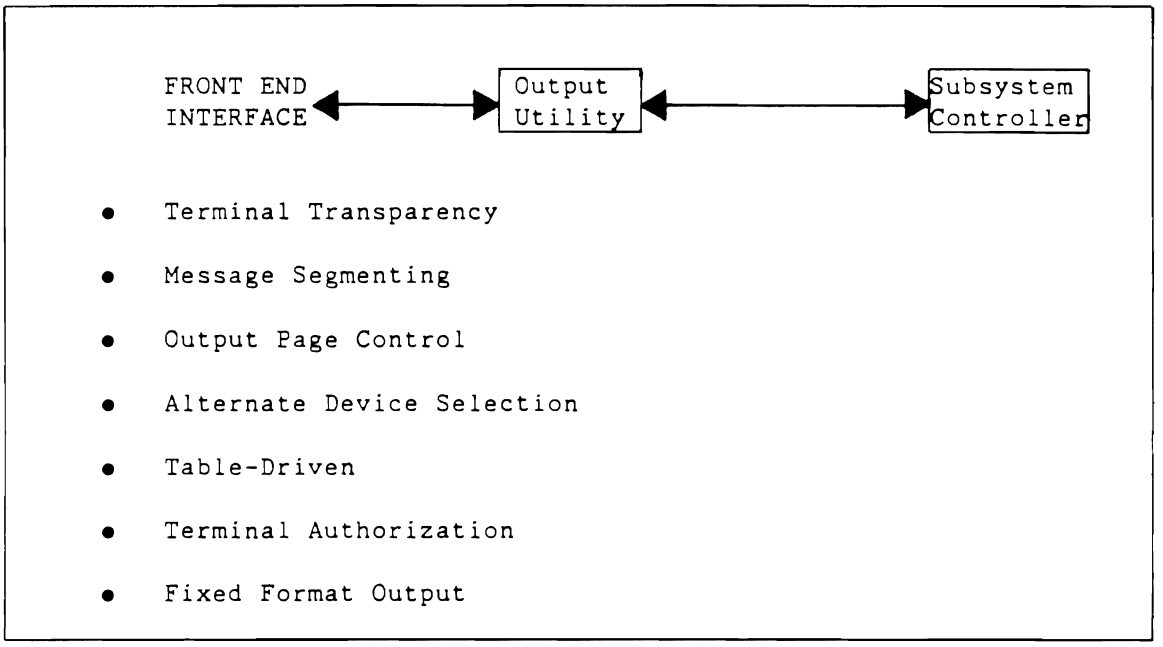


Figure 23. Output Utility

After a subsystem has created an output message it calls MSGCOL to queue the message for the Output Utility. (The fact that the message is to be routed through OUTPUT is noted in a message header field.) The Subsystem Controller then schedules the Output Utility, retrieving the message from the queue and passing it to OUTPUT for processing. The Output Utility then performs the following:

- Selects the terminal. The Output Utility analyzes its internal tables with respect to the particular terminal to receive its messages. If that terminal is not available, the utility refers to a specified alternate terminal.

- Formats output messages as specified in the header, copying data fields from the input to the output message. The Output Utility expands, columnizes, and inserts (sub)headings.
- Inserts teleprocessing control characters
- Passes message to Front End via the TP interface program

The message supplied by the application program for formatting consists of the message header and the fields to be displayed which are constructed within the application program:

42-BYTE HEADER

NAME	NO	HRS	HRS	PAY	NAME	NO.
A CYRUS	174312	40	03	17520C	T FIELDS	197902

HRS	HRS	PAY	NAME	NO.	HRS.	HRS.	PAY
40	00	19000C	C MICHAELS	199031	40	00	27400C

The Output Utility, operating against a table, will position the variable data within the defined format:

EMPLOYEE NAME	EMPLOYEE NUMBER	NORMAL HOURS	OVERTIME HOURS	TOTAL PAY
A CYRUS	174312	40	3	$175.20
T FIELDS	197902	40		$190.00
C MICHAELS	199031	40		$274.00

The Output Utility can also facilitate the preparation of reports containing several different message formats, for example, a title page, detail pages, and a summary page. The Output Utility can receive the message in segments for formatting purposes, one by one, and transmit them in the proper sequence to the terminal until the final segment is transmitted. Segmenting will also be utilized if an output message is too long for terminal transmission.

The Output Format Table (OFT) describes the output message formatting specifications for the Output Utility with one entry, optionally disk-resident, per output format. The Output Format Table is quite simple to generate. The format specifications are supplied by the user via Intercomm macros. Information to be displayed at a terminal can vary by changing the table entries, without change to the application program; hence, the application is independent of the format.

4.5 CHANGE/DISPLAY UTILITY

The Change/Display Utility is a subsystem that provides a simple file inquiry and maintenenace capability without user programming. A remote terminal operator can display an individual record from any BDAM, KSDS VSAM, or ISAM file supported by Intercomm in a fixed-character format at a terminal. Selected fields can then be modified within the record.

Like the Edit and Output Utilities, this module is totally table-driven. There is no program development required to display any fixed-format records. The only requirement for the display of record data (binary, hexadecimal or character) is the creation of a disk-resident Format Description Record (FDR) which describes the record format to the Change/Display Utility.

An entire on-line application can be installed using the Change/Display Utility; thereby coding no application programs, only table entries describing the file characteristics and associated output format specifications.

When an operator enters a Change or Display transaction, the utility is scheduled and the message is retrieved through the Subsystem Controller. The Change/Display Utility retrieves the format description record and the record from the file. For DISPLAY, the message is sent to the Output Utility and processed against a format table entry previously specified. To accomplish a CHANGE transaction, a file update is performed and a message is sent back to the operator, indicating the update was completed successfully.

To display a record from a file, the operator keys in the display verb (DSPL), the keywords for the file name, and the key of the record to be retrieved as shown below:

```
DSPL
FLN      DDNAME01
KEY      7432-14
END
```

Or, the operator can enter the DISPLAY transaction specifying not only file name and key but overriding report number as shown below:

```
DSPL
FLN      DDNAME01
KEY      7432-14
RPT      300
END
```

To modify fields in a file record, the transaction CHNG, file name, record key, and the user-defined field name to be changed is entered (see below). Current field content and the new value is verified and rejected if not valid. The application designer determines if verification of field content is required. Because a CHANGE transaction utilizes a keyword input, the operator is able to enter as many field names and changes as required. Only one record from one file can be accessed with each transaction.

```
CHNG
FLN     DDNAME01
KEY     7432-14
FDN     PRDNO
VRY     701324
DTA     791324
END
```

The Change/Display Utility incorporates the following facilities:

- Ability to preprocess the key supplied. Characters entered can be converted to binary values to access a specific block on a BDAM data set.
- Field editing for display
- Conversion to character, packed decimal, and binary. Characters entered for CHNG are converted to the format appropriate to the field type.
- Padding and truncation of input fields
- Negative character fields retrieved by DISPLAY for transmission to a terminal may be formatted with their minus sign, as required.

DISPLAY can be utilized in lieu of writing an application program when application logic requires retrieval of one record from one BDAM, KSDS, or ISAM fixed-length file. CHANGE can be used when application logic requires update of fields in one record of one file.

The Edit, Output and Change/Display Utilities are described in the Utilities Users Guide.

Chapter 5

MAIN STORAGE ORGANIZATION AND RESOURCE MANAGEMENT

5.1 REGION ORGANIZATION

The Intercomm system executes as a job under the IBM System/370 operating system, in either a single OS/MFT partition, an OS/MVT region, an OS/VS1 partition or OS/VS2 address space, in a multiprogramming, multithreading, on-line environment.

NOTE: MFT systems require the ATTACH and IDENTIFY features to utilize the Intercomm Asynchronous Program Loader.

With any of these operating systems, the user is able to enter and execute any number of concurrent independent jobs in other partitions or regions while Intercomm is executing (provided CPU storage is available and excessive interference with on-line I/O devices is not introduced by "background" jobs).

The operating system treats Intercomm as one execution module, however, it is a module containing many system programs, user-written application subsystems, and utility processing programs.

In providing on-line system functions and centralized control of multiple applications, the Intercomm region (or partition) is composed of the following:

- Resident Intercomm routines and tables
- Resident subsystems
- Nonresident subsystems, dynamically loadable
- Nonresident subsystems, loaded based on planned overlay structure
- Nonresident service routines and table entries
- Dynamic subpool area

Figure 24 shows the components of a single Intercomm region.

The following description refers to the components shown, and describes overlay processing and dynamic program loading; Intercomm's Resource Management facilities are also described, such as the storage cushion feature, optional core use statistics and special storage management facilities:

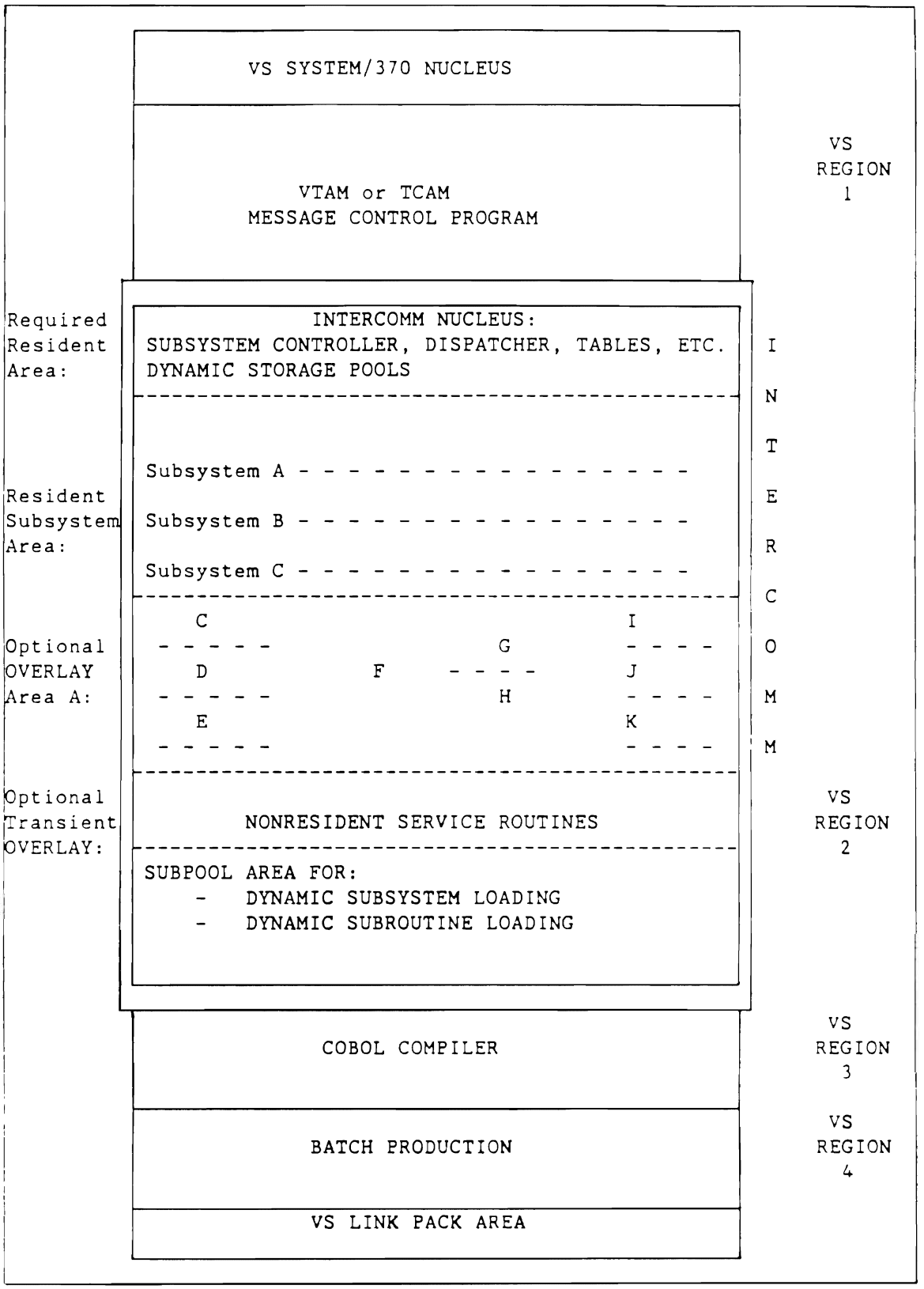


Figure 24. System/370 Main Storage Layout with a Single Intercomm Region

- Resident Intercomm Routines

  These routines are required for Intercomm functions and must be resident in main storage. Residing in this required area is the Intercomm Front End and such routines as the Subsystem Controller, Dispatcher, File Handler, and associated service routines.

- Resident Tables

  Such tables are necessarily resident in that they specify actual control functions of Intercomm. For example, the System Parameter Area (SPA) describes system-wide characteristics. Resident tables share the area with the resident Intercomm routines, and include the user-defined Dynamic Storage Pools area.

- Dynamic Storage Pools

  This is an area of main storage in the Intercomm nucleus where, as storage is required for Intercomm routines or application program workspace, it is obtained dynamically; that is, as and when required. The pool area is dynamic in that the composition varies and areas are assigned, released, or made available for use as soon as a program indicates the area is no longer needed.

  Multiple applications may be executing concurrently, so the area of storage that is modified during a subsystem execution must be unique to each message; the actual location of working storage for a subsystem execution is variable. The dynamic pool area in the Intercomm region provides work space known as dynamic working storage.

- Resident Subsystems

  System performance is improved if the most frequently used subsystems are resident. Proper planning will determine which subsystems should be resident to provide maximum throughput and low response time.

- Nonresident Subsystems, Dynamically Loadable

  Nonresident subsystems can be defined as dynamically loadable into the dynamic subpool area. These subsystems are loaded as required. Reusable subsystems remain resident until prescribed message processing limits are reached or message traffic ceases; non-reusable subsystems are loaded for every message.

- Nonresident Subsystems, Planned Overlay

  These subsystems are organized and loaded in a planned overlay structure. The Intercomm region contains one overlay region. (OVERLAY A as in Figure 24.) The OVERLAY A overlay region has special characteristics in that groups of subsystems are loaded to process messages concurrently. Subsystem loading is based on message traffic and scheduling criteria. Subsystems are loaded only when required.

- Nonresident Service Routines

  These are infrequently used routines which are called when they are required. They may reside in the transient overlay area.

- Nonresident Table Entries

  Infrequently used table specifications, for example, those for the MMU and Edit and Output Utilites, can be contained on disk and loaded when in demand.

5.2 OVERLAY AREAS

Intercomm provides an overlay area for message processing subsystems. This area in the Intercomm region aids in providing optimal use of main storage by subsystems organized in a planned overlay structure. The Dispatcher and Subsystem Controller provide overlay management. Once the overlay region is active, criteria for changing overlays (program swapping) are:

- No messages are in process in the current overlay.
- Input sources for all subsystems of the current overlay have either: been checked for messages and none found; or, the maximum number of messages to be processed at one time while the overlay is in main storage has been reached.
- No tasks for the current overlay are on either the Dispatcher execute queues or the timer or event queues.

5.3 DYNAMIC SUBSYSTEM LOAD FACILITY

The dynamic load facility allows Intercomm users the option of dynamically loading and deleting a subsystem or subsystems during execution of Intercomm.

This added function provides both greater flexibility in system design and the capability to correct application programs and make the corrected copy immediately available on-line. The subsystem (or

subsystems) need not be linkedited into the monitor module; a system interface resolves the external references made within each subsystem as it is loaded for processing. An optional dynamic linkedit may be performed at system initilization to modify the subsystem load module and resolve references to resident programs. In addition, further flexibility is provided by the ability to vary the maximum amount of storage defined as usable for concurrently loadable subsystems while Intercomm is executing.

In general, any subsystem defined as reusable is left resident in the dynamic area and rescheduled as required, for as long as the storage it occupies is not required for a subsequent subsystem load. A non-reusable subsystem will be reloaded for every message. Within this framework any reusable subsystem will process a number of messages, if they are available while it is loaded.

There are two methods of dynamically loading a program--by load list specifications, or by a straight load. Intercomm's Asynchronous Loader is used for both methods of loading.

The load list, a faster method, is recommended for frequently used subsystems. However, an additional 50 bytes of storage per subsystem is required. On the other hand, application program substitution is more immediate if the load list is not used and a subsystem is loaded directly.

The substitution feature, the capacity to change or correct a subsystem, relink it and substitute the new version for the old during live execution, is one of the most useful enhancements presented with Intercomm. Substitution may be requested via the LOAD system command and will force reinitialization of the load list.

5.3.1 Dynamically Loaded Subroutines

This facility allows an on-line application to readily use a subroutine that is--transparent to the caller--dynamically loaded. This is of great use to OS users, as it eliminates the need for subroutines to be resident, and to VS users, as it allows subroutines to be dynamically modified while the system is active and may reduce the working set requirements.

Subroutines eligible for this facility are defined by a system macro that specifies whether they are resident or dynamically loaded, and whether they are reentrant, reusable or non-reusable. If dynamically loaded, the macro specifies whether a BLDL list is to be maintained.

By default, the subroutines are deleted when unused (use count=0). Alternatively, subroutines can be deleted only when they have been inactive for a user-specified time interval. On low storage conditions, these parameters are overridden to free storage.

Dynamic linkedit facilities are used to permit dynamic modification of these subroutines in a manner similar to modifying dynamically loaded subsystems. Loading of subroutines is overlapped through subtasking, and may be requested via the LOAD command.

5.4 GENERALIZED SUBTASKING

Nearly all of Intercomm functions as a single task, executing under one Task Control Block (TCB). However, certain Intercomm functions which use OS facilities containing embedded wait conditions are executed as subtasks. The Generalized Subtasking facility manages the subtasked activities. Many Intercomm facilities that require subtasked execution use Generalized Subtasking and, in addition, this facility is available for user programming. A pool containing a user-specified number of subtasks is created (attached) at the beginning of execution, then remains dormant until needed. Any activity requiring subtasked execution is executed using one of these subtasks. Intercomm uses this facility for Dynamic File Allocation, SETL, ESETL and GET processing. Candidate user code for this facility would be those programs having actual or implied waits or a program not suited to normal on-line execution.

Generalized Subtasking allows the subtasked program to appear identical to a normal subroutine. The subroutine is accessed through a macro which simulates a standard CALL type linkage. Intercomm intercepts the CALL and implements the coding under the subtask facility.

Generalized Subtasking can be used to interface software packages with Intercomm that do not conform to Intercomm coding standards and which would otherwise seriously impair Intercomm's performance.

5.5 RESOURCE MANAGEMENT

Intercomm Resource Management techniques provide the optimal environment within which the user may obtain and audit subsystem resource efficiently and with full diagnostic capability in the event of system/subsystem failure. Storage requirements for Intercomm Resource Management are minimal, varying from 1K-5K (dependent upon system load factors and extent of Resource Management involved) and at user convenience, Resource Management facilities can be optioned independently. The extent to which each option contributes to the effective performance of the others should be considered in environmental design.

The foremost resource in any on-line system is dynamic storage. System integrity, as well as system efficiency and throughput, are affected by the integrity of and control exercised upon this resource. Intercomm's main storage management services are intended to ensure integrity and provide efficient use of dynamic storage.

As each message is processed by an Intercomm application program, all system use of main storage for this application may optionally be monitored. If an application program, directly or indirectly, neglects to free the storage obtained during the processing of the message, the system frees it automatically at message termination. Provision is made for CONVERSE time-out which will result in the release of main storage obtained by the application program if the operator fails to respond within a specified time interval. In addition, if an application program tries to free storage it has not obtained for the message currently being processed (or transferred to this message), the system prevents the operation from being executed. In this way, the use of main storage is monitored to prevent bugs in the application program from causing a total on-line system failure.

5.5.1 Storage Cushion Feature

Resource Management includes a storage cushion feature (unless specifically bypassed by the user). The storage cushion, of user-selected size, is defined as a block of main storage, dynamically acquired at startup and held by the system as a protection against temporary shortages of main storage. If a request for main storage cannot be met, the storage cushion is used on the condition that no further threads are activated until the cushion is freed. The degrading effects of a shortage of dynamic main storage are minimized by using this feature.

5.5.2 Auditing and Purging

The optional resource auditing and purging capability provides for a chain of control blocks built for every active thread. These blocks correspond on a one-to-one basis with resources acquired by the thread. If resources are not released by an application program, a thread resource dump is provided to print out the control block chains, showing which thread was in control of what resources such as storage or files, through which module the resources were obtained, and in what order acquisition occurred. Purging is accomplished by releasing the resources represented in the control block chain for an application thread when the thread normally or abnormally completes. After a thread time-out, ongoing I/O is protected via purge prevention until data transfer has terminated.

Intercomm, therefore, is able to provide adequate control over resource ownership with resource audit/purge. Debugging is materially simplified and additional ABENDs are reduced through the cleanup facility, which is activated if a subsystem fails to release resources for any reason, including abnormal end. The thread resource dump is automatically invoked at any program check.

5.5.3 Creation of Dynamic Storage Pools

As a companion to audit/purge or as an independent option, management of core allocation, via user-defined preassembled pools of main storage, is offered with Intercomm. Available space is sectioned by the creation of storage pools in specifiable blocksizes; any number of pools and blocks within pools may be generated to fit user requirements. A status dump is automatically provided when a program check occurs. It consists of a block-by-block listing of the status of the assembled-in pools and storage cushion.

The pool option manages main storage allocation to eliminate fragmentation problems and also, through indexed access to the storage pools, provides a significant efficiency in the speed with which main storage can be obtained and freed. This speed increment is more than sufficient to offset any overhead associated with the implementation of the full scheme of Resource Management, and to permit use of all debugging, analytic and housekeeping aids at no loss in system throughput.

5.5.4 Core Use Statistics

Extensive information on the use of storage pools is provided as an Intercomm Resource Management option. Global and user-selected detail statistics, obtainable as mutually independent facilities, are printed out off-line in a combined format at a user-specified time interval. Inclusion of one set of statistics may be made without reference to the other.

With Release 9.0, the core pools may be assembled and linkedited into a load module separated from the rest of Intercomm. At system startup, Intercomm will prompt the operator for the name of a core pool module to be used during execution, then load that module. This Dynamic Core Pool feature enables the operation staff to employ a different set of core pools at every startup without a complete relink. In addition, it enables users whose load module size is restricted to use the Intercomm core pools without forsaking other features.

5.6 LINK PACK AREA CONSIDERATIONS

Many Intercomm components, such as the File Handler, Dynamic Data Queuing routines, Message Mapping Utilities, etc., are coded in a reentrant manner so as to be eligible for inclusion in the Operating System's Link Pack Area. When using the Multiregion version of Intercomm or File Handler services from batch programs, these components can be put in the Link Pack Area where they can be shared by multiple regions to save on the storage which would be required if they had to be included in each region.

Chapter 6

SYSTEM CONTROL FUNCTIONS

6.1 OVERVIEW

Intercomm provides the user with a number of preprogrammed, fully tested, easily implemented facilities for controlling and monitoring the on-line environment. These facilities (system control functions) are generally table-driven. The user can activate/deactivate system control functions without impact on user applications. The system control functions discussed in this section are:

- Security Controls
- Message Logging
- Checkpoint/Restart Capabilities
- System Statistics
- Control Terminal Facilities
- Testing Facilities

6.2 SECURITY CONTROLS

Intercomm provides the user with complete system security via the following security options:

- Station Sign-on/Sign-off Security
- Transaction Sign-on/Sign-off Security
- Station/Transaction Sign-on/Sign-off Security

Any type of security check, or any combination of the available types, can be employed by the user to control which operator is allowed to enter which transaction from which terminal. Figure 25 shows the general flow of security processing.

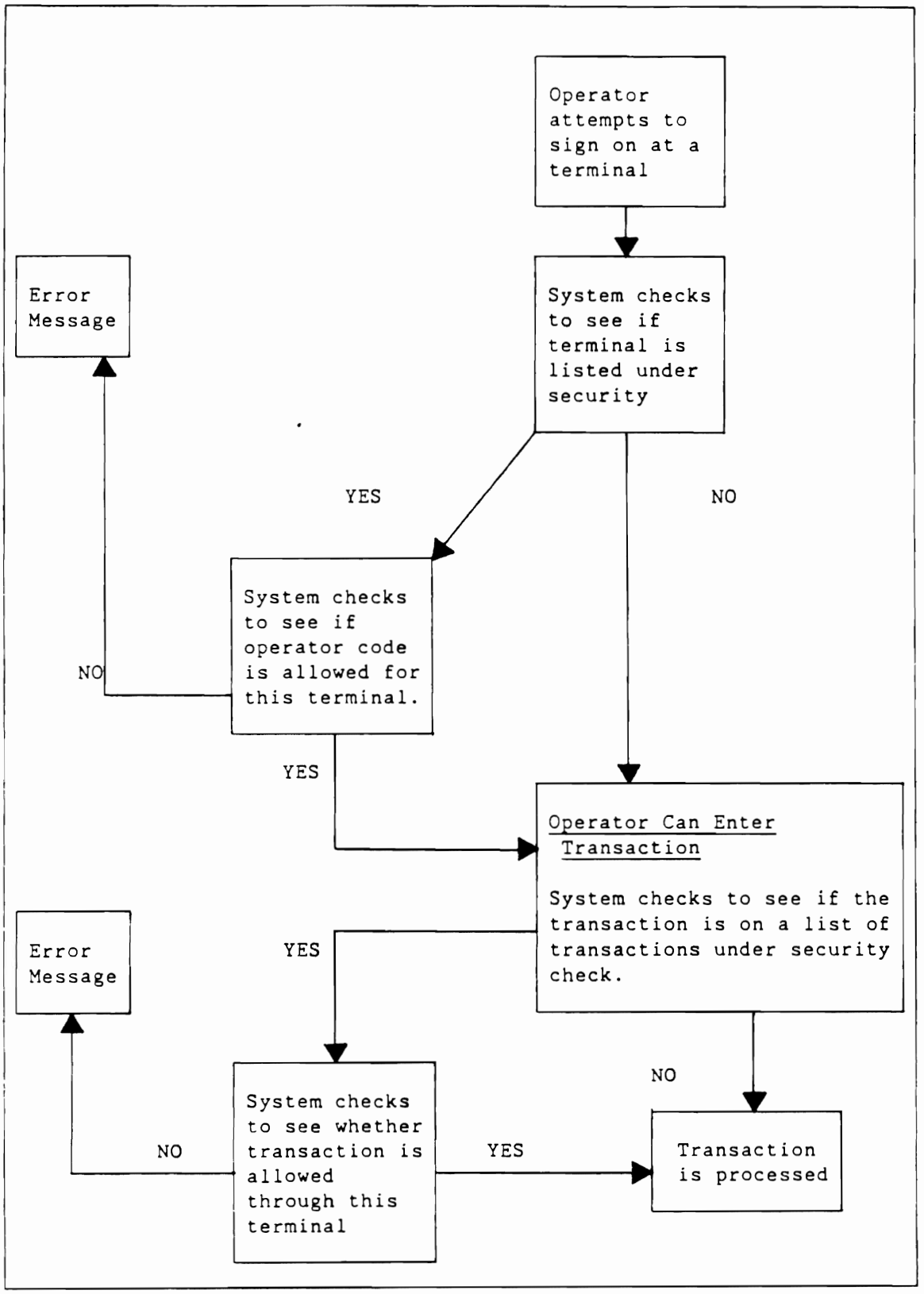


Figure 25. General Flow for Security Processing

6.2.1 Station Sign-on/Sign-off

Intercomm allows the user to assign security keys to terminal operators. For any terminals in the network marked as being a sign-on/sign-off terminal, the operator enters a security code as the first input of a terminal session. This code authorizes the operator at the terminal to enter any transaction for which the terminal is cleared.

An input transaction for which the operator is not cleared causes automatic rejection of the input message. At the completion of a terminal session, the operator signs off to disassociate the authorization from the terminal. In addition, the user may optionally include automatic sign-off based on a time interval stated for the terminal, or a default value taken from the System Parameter Area (SPA).

6.2.2 Transaction Security

The user is allowed to specify the valid transactions acceptable from any terminal in the teleprocessing network. Entry of a nonspecified transaction from a given terminal causes automatic rejection of that particular input transaction.

In addition, certain verbs can be marked as being acceptable only from the system control terminal. Entry of such a transaction from any other terminal will not be valid. The specification as to which transactions are covered by this latter facility is dynamically modifiable via system control commands.

6.2.3 Station/Transaction Sign-on/Sign-off Security

This combination controls which operators may enter which transactions at which terminals.

6.2.4 User-Written Security

Intercomm users can also include their own security checks, in addition to or instead of the security checks provided. This is easily facilitated through the use of strategically placed EXITs in the Intercomm code.

6.2.5 Extended Security System

Intercomm's Extended Security System (ESS) is designed to provide comprehensive control of access to system resources in a multiregion or

single region Intercomm system. The security environment is defined dynamically using the ESS command language, thus eliminating the need to maintain security information in static tables and to interrupt service whenever the security environment changes. Furthur details are provided in Extended Security System.

6.3 MESSAGE LOGGING

The system log (INTERLOG) contains a historical record of all traffic within Intercomm. It provides for system control and complete documentation of performance. It is a variable-length sequential data set which may reside on disk or tape at the user's option. The Intercomm log records the phases in message processing, and additionally and optionally, the file updates performed. (See Chapter 10, File Recovery.)

6.3.1 System Log Entries

System log entries are automatically posted at key processing points. Each message is logged at the time of entry on a subsystem queue via the message queuing facility. The Subsystem Controller posts a system log entry indicating that a message has started processing through an application program. When a subsystem returns control to the Subsystem Controller, a system log entry is posted to indicate completion (normal or abnormal) of the message that was in process. The Front End logs the output message when queued for transmission to the terminal, and when it is transmitted. Based on the time/date stamp of these entries, a timing analysis utility supplied with Intercomm may be used off-line to produce a report of message queuing and processing time. These statistics are generated for messages by terminal and/or by subsystem, and provide system totals. (See Section 6.5, System Statistics.)

An important function of the log is that it can be used as a debugging aid. For example, in testing out a program there may be difficulty in analyzing why it is not functioning properly. Subsystem created entries on the system log can be used to conveniently trace the message path. The log table within the logging routine can be adjusted so that entries which would have been made during testing will be suppressed during live execution (production).

6.3.2 User Log Entries

Aside from Intercomm system log entries on INTERLOG, the user can gather invaluable information about the system from various user log entries. These additional entries can yield statistics about a particular operator or record exceptional conditions in the system. Also, the log can record that file update controls have taken place.

Intercomm allows user programs to place application log entries on the system log data set; these entries can help clarify the status of message processing in the case of system failure. Because application-dependent loggings can be performed at any point during actual processing of a transaction, indication of the point in application processing where failure occurred is possible.

6.3.3 Logging Control

Users can selectively start/stop Front End, subsystem and/or file logging by entry of a system control command. Note that suppression of logging negates restart capabilities and suppression of Back End logging negates charge-back accounting called System Accounting and Measurement (SAM). This facility is primarily intended for use in test systems not normally using recovery or SAM. For these systems, logging could be turned on to gather statistical information for a short duration and then be turned off.

6.4 CHECKPOINT/RESTART CAPABILITIES

The Intercomm standard checkpoint facility enables system integrity to be maintained across a system failure. All internal information necessary is checkpointed and restored by the system. In addition to the information Intercomm checkpoints from its own tables, this feature allows the user to request his own data to be checkpointed; the user can specify a given area of main storage or the size of the user SPA area to be checkpointed by the system. The checkpoint data is written to the log at checkpoint time and, at restart time, the data is restored exactly as it was when the last checkpoint was taken.

The message restart capability of Intercomm provides the most complete message recovery possible with the least overhead. Utilizing the Intercomm log, message restart restarts messages from the point of system failure, restoring the status of all messages that had completed input transmission. Provisions are included so that any message already completed is not reprocessed. The user need only maintain a single BSAM log file for logging (if optioned, file image logging resides on this single log as well), and any size record up to the hardware limitation of the storage device may be written to the log.

Recovery consists of reading the log in reverse sequence and placing uncompleted messages back on the application program or terminal queues. In performing the recovery analysis of message log entries, a message accounting routine contributes to the fast speed of the restart by abbreviating and minimizing the scan of message log entries.

Messages can be classified differently within the terminal or subsystem message set; therefore, selectivity is possible. Messages can be classified as: critical and always to be restarted; desirable and to be restarted if possible; noncritical and not to be restarted. The message restart capability also permits the user to specify the relative importance of message logging on either a terminal or a subsystem basis, or both.

Both the restart and normal modes of Intercomm operate concurrently within a single execution. New messages are accepted from a live terminal network as soon as repositioning of the log to the failure point indicates reprocessing of messages is to begin. Utilization of FIFO queues ensures that processing of live messages does not occur before processing of restarted messages is completed.

The Restart facility is entirely transparent to the application programmer, the only user responsibility being that of table entries made at the system level. Intercomm's message restart is derived from definition of the critical nature of the restart action for each pertinent message thread. Additional status checkpoints taken on the Intercomm log tape decrease the time needed for the recovery program to retrieve messages subject to critical recovery. Thus, the user can obtain urgent restart of critical messages and achieve speedy recovery of the system from failure. With Intercomm Restart/Recovery, the previous processing level can be regained and new message processing can continue without excessive loss of time. The entire message restart system is coordinated with the recovery of files and/or data base management system file data, as appropriate; the File Recovery Special Feature is also implemented through utilization of the Intercomm log.

6.5 SYSTEM STATISTICS

The statistics for performance analysis and system control that are available from Intercomm include the following:

- Terminal Status may be requested from the master control (supervisory) terminal at any time. This status may be for a terminal, a line, or the entire network. (See Figure 26 for sample report.) Messages queued status may also be displayed and used to determine transmission backups, low storage conditions, etc.

- File Statistics may be requested from the master control (supervisory) terminal at any time to assess file organization. Each time the File Handler performs input/output operations, it optionally maintains the statistics of the files. (Figure 27 illustrates the report.) Off-line printing of statistics printed at a user-specified interval is also available.

- Subsystem Statistics are generated on the master control (supervisory) terminal on request. The statistics are by subsystem, showing the number of messages processed by each subsystem and are maintained during the on-line operation by the Subsystem Controller. (See Figure 28.)

- Off-line Statistics may be generated using the message log file created by the system. The statistics generated may be by terminal, by transaction, or by application program, and produce:

  -- Traffic histograms by terminal or subsystem giving number of inputs for each half-hour interval, and a title line showing total input count, maximum terminal or subsystem use and time of maximum use. (See Figures 29 and 30 for sample printout by terminal.)

  -- Response time reports by subsystem including response time summary and response time analysis. (Response time is broken down to one-second intervals and statistics for the subsystems and Output Utility are shown for each hourly interval.)

  In addition to providing an off-line program to analyze the system log, the Intercomm-supplied analysis routine may also print an hourly report of volume in the system.

- Storage Utilization Statistics may be optioned and printed out off-line in a combined format at the time interval specified by the user. These statistics constitute extensive global and/or detail core use information. (See Figure 31.)

- System Tuning Statistics is an optional facility that allows Intercomm to gather information relating to its own performance characteristics, particularly those characteristics that are amenable to tuning via parameter manipulation. This information is maintained in an area within the System Parameter Area. Periodically, accumulated data is printed out to give definitive information that can be input to adjust system parameters. This facility is invoked by inclusion of a printout DD card. The time interval for report printing is user-specified. Minimal overhead effect is experienced with this statistic gathering feature.

- A System Accounting and Measurement Facility (SAM) is provided to detail resources and facilities (including storage) used per subsystem thread or per terminal input. Generated totals may be used for charge-back accounting. Statistics are gathered on-line, stored on the Intercomm log, and reported via an off-line utility.

TERMINAL	LINE	POLL	STATUS	REASON
CNT01	030	0A09	UP	
NYC01	030	0A0A	DOWN	
GRN01	021	0909	DOWN	I/O ERRORS

Figure 26. BTAM Front End Status Display

DDNAME	SELECT	ACCESS	TOTAL	AVERAGE
INTERLOG	527	669	669	1.27
SYSPRINT	0	526	526	
LOGOOO	54	54	54	1.00
STATFILE	87	749	749	8.61
DPDSKQRB	40	22	22	0.55
RCTOOO	6	6	6	1.00
DESOOO	4	4	4	1.00
TOTALFIL	4	4	4	1.00
QUEUEN	21	13	13	0.62
QUEUEU	29	8	8	0.28
VRBOOO	1	1	1	1.00
QUEUEB	6	2	2	0.33
ACCTFIRB	1	1	1	1.00
SUMMARY	780	1059	1059	2.64

Figure 27. File Handler Statistics Report

SUBSYSTEM CODE	TOTAL NUMBER OF MESSAGES PROCESSED	COUNT MAXIMUM USAGE
AA	1046	3
QR	1512	4
.	.	.
.	.	.
.	.	.
TOTAL	15482	36

Figure 28. Subsystem Statistics Display

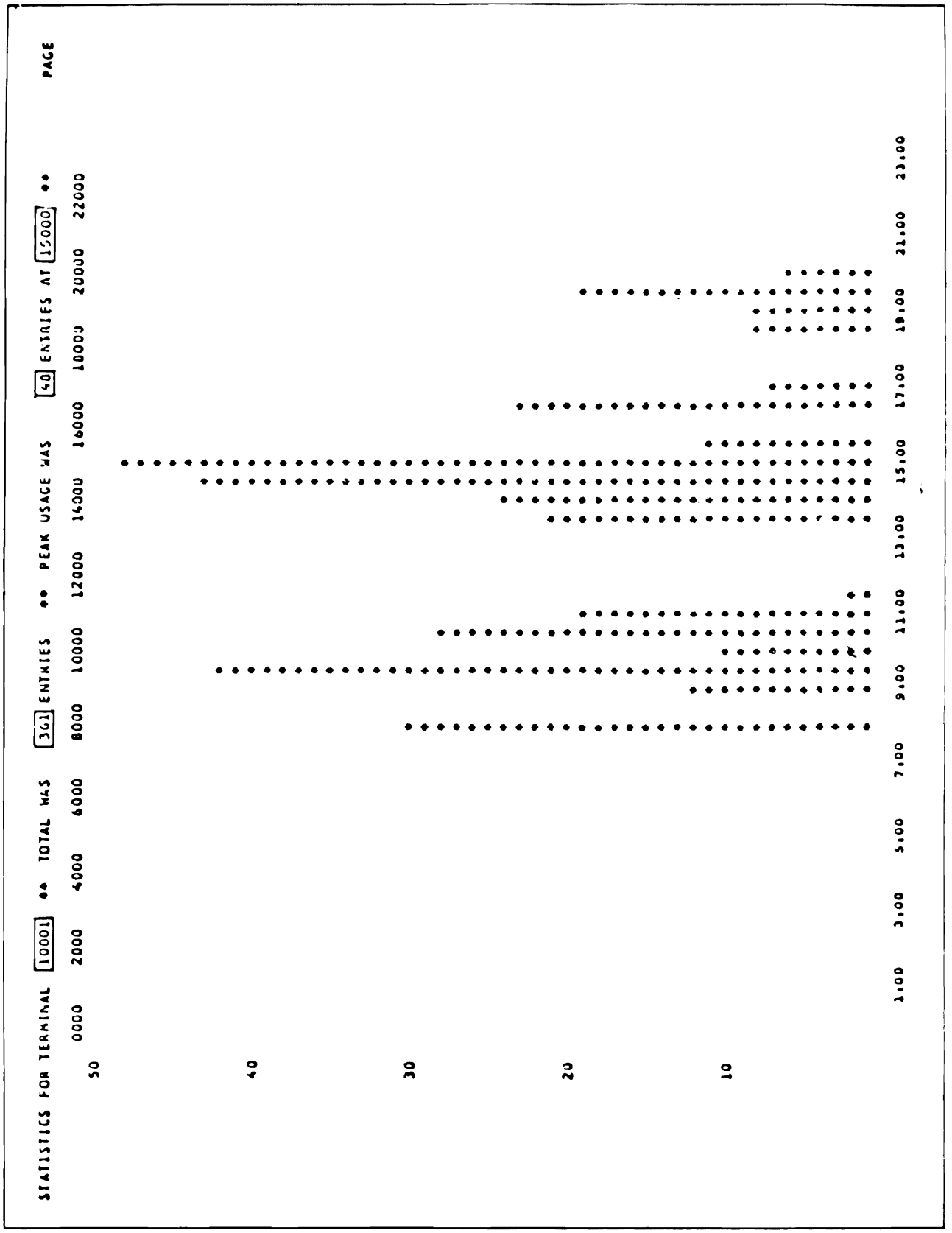


Figure 29. Sample Histogram for a Terminal

STATISTICS FOR TERMINAL 10001 ** TOTAL WAS 361 ENTRIES ** PEAK USAGE WAS 40 ENTRIES AT 15000 ** PAGE 00

SIGN ON TIME # 0601 SIGN OFF TIME # 2005 PEAK SUBSYSTEM WAS ***** EPD1 *****

SUBSYSTEMS		SON0	SON1	IGI0	IGI1	DIS0	DIS1	EBP0	EBP1	EIP2	EPD0	EPD1	EPD2	FSS2	CCL0	CCL1	CCL2	ECC0	ECC1
COUNT		1	1	15	18	12	16	5	6	4	35	41	26	1	13	13	14	19	27
SUBSYSTEMS	EWR0	EWR1	EWR2	CAS0	CAS1	CAS2	INM0	INM1	INM2	ESC2	CCC0	CCC1	CSC3	CSC2	CPD0	CPD1	CPD3	CPD2	
COUNT	3	3	3	6	6	7	5	5	3	6	2	2	2	2	3	3	3	6	

Figure 30. Sample Statistics for a Terminal

CORE USE STATISTICS TIME 15.42.03 86.118

STORAGES ISSUED	53765				
DOUBLE WORDS RECLAIMED	2258221	AVERAGE REQUEST LENGTH	43		
DOUBLE WORDS GRANTED	2258221	HIGH THIS PERIOD	11962	HIGH THIS RUN	12382
TOTAL POOL STORAGE	147880	POOL STORAGE AVAILABLE	103480	PERCENT AVAILABLE	70
REQUESTS FILLED FM ICOMPOOL	46889	PERCENTAGE FM ICOMPOOL	88		
DOUBLE WDS GRANTED FM ICOMPOOL	1904939	PERCENTAGE FM ICOMPOOL	85	AVERAGE LENGTH FM ICOMPOOL	41
DOUBLE WDS WASTED IN ICOMPOOL	71510	AVERAGE DOUBLE WDS WASTED	2	PERCENT WASTAGE	4
ICOMPOOL FAILURES	6871	PERCENT FAILURES	13		
STORFREES ISSUED	53862				
POOL BLOCKS FREED	46599	QUICK FREES	46644	PERCENTAGE QUICK	99
DOUBLE WORDS FREED	2248403	DOUBLE WORDS OUTSTANDING	5818		
REQUESTS NOT FILLED	0	PERCENT NOT FILLED	0		
AVERAGE SEARCH LENGTH	6				
RCB TABLE RELOCATIONS	0				

DISTRIBUTION OF CORE BLOCK SIZES

RANGE	NUMBER OF REQUESTS	CONCURRENCY-- NOW	HIGH	LOW	AVERAGE
1- 64	1992	72	81	10	67
65- 128	14402	51	86	4	56
129- 192	3042	10	21	2	7
193- 256	2644	26	30	0	23
257- 320	23035	4	27	1	7
321- 384	1129	1	8	0	3
385- 448	1404	1	6	0	2
449- 512	85	2	6	0	2
513- 576	445	3	7	1	3
577- 640	312	2	7	1	3
641- 704	65	0	2	0	1
705- 768	20	1	8	0	1
769- 832	2	0	1	0	0
833- 896	86	0	3	0	0
897- 960	53	0	3	0	0
961- 1024	159	0	2	0	1
1025- 1088	1673	2	7	0	1
1089- 1152	4	0	1	0	0
1153- 1216	10	0	3	0	0
1217- 1280	7	0	3	0	0
1281- 1344	2129	10	14	0	10
1345- 1408	4	0	1	0	0
1409- 1472	54	0	5	0	1
1473- 1536	28	0	3	0	0
1537- 1600	6	0	4	0	0
1601- 1664	8	0	4	0	0
1665- 1728	8	0	5	0	0
1729- 1792	103	0	4	0	1
1793- 1856	8	0	4	0	0
1857- 1920	12	1	2	1	1
1921- 1984	330	0	1	0	1
1985- 2048	36	5	7	2	5
2049- 2112	2	0	1	0	0
2113- 2176	1	0	1	0	0

Figure 31. Core Use Statistics (Page 1 of 2)

2177- 2240	1	0	0	0	0	
2241- 2304	0	0	0	0	0	
2305- 2368	0	0	0	0	0	
2369- 2432	2	1	0	0	0	
2433- 2496	259	2	2	0	2	
2497- 2560	2	1	1	0	1	
2561- 2624	0	0	0	0	0	
2625- 2688	0	0	0	0	0	
2689- 2752	74	0	1	0	0	
2753- 2816	0	0	0	0	0	
2817- 2880	0	0	0	0	0	
2881- 2944	0	0	0	0	0	
2945- 3008	0	0	0	0	0	
3009- 3072	0	0	0	0	0	
3073- 3136	0	0	0	0	0	
3137- 3200	1	0	1	0	0	
3201- 3264	2	0	1	0	1	
3265- 3328	3	2	2	1	2	
3329- 3392	0	0	0	0	0	
3393- 3456	0	0	0	0	0	
3457- 3520	0	0	0	0	0	
3521- 3584	0	0	0	0	0	
3585- 3648	0	0	0	0	0	
3649- 3712	0	0	0	0	0	
3713- 3776	0	0	0	0	0	
3777- 3840	0	0	0	0	0	
3841- 3904	0	0	0	0	0	
3905- 3968	0	0	0	0	0	
3969- 4032	30	0	1	0	1	
4033- 4096	9	0	1	0	1	
4097-262136	1	1	1	0	1	

POOL USE DETAIL STATISTICS

BLOCK SIZE	BLOCKS	REQUESTS	FILLED	FAILED	PERCENT FAILED	AVG FREE BLOCKS	DBLWDS ALLOCATED	AVG DBLWDS WASTAGE
32	50	147	65	82	56	3	242	0
64	16	1845	1845	0	0	11	10573	2
96	40	5651	5347	304	6	17	54860	1
128	40	8751	8635	116	2	9	128494	1
160	8	1286	1273	13	2	5	23692	1
192	6	1756	1717	39	3	3	37783	1
224	18	2051	2046	5	1	5	56709	0
256	112	593	593	0	0	102	18234	1
288	2	1008	953	55	6	2	33245	1
304	2	22010	16263	5747	27	1	601839	0
320	16	17	17	0	0	15	668	0
336	16	983	983	0	0	15	41277	0
352	4	6	6	0	0	4	261	0

Figure 31. Core Use Statistics (Page 2 of 2)

6.6 CONTROL TERMINAL

Intercomm includes provision for a terminal to be designated as the control terminal. This terminal can control the operational status of the on-line system. It can be either a specially designated terminal, a normal operator terminal, or the system console. It is notified of all exceptional conditions occurring in the on-line system, for example, program checks, abnormal terminations, and loops, and it has the capability to respond to these situations as required. The command functions available to the control terminal include the following:

- Start/stop of communications lines, terminals, or the network
- Start/stop of logging
- Start/stop of Resource Management statistics
- Open/close of on-line data sets
- Change of security requirements
- Display of various statistics
- Closedown of Intercomm (both normal and immediate)

Network, security, and system control, status and statistics commands are described in System Control Commands.

6.7 TESTING FACILITIES

Intercomm can operate in a test mode via any of the following methods:

- Message processing in batch mode
- Time-oriented simulation of terminals where disk data sets exist for each terminal simulated
- A combination of live and simulated terminals

These test facilities are utilized 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. Transactions are queued and passed into the system at an extremely high rate, causing multithreading to take place in the application program almost immediately. While this facility allows for pseudo high volume testing, it does not represent a projected processing capability based on random message arrival rates from a simulated network. This capability is provided by the terminal simulator.

With the terminal simulator, separate message queues are established on direct access 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 and message traffic a user expects.

The third type of testing facility allows the user to operate with all the terminals of the 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.

As an additional testing facility, the system log from a previous execution may be used to saturate the volume of messages to be processed. Log Input passes messages to the subsystem according to the implied intervals of the log date/time stamps, or by a factor of that time. The Log Input messages are introduced in the same time sequence as the original. Also, the Log Input facility can discard terminal output messages generated as a result of Log Input. As a result, the Log Input feature can also be used in conjunction with a live system. (Naturally, care must be exercised to insure appropriate file integrity when mixing live messages with Log Input messages.) Restrictions on the use of Log Input exist where subsystems make use of specific terminal-ID values since the Log Input output discard facility inserts a special terminal-ID value for all its messages that is then used by the Front End to distinguish output messages generated as a result of Log Input.

In addition, the Model System Generator (MSG) Special Feature allows the testing of system performance prior to coding user application programs. (MSG is discussed in Chapter 16 of this document.)

Chapter 7

FRONT END FACILITIES

7.1 OVERVIEW

The Intercomm Front End has been implemented in a highly modular fashion. The Front End separates the physical management of a device from the logical management of a device. The physical aspects of device management (line control) can therefore be handled by BTAM, TCAM, VTAM, or a hardware Front End interface while the logical device control is still provided by Intercomm. This facilitates an environment where multiple line control disciplines, e.g., BTAM and VTAM or BTAM and a hardware Front End, can be used with a single Intercomm system. Since application programs are coded in a device-independent manner (via use of Message Mapping Utilities or Edit/Output Utilities), the terminal network can be changed without impact on the applications.

This section discusses the standard Intercomm Front End which is also referred to as the BTAM Front End, as described in BTAM Terminal Support Guide. The facilities described, however, are also pertinent to TCAM/Intercomm users since even though the TCAM MCP executes in a separate region, Intercomm still maintains logical control over the devices supported under TCAM. VTAM is discussed in Chapter 20 since it is supported by Intercomm's SNA Terminal Support Special Feature.

Nearly all IBM asynchronous, synchronous and SDLC devices are supported by the Intercomm Front End, such as:

- 2260 series local and remote, clustered and unclustered, CRTs and printers
- 2741 leased and switched
- 2780 leased and switched
- 3270 series local and remote, leased line, clustered and unclustered, CRTs, printers, badge readers, and Selector Light Pen
- 3600
- 3790
- System/370 CPU-to-CPU, leased and switched
- All plug-to-plug compatible devices

Many non-IBM terminals are supported, such as Teletype (TWX) and Dataspeed 40.

TCAM users may operate with most of the terminals supported by that access method.

Additionally, the Intercomm Generalized Front End Interface (GFE), a Special Feature, allows access to nonstandard hardware via user exits in the BTAM Front End. GFE has been used to support CDC/M1000, PDP/11, Periphonics, SUPER NOVA, Varian, and other Front End communications processors.

7.2 MASTER CONTROL TERMINAL

The Intercomm master control terminal is a system supervisory terminal which controls the operational status of the on-line system. It can be either a specially designated terminal, a normal operator terminal which is also used as the master control terminal, or the console. This device is notified of all exceptional conditions occurring in the on-line system, such as program checks and program loops in application programs, and is given the capability to respond to those situations as required.

7.3 START POLL/STOP POLL

This facility allows the master control operator to dynamically stop all BTAM polling of terminals throughout the teleprocessing network. Polling is an invitation from the CPU to a terminal or line group that allows terminal transmission of the message. This applies to all leased lines and their associated terminals.

As a user-specified option, Intercomm may be directed to automatically stop polling under low storage conditions. Users must evaluate the applicability of this feature to determine whether suppression of input would resolve the storage contention problem.

Intercomm also has the facility to dynamically start and stop either a complete line or an individual terminal from being polled. These features give the operator a larger degree of control. When necessary, the operator can start all input again by issuing a Start Poll command.

7.4 IDLE INSERTION

Intercomm's BTAM Front End has a generalized ability to insert idle characters into an outgoing message. This facility allows the user to indicate in the device table, for any buffered device, that idle characters are to be placed in the message after each new line (NL) character.

7.5 SHORT VERBS

Intercomm usually requires a transaction identifier of four characters. This requirement is eased to allow the transaction identifier to be either two, three or four characters in length.

7.6 BUFFER MODE

Users of Teletype terminals frequently deal with long streams of input punched onto paper tape. The user, usually, cannot allocate main storage or processing time to treat the whole stream as one message although it is frequently punched onto paper tape as a single message. As a more efficient method for handling large input messages, Intercomm provides a buffer mode of operation for Teletype users. In this mode of operation, the user defines buffers in a buffer pool not necessarily of message length. As each buffer is filled, Intercomm takes the data out of the buffer's portion of the message and starts filling the next buffer with a subsequent part of the data. Data transmission is not stopped or delayed during this process. This mode of operation allows the segments of the message to be queued and even processed while more data is being received for the same message.

7.7 BACKSPACE CORRECTION

The Intercomm Front End allows the terminal operator using an unbuffered terminal, the same degree of error correction flexibility as the operator using a buffered terminal. Using a buffered terminal, the operator keys information at the terminal, visually verifies the input, enters corrections for errors into the terminal buffer, and then transmits the data to the central computer for processing. Using an unbuffered terminal, the operator cannot visually verify the input since it is transmitted character by character as it is typed. Intercomm's backspace correction feature allows the user, by software, to backspace in the computer buffer and correct errors that are detected in the typed and transmitted data. The application program receiving the message data gets only the correct data. All error data as well as any backspace characters used are eliminated by the Intercomm Front End.

7.8 TERMINAL CONVERSATIONAL FACILITY

This is the ability of the Intercomm Front End to communicate with any terminal in a conversational mode. It is even applicable to those terminals used on an autopoll basis. Conversational mode reduces polling overhead by discarding new input from a terminal until a response has been transmitted to a previous input message. This differs from the conversational facility for subsystems (which is discussed in Chapter 3) in that it applies to all terminals which might attempt to communicate with Intercomm and requires no application program intervention.

7.9 LOCK INITIALIZATION

Users are able to lock a terminal to an application program via a table specification, or an input message generated from the terminal, or a startup program. This capability is useful in situations where terminals are permanently assigned to specific application programs. In a situation where the operator might determine which application to use, or where the terminal might use different applications at different times, LOCK can be used, in coordination with UNLK, from the terminal. Macros that define the Front End to Intercomm may be used to specify the locking relationship between the terminal and the application program.

7.10 AUTOLOCK VERB

The terminal user can direct Intercomm to automatically lock the issuing terminal to a verb on its first entry from a terminal. The autolock facility is useful for transactions which result in a long-term, repetitive execution of one application, such as a data collection operation. The terminal can be unlocked by either the terminal operator or a subsystem issuing the UNLK command.

7.11 SPECIAL AID PROCESSING (For IBM 3270 Terminals)

When certain verbs are entered, Intercomm does not process AID key requests; instead, they are passed directly to the subsystem. This feature is verb-oriented as a user specification. If not specified, Front End Table coding may equate selected verbs with certain keys to substitute for terminal input or to override the input verb.

7.12 3270 GENERAL POLL

IBM 3270 terminals may be utilized in a general poll (rather than a specific poll) environment. With this technique, Intercomm issues a poll for each control unit on a line and allows the control unit to poll its individual units. Under BTAM, General Poll can substantially reduce line use for control sequences and usually improves response time and throughput capacities. Under General Poll, specific devices can be marked logically down with a TDWN command. In this case, output for these down devices is treated in the normal manner for any down device (queued or routed to an alternate device). Input messages from down terminals are discarded.

7.13 GLOBAL WTO ROUTING

Users may exert system-wide control over console operator messages, including control terminal, SYSPRINT, and Multiple Console Support (MCS) destinations.

7.14 FRONT END TABLE VERIFICATION

An optional feature is available for BTAM and TCAM to error check the Front End tables supplied to the system. Most errors that users make in building these tables are trapped by the Intercomm macros used to generate these tables. There are some errors involving complex relationships which cannot be checked at assembly time via the macro definitions. These errors are checked automatically by the system at startup time if this feature is invoked by the user. Errors are reported for resolution by the system programmer and the system is aborted only after all checks are performed.

7.15 NETWORK CONTROL COMMANDS

The master control terminal has the ability to dynamically alter network activity via transactions to start or stop transmission on a line or terminal basis. An alternate terminal can be specified when a terminal is shut down. See System Control Commands.

7.16 ALTERNATE ROUTING

Messages destined for nonoperational devices can optionally be routed automatically to a table-specified alternate device. If no alternate is specified, the messages remain queued until the destination terminal is operational. A network control command can be used to dynamically specify an alternate if required.

7.17 AUTOTPUP

If a terminal is taken out of operation due to transmission errors or other hardware malfunction, the Front End logic allows automatic activation of the device after a user-specified time interval.

7.18 QUEUE FLUSH

A system control command is available to flush message(s) queued for a particular terminal if the user wants to eliminate messages which are no longer of interest to the operator, for example, a multipage report.

7.19 FRONT END CONTROL MESSAGES

An application subsystem can create special control messages requesting verification of successful transmission of a set of messages, or requesting an override to normal CRT alternate input/output processing, or passing a group of related messages for transmission in an efficient queuing scheme (DDQ).

7.20 GENERALIZED FRONT END INTERFACE

The Generalized Front End Interface (GFE) Special Feature allows an Intercomm user to extend the capabilities of the Intercomm Front End to include support for nonstandard devices. The standard Intercomm Network Configuration Table structure is used to define each device. GFE provides interface to user-written routines which perform the actual I/O and related functions required to access the appropriate device(s). The user's access method replaces the access method in the Front End for the nonstandard equipment.

Provision is made in GFE to access user code for the following functions:

- Validate table entries at startup
- Line initialization at startup
- Read/write
- Terminal up/down transactions
- Start/stop line transactions
- Line termination at closedown

Since messages produced by the user READ routines are in turn processed by the Intercomm Front End, all control characters embedded in the message text must conform to a specific device type. Similarly, output messages passed to the user WRITE routine will contain control characters added as required by Intercomm. The user's table definitions for each GFE line must indicate which of the following device types appears to be in use: IBM 2260 remote, IBM 2740 (Model 1 or 2), IBM 3270 (local or remote), IBM 1050. Naturally, any device which simulates or is plug-to-plug compatible with these devices can be used with GFE. See the Generalized Front End Facility.

Input/output messages are formatted as standard variable-length records. Line control characters delineating beginning and end of message are supplied by user code on input and by GFE for output via device-oriented table entries.

7.21 TCAM INTERFACE

An interface to the TCAM MCP region is provided via the GFE Special Feature. TCAM process and destination queues are used for message terminals. Most facilities available for BTAM terminals and commands for network control in Intercomm are available. Detailed information can be found in the TCAM Support Users Guide.

Chapter 8

FILE HANDLER FACILITIES

8.1 ACCESS METHODS

The File Handler makes all standard data set organizations (sequential, direct, indexed) and processing techniques (by logical record, by physical block, and indexed, sequential, or random accesss) available to Intercomm subsystems written in any language. Users may also interface with their own EXCP-level access methods or BPAM via standard IBM macros, replacing WAIT macros with Intercomm DISPATCH macros. File Handler facilities are described in detail in the Operating Reference Manual, access parameters in the Programmers Guides.

The following access methods are supported by the File Handler and are available to all application programs:

- BDAM--fixed-length and variable-length, keyed and direct access
- BSAM--normal support
- QSAM--normal support
- BISAM--fixed-length and variable-length
- QISAM--fixed-length and variable-length (optionally treated by File Handler as BISAM)
- VSAM--keyed, sequential and direct

8.2 BISAM/QISAM REPLACEMENT

One of the more powerful File Handler facilities for ISAM files is the optional replacement of QISAM with BISAM. This transparent capability functions in the following manner. Application program QISAM-type requests are treated by the File Handler as if the request were made for BISAM, thus eliminating multiple DCBs, and allowing record-level range of exclusive control. A large reduction in the use of main storage is realized upon elimination of operating system QISAM routines, and the associated channel programs, I/O areas and control blocks.

If this QISAM via BISAM option is not utilized, the File Handler can support both QSAM and QISAM in an overlapped manner. If specified as a user option, all GETs to QSAM and QISAM files are overlapped with other processing in the Intercomm environment. Without modifying the operating system, Intercomm overlaps GETs to QSAM and QISAM files with other Intercomm productive work, without placing the entire task issuing the GET into a wait state. In addition, the File Handler leaves one QISAM DCB open throughout the day for users of each QISAM file. If concurrent use of the QISAM file by two or more applications is required, a second (or third, etc.) DCB is opened as required. Thus, for the majority of messages, the QISAM DCBs will not have to be opened. This applies only to QISAM since all other access methods with Intercomm require only one DCB for the file.

8.3 VSAM SUPPORT

VSAM support is provided by the File Handler. Subsystems coded to access ISAM files can operate against files converted to VSAM and function with little or no change. However, File Handler service routines developed for VSAM allow application subsystems to take advantage of specific VSAM facilities and VSAM feedback information provided at the completion of file access. KSDS, ESDS and RRDS files may be accessed directly (Key, RBA, RRN) or sequentialy. Generic (partial) keys may be used and alternate index and path processing is supported.

With Release 9.0 of Intercomm, the VSAM Local Shared Resources facility may be employed. This allows the user to specify that a common VSAM buffer pool be built and select the VSAM data sets to share it. Use of this facility can improve virtual storage utilization, and cut down on the number of VS paging operations and file I/Os.

8.4 EXCLUSIVE RECORD CONTROL

At the time of a READ or a GET function, exclusive control of a record is obtainable at the data set level (QISAM), physical record level (BISAM), and by block (BDAM). This prevents loss of updates by simultaneous file updating. (Optional QISAM using BISAM provides exclusive control at the physical record level.) VSAM support provides exclusive control at the control interval level.

An exclusive control time-out feature causes automatic release of records held by a program for longer than a user-specified time interval. This extends to programs that are cancelled, abnormally ended, or program checked. This feature negates the possiblity of a mutual or deadly embrace occurring within the Intercomm environment.

8.5 RESOURCE MANAGEMENT

File resource management is optionally provided by Intercomm resource auditing and purging facilities. A chain of control blocks is built for every active thread, with blocks corresponding on a one-to-one basis with file resources acquired by the thread. Auditing provides a thread resource dump showing which thread was in control of what files, through which module file resources were obtained, and the order in which acquisition occurred. Files are automatically released via the purging facility by releasing the corresponding control block chain for a thread when the thread normally or abnormally completes.

8.6 FILE HANDLER STATISTICS

Each time the File Handler performs input/output operations, it will optionally maintain valuable statistical data reflecting file use. The user can automatically obtain such information as the number of times each file was accessed (separating access into inputs and outputs if desired and/or a separate listing of GETs, PUTs, READs, and WRITEs). Additionally, statistics for a particular file can be obtained by a terminal-entered transaction. With extensive statistics, the report provides a summary line showing activity for all files. Figures provided by the File Handler statistics are cumulative and can be obtained on an interval suitable for the particular Intercomm installation.

If VSAM LSR is used, VSAM buffer pool statistics will also be reported to aid in the tuning of the pool, and may be displayed on-line.

8.7 UNDEFINED RECORD SUPPORT

Undefined record format is supported. The application programmer specifies a parameter in the call to the File Handler indicating RECFM=U file processing. The user is responsible for any blocking/deblocking of undefined records.

8.8 DUPLEXED OUTPUT

Any sequential output data set can be automatically duplexed by appropriate specification to the File Handler. This facility can be used to create duplicate log data sets for safety in failure/recovery situations or to duplicate a critical user file as might be used in a data collection application. This facility is totally transparent to the user program.

8.9 DYNAMIC FILE ALLOCATION

This Intercomm Special Feature implemented through the File Handler is discussed in Chapter 19 of this document.

8.9.1 Dynamic Deallocation and Reallocation via Command

Under MVS and XA, an on-line command exists to dynamically deallocate and later reallocate a file defined in the Intercomm execution JCL. This makes it possible to avail a file for batch and on-line use alternately without stopping Intercomm execution.

8.10 SEQUENTIAL FILE ABEND PROTECTION

Under Release 9.0 of Intercomm, the user may select sequential disk output files to be protected against out-of-space (B37,D37 ABEND) conditions. Out-of-space conditions for these files will be trapped by Intercomm at which point the file-handler will switch to output to a user-specified alternate data set. As a result, no data is lost and monitor execution continues without interuption.

8.11 BATCH SUPPORT

A batch program can use the File Handler for file access by coding CALL sequences as in an on-line program and providing the same control blocks and parameter lists. The batch program must be linked with the File Handler and other requisite system routines as described in the Operating Reference Manual.

Chapter 9

INTERCOMM AND MVS

9.1 INTRODUCTION

Intercomm has been extensively enhanced to achieve the highest possible performance in a virtual storage environment when executing under MVS or XA. Intercomm is designed to enable the user to take advantage of special features available only in MVS operating systems. Additionally, all the Intercomm versions are fully operational under VM.

9.2 THE PROBLEMS OF VIRTUAL STORAGE ENVIRONMENTS

The IBM virtual storage operating systems provide many powerful capabilities to users, and in many cases can increase overall computing performance. A serious problem is that the techniques to implement the virtual systems' capabilities can seriously hinder teleprocessing performance.

By having units of the workload swapped or paged in and out of real memory, a computer can execute a larger workload in virtual storage systems than in non-virtual storage systems. The paging technique can increase the total throughput production of the computer. However, while paging attempts to optimize throughput of the entire workload, the performance of individual jobs can be affected negatively, particularly the teleprocessing operations where individual terminal response time is critical. In a virtual storage system, when non-real storage is referenced, the teleprocessing monitor loses control to the operating system until the needed storage can be paged in. Since teleprocessing systems with many applications can become quite large, it is important to minimize the performance-degrading effect of page faults.

The hardware/software techniques used by IBM's VS operating systems have increased the CPU time required for processing most computing jobs. This increased CPU time has resulted from more costly operating system services, some of which represent moderate increases, others drastic increases. Unfortunately, teleprocessing monitors usually use more of those costly services and thus tend to incur high CPU costs. Special techniques in the monitor can lessen this effect.

An additional requirement of a VS teleprocessing monitor is support for special VS features, such as the VSAM access method and the comprehensive Systems Network Architecture (SNA) environment with its VTAM access method. VSAM is supported by the File Handler in the VS versions of Intercomm. Intercomm's support for the SNA environment, including VTAM, is a Special Feature which is described in Chapter 20.

9.3 THE INTERCOMM PERFORMANCE SOLUTION

Intercomm provides a set of powerful facilities to provide optimum response time and throughput capabilities in a virtual storage environment. These features have been custom-implemented to provide the greatest on-line performance improvement with the least impact on other workloads within the computer.

9.3.1 Virtual Storage Scheduling/Fast Path

The optimum program scheduling/loading technique recognizes that the hardware assist facilities of VS are most appropriate for managing the many and varied on-line application programs. Therefore, Intercomm's Virtual Execution Group storage scheduling utilizes operating system paging to manage program residency. Virtual storage scheduling optimizes page loading to ensure the highest level of page utilization with the least number of page faults. For designated high-priority programs, Intercomm's Fast Path virtual storage scheduling technique provides absolute minimum response times. This technique is designed to replace the cumbersome OS Overlay loading while providing the same control of clustered subsystems for message processing.

9.3.2 Anticipatory Page Loading

To supplement the Virtual Execution Group storage scheduling, Intercomm's Look-Ahead-Page-Load facility checks each critical page boundary movement to insure needed pages are resident in real memory. Thus, when a potential page fault is detected, Intercomm can pre-load the necessary page(s) overlapped with processing for other message traffic. Significantly, this facility loads pages only as required, thereby avoiding massive and unnecessary page loads when a message is originally received by the system. This facility may not be used in an XA system.

9.3.3 Specialized Main Storage/Table Management

A well-designed teleprocessing system must use the minimum number of main storage pages necessary for message processing. Intercomm's VS models have been designed to accomplish this through the following techniques:

- Virtual Storage Scheduling

  The Fast Path/Virtual Execution Group storage scheduling facilities (see Section 9.3.1) minimizes main storage utilization for program residency requirements.

- Resource Management

  Intercomm's powerful Resource Manager provides special main storage allocation for buffers, work areas and dynamic areas that minimizes paging by grouping often-used areas contiguously. Intercomm avoids GETMAIN/FREEMAIN processing that is not only costly but can fragment virtual storage.

- Table Management

  Special table management coding groups related table entries in the same areas of virtual storage and avoids wasteful paging to scan tables.

- Modular Construction

  The highly modular Intercomm system (over 500 separate modules) is well-suited for efficient execution in a virtual storage system, and for linkedit ORDERing of the most frequently used modules (Control Sections).

Chapter 10

FILE RECOVERY

10.1 INTRODUCTORY CONCEPTS

Intercomm has been designed to anticipate, detect and recover from most error situations without bringing down the entire teleprocessing system. Teleprocessing devices and teleprocessing application programs can and will fail. In most instances following failures, Intercomm continues to operate in a degraded mode, minus the failed components. Alternatively, Intercomm can come down gracefully after a failure by completing all work in progress at the time of failure.

Certain conditions, however, can and will occur that effect immediate termination of all processing in Intercomm. These include power failure, machine failure, data base destruction, and operating system failure. In these and other total failure situations, Intercomm automatically provides for the complete recovery of teleprocessing applications. This recovery includes the restarting of all messages in progress at the time of failure, the recovery of message queues, the recovery of checkpointed data, and the coordinated recovery of files and data bases.

Key points to the Intercomm Restart/Recovery facility are summarized below:

- No User Support Required

  All mechanics of implementing message restart and file recovery are supported by Intercomm-supplied software. User application programs that are to be restarted are no different in format or content than nonrestarted programs. The restart facility is generally transparent to the application programmer. User responsibility in restart is limited to table entries that delineate those programs and terminals for which restart is to be performed. Other table entries specify those programs that may update data base files. These table entries are generally the extent of user responsibility in providing restart capability.

- Fast Restart

  The method of recovery in Intercomm is a rapid "warm" restart. It does not require an off-line program following a failure; live Intercomm is merely restarted. Restarted Intercomm first reads the system logging (journal) file backwards from the point of the failure as far back as necessary for recovery. With the necessary items recovered, live Intercomm starts up, typically only a few minutes later.

- Data Integrity

  The user is afforded complete data integrity following a failure. If a file or its indexes were physically or logically destroyed as part of the failure, the file can be reconstructed from its last backup by Intercomm-supplied software. If the file is intact following a failure, the file is recovered in such a manner that updates in progress at failure time can be reinstated without duplication. The recovery is coordinated for data files, DBMS files, checkpointed system table entries, and message traffic.

- Message Integrity

  All messages and queues are recovered and restarted as appropriate. In certain cases, to ensure data integrity, subsystem messages previously completed must also be restarted to effect necessary interaction with files or data bases.

- Selective Restart

  Users can specify the extent of the restart on a terminal-by-terminal and program-by-program basis. The overhead of restart can be limited to necessary system components, via the restart "always" or "never" specifications. Additionally, a third specification is provided; the restart desirable condition. If restart is desirable, but not mandatory for a program or terminal, then in the course of restart for those 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 or not desirable components have been fully restarted. Restart will proceed as far backwards through the Intercomm system log as 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. However, the read-back point of the file is defined as the point necessary to retrieve all mandatory items and is not influenced by desirable items. Those terminals or subsystems for which the restart specification is neither desirable nor mandatory utilize logging as a user option, again on an individual program and terminal basis. Selective logging can reduce system overhead in logging that would otherwise be wasted on noncritical components.

- Coordinated Checkpointing

  As described in the Operating Reference Manual, checkpoints are taken, at a user specified interval, of critical system table entries and counters. A pointer to the checkpoint file data is logged on the system log at checkpoint time. Checkpointed system data, and optional user data, is also restored during message restart.

- Single Log File

  Only a single log is required for Front End, Back End and file image logging. This log can be either tape or disk. Further, a technique for writing on this log has been implemented such that any size record can be written to this log (up to the block size specified for the log file INTERLOG) without devoting buffer space for maximum record sizes. Additionally, log records are blocked for noncritical entries and unblocked for critical entries.

- Log File Recovery

  For tape and/or disk logging, an off-line utility, ICOMFEOF, is provided to determine the end of file after a system failure where the operating system does not properly close the file. For on-line disk logging, a sequential file Flip/Flop facility (x37 abend recovery) is provided to handle logging in smaller file increments than an entire day (or days). Both of these features are described in the Operating Reference Manual.

In the following sections, message and file recovery concepts of the Intercomm system are described. All aspects of File Recovery under Intercomm, including specifications for coding control information, are detailed in the File Recovery Users Guide. The last section describes the on-line Backout-on-the-Fly facility for immediate file recovery after application program failure. Data Base file recovery is described separately in the DBMS Users Guide.

10.2 MESSAGE RESTART

Intercomm is an event-driven system whereby work activities are initiated in response to a message. Therefore, the core of Intercomm 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. INTERLOG contains entries for all messages that are subject to recovery. It includes entries that make possible a determination of message status at the time of failure. Message status is an important factor in determining the read-back point and is defined by one of the following categories:

- Received and completely processed prior to the last checkpoint;
- Received and completely processed subsequent to the last checkpoint;
- Received and in process at failure;
- Received but processing not started.

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 permits backward reading to proceed only as far as is necessary to retrieve those messages needed for restart.

When messages are recovered from the log they are placed on the queues for their destined subsystems or terminals as the restart phase concludes. Thus, queues are rebuilt, not recovered, following failures.

The restart process is initiated when the RESTART parameter is found in the PARM field of the Intercomm execution (EXEC statement) JCL. This RESTART parameter is the only change required to distinguish a restarted Intercomm run from a normal Intercomm run. When a RESTART is recognized, the restart phase of Intercomm analyzes the restart log and rebuilds the queues. Then the normal mode of Intercomm starts reprocessing messages placed in the queues by restart, while simultaneously receiving and processing messages from the live terminal network (FIFO queuing insures that restarted messages are processed prior to live messages). Optionally, a serial restart facility (described in the Operating Reference Manual) may be used to force processing of all restarted messages prior to accepting new input from the network.

The entire Intercomm message recovery system is coordinated with the recovery of data files and/or DBMS files, as appropriate. In some circumstances, an original message from a terminal will proliferate messages to many subsystems as a result of a design approach utilizing subsystem-to-subsystem message switching. Thus, one or more chains of processing will occur in response to the original message. If two or more subsystems in these chains may make potentially interdependent changes to the same data files, then the corresponding Subsystem Control Table (SCT) entries must all specify LOG=YES and RESTART=YES. This will ensure that the originating (mother) subsystem is restarted if any subsequent program did not complete, while the daughter subsystems are not. Therefore, the entire processing chain is restarted from the beginning.

10.2.1 Message Logs

The log facility for Intercomm utilizes a data set whose ddname is INTERLOG. This log contains entries reflecting the status of messages for subsystems and terminals, and entries of before and after images of data files being updated. Also included on INTERLOG are user log entries, checkpoint records, message accounting records, etc. (See Figure 31).

INTERLOG may be specified on its DD statement as being tape or disk. If the DD statement for INTERLOG is omitted then no logging will be performed and of course no restart is possible. The computer operator is notified at startup time if INTERLOG is not specified, or if the file cannot be opened. If you intend to use the restart facility, the computer operator's response to this message should be to cancel Intercomm and request programming assistance before proceeding.

INTERLOG records appear as standard undefined records (that is, RECFM=U) but can be read using QSAM (RECFM=VB). Special techniques are utilized in creating the log, as follows:

- Access Method

  BSAM rather than QSAM is employed.

- Variable Buffer Size

  "Average" length buffers (specified via the system SPA table) are normally used. Where possible, log records are treated in blocked mode. However, if a record to be logged exceeds the average buffer size, it will be logged in its own buffer of the appropriate length. Any size record (up to the DD statement BLKSIZE specified for INTERLOG) can be logged.

- Synchronous Logging

  Recognizing the need for priority in logging items pertaining to critical user components (critical relative to restart), certain records are written immediately. For example, log entries for an important subsystem are made immediately by adding the entry to the current buffer, then immediately writing out the buffer. INTERLOG is blocked for noncritical items, and effectively unblocked for critical items. Critical subsystems and critical terminals are designated via macro parameters in the associated tables. File recovery records are automatically logged synchronously.

Logging and restartability for terminals, subsystems, and regions (in a Multiregion environment) is controlled by macro parameters in the associated tables. Figure 31 describes the Intercomm log records. See the Restart Use column for those log entries that are utilized in Restart/Recovery. All other log entries, including user log entries, are ignored.

10.2.2 Message Accounting

To make the warm restart concept function as rapidly as possible, restart involves reading the log backwards only as far as is required to recover all necessary messages. This "how far back" information is developed by the Message Accounting routine, MSGAC, a subprogram of LOGPUT, the message logging routine. MSGAC operates as part of the live Intercomm environment. LOGPUT examines every log entry to determine if this new entry reflects a change in the "read-back point" of the log tape. For every 255 completed messages, MSGAC will insert message accounting records onto INTERLOG.

These records reflect the read-back point. When restart begins reading INTERLOG backwards, the first message accounting record encountered will identify the location of the actual read-back point.

10.3 MESSAGE RESTART LOGIC

When data base or file recovery is not utilized, the restart logic is quite simple. When reading the log 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). As the log is read backwards, message log entries will be encountered in this order:

- Subsystem Completed (normally or abnormally)
- Subsystem Started
- Message Queued for a subsystem (header/text entry)

If the message completed successfully, was flushed, or failed security, the message is not restarted. If the message failed in processing by a subsystem (time-out, program check, etc.) then this message may be restarted. If the message began processing but had not completed at the time of failure, then the message may be restarted and its log code on the queue is set to "R" indicating that it was an in-process message being restarted.

The above rules are the criteria applied to a single message out of context; they may be overridden by the considerations listed below:

- Ancestral Messages - if any "ancestor" of a message has been restarted for any reason, the message is discarded. This rule requires some clarification: if during the processing of message A, the subsystem generates message B and queues it for processing via MSGCOL or FESEND, 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 message to its mother, the mother's mother, and so on. These are collectively the message's ancestors. A message is only restarted if all its children are discarded. This applies to Front End as well as Back End messages. However, if the ancestor is not logged, or not marked for restart, only the child (subsystem or terminal output) is restarted.

- Conversational Messages - if the message is part of a conversation, that is, part of a subsystem calling CONVERSE, and so specified on the subsystem's table macro, the message is 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 did not complete).

- Lost Messages - messages that are lost because queues are full will not be restarted.

In a Multiregion Intercomm system, message restart and file/data base recovery is possible only in those regions which create their own Intercomm log. In Satellite Regions, only 01-30-FA logging of subsystem processing is done, along with log records necessary to message restart, checkpointing and file recovery. In the Control Region, terminal transmission logging (F2-F3) and Multiregion message queues (for Satellite Region transmission) are rebuilt at restart time. Further details are described in Multiregion Support Facility.

In cases where data base or file recovery is not included, the only integrity problem concerning a restart involves those messages that were in process at failure time. Thus, if a message was being transmitted when a power failure occurred, the restarted Intercomm would retransmit that entire message or DDQ (FECMDDQ request). An input message that was being received, and not yet logged, must be resubmitted by the terminal operator.

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 31. 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; 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 31. INTERLOG Entries (Page 2 of 2)

10.4 FILE RECOVERY CONCEPTS

The previous section described the concepts utilized in Intercomm's restarting of messages where data base or file recovery was not involved. This section concerns the recovery of files and the coordinated recovery of related and unrelated Intercomm messages. The restart concepts previously presented for messages are also the basis for file restart/recovery. Restart facilities have been extended to provide for the special requirements of data file integrity following system failures.

The chart below lists file access method support by the File Recovery facility.

Access Method	Supported
BDAM	Yes
BISAM, QISAM	Yes
VSAM	Yes
IAM	Yes (ISAM)
BSAM, QSAM	No

The basic concepts of File Recovery also apply to specific data base managers supported under Intercomm. For additional details on Data Base Management System recovery, refer to the Intercomm DBMS Users Guide.

The focal points for file and data base recovery capabilities are the Intercomm checkpoint and log files, and the data base activity log.

10.4.1 Checkpoints

The Intercomm checkpoint is not a "checkpoint" in the normal data processing usage where a picture image of core is written as a checkpoint to some data set. Rather, the Intercomm checkpoint function saves only a few critical table entries on disk, involving only minimum I/O activity for the checkpoint (data saved in one checkpoint record). To Intercomm, the checkpoint has special significance relative to file recovery in that it indicates all Intercomm subsystems performing file updates have been quiesced (no new messages started) until checkpoint processing has completed.

The Intercomm checkpoint functions as follows:

- Users specify a checkpoint time interval such that when this interval expires the checkpoint process begins;
- All subsystems that may at any time perform an update activity to a file that is to be recovered are identified by the subsystem SCT entries. During checkpoint, these update subsystems are marked as nonschedulable so that no new messages will be started through these subsystems. (Messages can be received for these subsystems but will remain in the queues.)
- Intercomm examines each of these update subsystems for current message processing activity. When all activity for these subsystems has concluded, the checkpoint can begin. Meanwhile, all other activity such as nonupdate subsystems, terminal I/O, the Output Utility, etc., continues without interruption.
- A checkpoint record containing pertinent table data is written to the checkpoint data set.
- A checkpoint record is written to the Intercomm log noting the date/time of the checkpoint.
- Subsystems that had been marked nonschedulable are started up again. Normal file update activity resumes.
- The interval timer for the next checkpoint is set.

From the above it can be seen that a checkpoint merely represents a "clean point in time" to which a restore can be made. At this point in time the file was intact on disk and no modification or update activity was in progress. The basis for file recovery is to proceed backward to recover files from the point of failure to this checkpoint, then restore the checkpointed table information, and finally to proceed with the live system.

10.4.2 File Activity Logging

The Intercomm File Handler provides for update (delete, add, insert) activity logging for BDAM, ISAM, and VSAM files. These log entries include before- and after-images that are used to recover a file. The file activity log is the same data set (INTERLOG) for Intercomm File Handler entries as for message status entries.

10.4.3 Destruction of Files

As part of the system failure, it is possible that all or a portion of a data file (and its indices) are physically or logically destroyed. In this case it is the user's responsibility to restore the file back to its condition at failure time. This is accomplished in two distinct operations:

- first, the last backup (complete copy) of the file is reloaded, that is, an off-line file restore (tape restore, tape to disk) is performed;
- second, all after-images from the Intercomm log are applied to the file in their original order by an off-line utility included with the Intercomm File Recovery facility.

When the above procedures have been performed, the file is returned to the status at failure time and normal recovery processing can proceed.

10.4.4 Normal Recovery

If a system failure has occurred without destroying the file, normal (on-line) recovery can be immediately started. (If data destruction occurred, then the above stated complete recovery must first be performed before proceeding with normal recovery.) Normal recovery involves backing out file changes to the last checkpoint. (Since Intercomm checkpoints are typically rapid with little performance interference, checkpoints can be frequent.) With frequent checkpoints, normal recovery need not involve much data restoration.

The backing out to checkpoint process involves applying before file images to files in the opposite order (backwards order) to their original update. Therefore, updates are reversed back to the checkpoint time. Additions are reversed by deletion; deletions are reversed by addition.

10.4.5 Coordinated Message Recovery

While the normal file recovery is being performed, Intercomm message recovery proceeds concurrently. The coordination between Intercomm message and file recovery is software controlled, transparent to the computer operator. File recovery is performed by the Intercomm restart itself, and thus proceeds simultaneously with message restart.

As part of data file recovery, the files subsequently need to be brought from the time of the checkpoint forward to the time of the system failure and into live mode. This is implemented by expanded Intercomm message recovery. Message recovery was previously described in detail. Essentially this message recovery involves returning to the queues those messages that were in the queues at failure time, and entering into the queues those messages that were in process at failure time. Those messages that were partially processed are requeued with a special log code of "R". Thus, only partially processed or totally unprocessed messages are normally put into the message queues following restart. However, if file recovery is involved, Intercomm will also restart those messages causing file updates which had completed but had done so after the last checkpoint was taken.

Thus, message restart involving file recovery proceeds as follows:

1. If necessary, complete recovery is performed off line including file reload and application of after-images for any destroyed file.

2. Data files are returned to checkpoint time status by backing out updates (before-image operation).

3. Concurrently with point 2, Intercomm recovers messages that were

   - queued but not started
   - started but not completed (may have updated a file)
   - completed but had updated files since the last checkpoint

4. After points 2 and 3 are completed, restarting of message processing begins, and updates that had been reversed by returning to checkpoint will be reapplied as the messages that caused those updates are reprocessed.

5. Concurrently with point 4, Intercomm resumes live mode.

6. If the Serial Restart Facility is implemented (see the Operating Reference Manual), then live mode Intercomm may be selectively controlled or postponed until point 4 is completed.

Output messages generated while processing restarted messages will be transmitted. Subsystem logic can identify restarted messages via the log code (C'R') in the message header and prepare special message text as required, to notify the terminal operator of the possibility of duplicate output.

10.5 BACKOUT-ON-THE-FLY

Backout-on-the-Fly provides on-line dynamic reversal of file updates following a subsystem failure, and is executed following the occurrence of these situations:

- Subsystem thread program check
- Subsystem thread timeout
- Specific requests by a subsystem

Backout-on-the-Fly follows the same methodology as file recovery: before-images are applied to updated files in the appropriate reverse sequence. The facility differs from restart file recovery in that Backout-on-the-Fly compares the failing subsystem's "after-images" with the file's current images to ensure that no intervening subsystem subsequently updated the same record. In the event of a mismatch of after-to-current image, an operator reply to a console message can choose to either abend Intercomm or ignore the situation. If the abend is chosen, the normal file recovery scheme can be used to successfully apply the before-images back to the last checkpoint.

Backout-on-the-Fly requires the Dynamic Data Queuing Facility. Backout-on-the-Fly places the thread's before- and after-images on a DDQ. If the thread completes successfully, then the DDQ is deleted. If the thread fails, then reversal is performed.

Backout-on-the-Fly logs and reverses all files marked as reversible to standard file recovery through the normal file recovery control options. Additionally, Backout-on-the-Fly is selected by subsystem. The overhead (creating and writing to DDQs) is incurred for only those subsystems deemed likely to fail. However, Backout-on-the-Fly will log only File Recovery's recoverable files regardless of selected subsystems.

If all file logging is shut off through the system "stop log" command, Backout-on-the-Fly remains functional. If a subsystem performs both file and DBMS updates, then Backout-on-the-Fly may be incompatible. Exceptions are ADABAS and IDMS, which also provide a comparable facility. SYSTEM 2000 may be compatible if all subsystems use deferred updates as a programming standard, with the last subsystem action prior to GOBACK being to apply updates. DBMS situations should be examined carefully when using Backout-on-the-Fly.

See File Recovery Users Guide for implementation procedures.

Chapter 11

DBMS SUPPORT

11.1 INTRODUCTION

The Intercomm/Data Base Management System (DBMS) interfaces are Special Features which allow the user to implement an efficient data base/data communications (DB/DC) environment. With Intercomm, the user is able to independently select the optimal DBMS (one or more) for the application environment with the knowledge that provision has been made for fully integrated support which includes DBMS restart/recovery (up to the maximum inherent capabilities of the DBMS). The Data Base Management Systems for which Intercomm interfaces currently exist include:

- ADABAS--a product of Software A.G. of North America, Inc.
- DL/1 (IMS DB)--a product of IBM Corporation
- IDMS--a product of Cullinet Corporation
- Model 204--a product of Computer Corporation of America
- System 2000--a product of S.A.S.
- TOTAL--a product of Cincom Systems, Inc.

For details on the specific Data Base Management Systems, please consult the appropriate vendor.

In addition, a Generalized DBMS Interface (GDB) is available with Intercomm. GDB provides a framework from which a user-developed or other commercial DBMS might be supported with Intercomm.

Three different approaches have been used in providing Intercomm/DBMS support, as described in the DBMS Users Guide. These are illustrated in Figure 32 and are as follows:

- Customized Interface

  Specially developed support for a specific DBMS. This category applies to DL/1 support (no vendor routines required) and TOTAL support (vendor routines required).

- GDB Interface

  Vendor-developed support which requires Intercomm GDB support. This applies to Model 204. The DBMS vendor supplies the interface routines.

- Vendor-Written Interface

  The support is provided totally by the DBMS vendor, in consultation with the Intercomm staff and uses Intercomm facilities such as the Dispatcher to await completion of the Data Base activity requests. Such Data Bases are IDMS, ADABAS and System/2000. Therefore, a fee is still charged to users of the interfaces, even though Intercomm includes no specifically provided interface routines, but does provide calls to vendor-written entry points for startup, closedown, purging, etc.

The techniques used to accomplish each DB/DC integration also vary according to the specific DBMS. Generally, the interfaces include six major functional areas:

- On-line Initiation

  A communication path between Intercomm and the DBMS is established and data base sign-on procedures are performed.

- On-line Termination

  An orderly disassociation is made between the DBMS and Intercomm.

- Command Processing

  Requests for DBMS services are made by user application programs and are executed by the DBMS.

- Contingency Processing

  A program is executed after failure of Intercomm, an on-line Intercomm application, and/or the DBMS in order to resolve the problems of continued operation.

- Checkpointing

  A quiesce is issued by Intercomm and/or the DBMS in order to establish a common point from which to do a restart.

- Recovery

  A reconstruction of the data base and Intercomm environment is made prior to a restart from a failure condition.

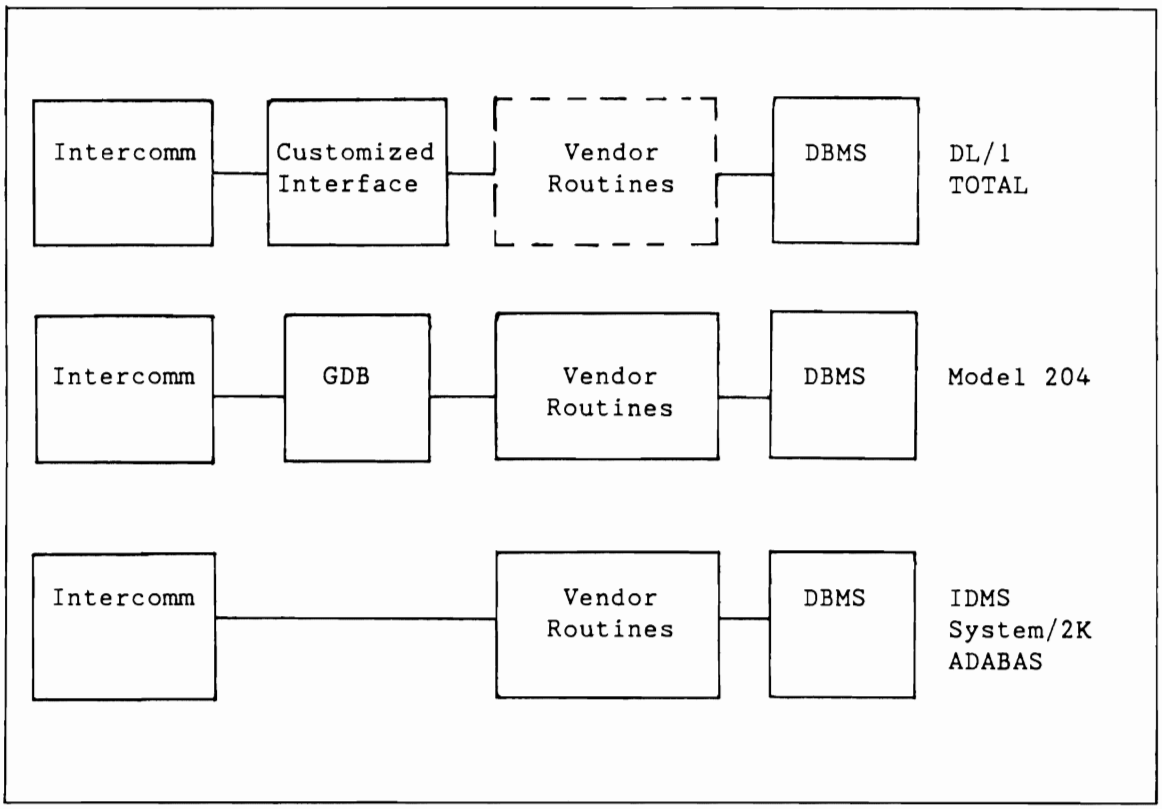


Figure 32. Design Approach to DBMS Interface Support

11.2 DBMS INTERFACE ENVIRONMENT

The on-line DB/DC environment with Intercomm and a Data Base Management System consists of:

- The Intercomm region and associated user application programs requesting access to the data base
- The DBMS region controlling all data base access (in some cases the DBMS may be ATTACHed as a subtask in the Intercomm region)
- Batch regions requesting access to the data base

All Data Base Management System functions are centralized within a region separate from the Intercomm or batch jobs. The DBMS region usually supports concurrent use of its facilities by all jobs, whether these separate jobs access the same or different data base files.

Intercomm is one of multiple possible users of the DBMS, each of which resides in a separate region. The Intercomm region consists of the Intercomm monitor along with all on-line application programs, including those which are using the DBMS for file access. Other DBMS users might be batch appplication programs, each residing in a separate region. All logic required for data base access and control (exclusive control logic, data base logging logic, etc.) is contained in the DBMS region; thus, there is no duplication of code in other regions. The only logic required in each user region is the interface logic for communication of data base requests to the DBMS region.

As depicted in Figure 33, if the Intercomm Multiregion Support facility (a Special Feature) is used, multiple Intercomm regions may be accessing the DBMS region. Certain restrictions exist in this environment, particularly when data base updates are performed; please refer to the Intercomm publication, Multiregion Support Facility. The multiregion version is required to support use of multiple DBMS.

Whenever a data base function is required by either a batch program or on-line application subsystem, the program involved need only call the interface module in its own region; the request is passed on to the DBMS via an interregion SVC (Supervisor Call). For an inquiry-only application, the required interface components are simple since the data base is not being altered. The only necessary recovery procedure is a dump/restore capability to be used in the event the data base is totally destroyed.

For the on-line DBMS update environments, a method of maintaining data base integrity is necessary. This is provided through a combination of Intercomm and DBMS checkpoint, logging, and restart facilities. Whatever the failure condition, e.g., destruction of data base, failure of a batch program, etc., the DBMS user is guaranteed the ability to fully reconstruct the data base.

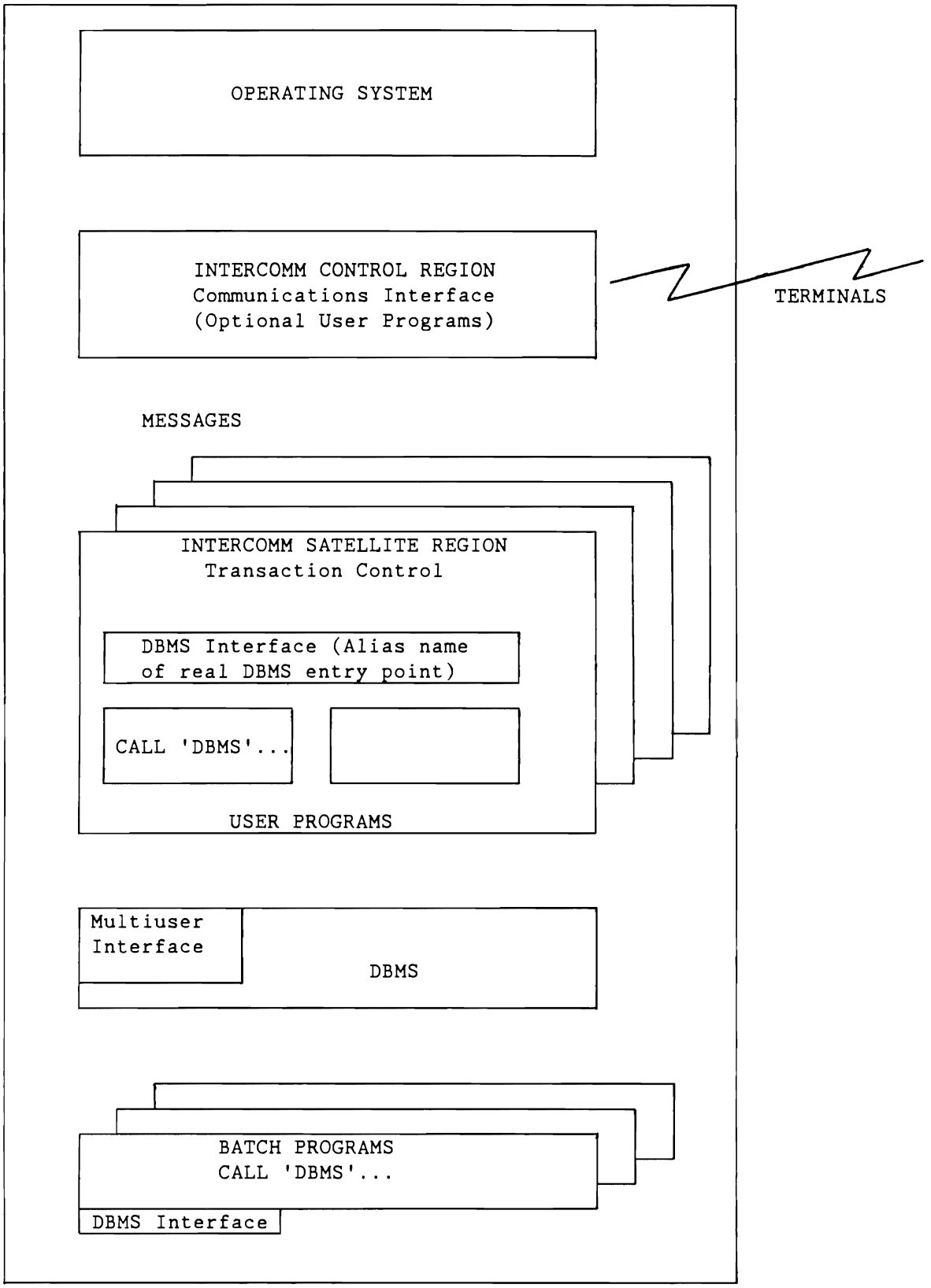


Figure 33. Intercomm and DBMS Environment (Multiregion Version of Intercomm)

In summary, two types of programs exist in the DBMS region when utilizing an Intercomm/DBMS facility:

- The data base access logic modules for performing the I/O acitivity to the data base(s) defined to the system
- The interface logic to user regions which:
  - -- Coordinates interregion communication
  - -- Provides special logic to ensure that concurrent DBMS request processing does not affect data integrity
  - -- Provides for checkpointing
  - -- Stacks requests when DBMS processing is single threaded

Similarly, two types of programs exist in the user regions:

- The user program logic requesting access of the data base (Intercomm application subsystems or standard batch jobs)
- The interface logic to the DBMS region, including interregion communication, and optionally, multithreading provisions and recovery provisions.

11.3 GENERALIZED DBMS INTERFACE FACILITY

The Intercomm Generalized Data Base Management System Interface (GDB) consists of a series of programs which allow data base access from multiple application subsystems, while providing data integrity across program and system failure. GDB includes all but specific data base access logic which must be provided by the user. Since programs not under Intercomm's control, i.e., batch programs, may require concurrent and/or overlapping use of the Data Base Management System, GDB also includes a provision for utilization of the DBMS by batch programs as well as on-line Intercomm programs.

In the situation where the specific data base logic is not supplied, all the control programs for DBMS interface and effectual use of the DBMS by on-line or batch programs are supplied by Intercomm. This includes startup, closedown, and restart/recovery procedures. In a sense, the Intercomm Generalized DBMS Interface can be said to be the data base management facility through which users are able to adapt Intercomm to the DBMS requirements of their own applications. GDB supplies a simple method of interfacing a DBMS with Intercomm while maintaining data base integrity with a minimum of programming effort. This is achieved by supplying the mechanisms to pass control to user-supplied routines at the following critical points in processing:

- At startup, restart, and closedown of the Intercomm region
- At initiation of a data base request
- At termination of a data base subsystem thread (normal and abnormal)

Message Restart, a standard Intercomm feature, is available for coordination of message restart with data base recovery.

11.4 CUSTOMIZED DBMS INTERFACES

Three of the Intercomm DBMS interface facilities developed for vendor-supplied DBMS are not based around the GDB framework. These are:

- DL/1 (IMS DB)
- TOTAL
- System 2000

11.4.1 DL/1 (IMS DB) Support

Intercomm and Data Language/1, the IBM-supplied DBMS, execute in separate and distinct partitions or regions. Users of the DL/1 facility may be on-line Intercomm subsystems residing in the Intercomm region or batch IMS application programs residing in separate regions. The single DL/1 region supports use of DL/1 from each user region, whether accessing the same or different data bases. Required DL/1 logic is all contained in the DL/1 region (exclusive control logic, data base logging logic, etc.). The only logic required in the on-line region is the Intercomm-supplied interface logic for communication of user DL/1 data base requests to the DL/1 region.

Intercomm is defined as an IMS batch program and uses the standard, supported IMS batch interfaces. Intercomm and its subsystems remain fully multithreaded with this interface. DL/1 CALLs, however, are serviced in a single threaded fashion. That is, only one DL/1 CALL is processed by IMS at a time. Multithreaded updates of different data bases can be accomplished by segregating subsystems that perform those updates into different Satellite Regions of the Multiregion Support (MRS) version of Intercomm. Multithreading inquiries against the same data bases can similarly be achieved. Restart/recovery is accomplished through standard IMS recovery procedures.

11.4.2 TOTAL Support

Support for Cincom Systems' TOTAL is provided to Intercomm users through the Intercomm/TOTAL Interface facility. Two modes of Intercomm and TOTAL operation exist. Intercomm and TOTAL may operate in the same region (or partition) as a main task and attached subtask, respectively. In the alternative mode of operation, Intercomm and TOTAL reside in two separate regions (partitions). In this case, TOTAL must be executing when Intercomm is brought up; all communication between the two tasks is initiated through the use of an interregion SVC, supplied by TOTAL. Coordinated Intercomm/TOTAL support provides data base integrity and affords restart/recovery procedures, including coordinated checkpointing.

Whether TOTAL is operational in the same region as Intercomm or in a separate region, the facilities of the TOTAL DBMS may be utilized by one or more batch regions. Off-line programs may access and update the on-line data base while Intercomm is up; data integrity will still be maintained through the use of the procedures provided. Intercomm's TOTAL support allows on-line and batch programs to run concurrently while accessing and updating the same data base.

11.4.3 System 2000

Support for System 2000 is provided for both the Natural Language Interface (NLI) and Procedural Language Interface (PLI). A vendor-supplied subsystem processes NLI terminal input and routes data base access requests to System 2000. User-coded application subsystems using PLI require a precompile function to incorporate the System 2000 interface requests.

System 2000 operates in a separate region from Intercomm. Restart/recovery provisions are included which allow all but coordinated DBMS and Intercomm checkpointing. Therefore all restart/recovery processing spans the beginning of on-line execution until system failure.

11.5 DBMS INTERFACES VIA GDB

Some Intercomm DBMS interfaces have been developed by DBMS vendors in conjunction with the Intercomm development staff. Such interfaces take advantage of the existing Generalized Data Base Management System Interface logical structure. DBMS supported in this fashion are:

- ADABAS
- IDMS
- Model 204

DBMS implemented via GDB or GDB entry points offer the user all the GDB facilities previously described in addition to the salient features of the individual DBMS mentioned in the following discussion.

11.5.1 ADABAS Support

ADABAS, a product of Software A.G., is a DBMS utilizing inverted file structure. The interface is comprised of a Software A.G.-supplied interface program. ADABAS operates in a separate partition or region and can be utilized by one or more batch regions while Intercomm is operational. All the functions of ADABAS are available to the Intercomm user. ADABAS in no way changes the standard Intercomm environment; the Intercomm implementation of ADABAS requires no modifications to the standard ADABAS CALL sequences. Standard CALL statements, as specified in the ADABAS Reference Manual published by Software A.G., are used for all data base activity against ADABAS files.

11.5.2 IDMS Support

The Integrated Database Management System (IDMS) is a software product marketed by the Cullinet Corporation. A comprehensive DB/DC environment is provided for IDMS via Cullinet-provided interface routines.

The Intercomm support elements allow IDMS to execute either in a region separate from Intercomm, or as a subtask of Intercomm. Full availability of IDMS facilities is provided. IDMS can be called from one or more batch regions while Intercomm is operational. Intercomm support does not modify IDMS requirements as specified in the IDMS Schema/Subschema Data Description Language Reference Guide. Support includes coordinated message restart and automatic data base recovery provisions.

11.5.3 Model 204 Support

Model 204 is a software product marketed by Computer Corporation of America (CCA). The DB/DC environment is provided via Intercomm- and CCA-provided interface routines. Model 204 is executed in a separate region from Intercomm and can be utilized by one or more batch regions concurrently with Intercomm. Model 204's standard calling sequences are not altered by Intercomm support. Interfacing between Intercomm and Model 204 consists of Intercomm's GDB supplemented by an Intercomm/Model 204 Interface.

Model 204 can access, retrieve and update records in inverted files stored in a data base. The user is assured the highest possible level of integrity and security. When Model 204 data base integrity is coordinated with Intercomm, the data base is fully recoverable from failure situations in which it was either physically or logically destroyed.

Chapter 12

STORE/FETCH FACILITY

12.1 GENERAL

Intercomm's Store/Fetch Facility provides an application program (subsystem) with the facilities to:

- STORE

  Save data either in main storage or on disk. Data is identified by a user-defined key which may be from 1 to 48 bytes.

- FETCH

  Retrieve stored data from main storage or disk.

- UNSTORE

  Free stored data from main storage or disk when the information is no longer needed by an application subsystem.

Stored data may consist of tables, counters, switches, messages, subsystem parameters, print lines or any information which a subsystem may wish to save and/or retrieve. Depending on the nature of the application, data may be defined as permanent (available until explicitly unstored), semipermanent (available at restart time, but scratched at normal startup), or transient (always scratched at normal startup or restart). Further detail may be found in Store/Fetch Facility.

12.2 MODULAR PROGRAMMING

Store/Fetch facilitates the use of modular programming techniques. In a modular environment, each subsystem typically performs one specific function. A group of related subsystems may work together to perform a major task, with each subsystem working on a specific portion of the task. Results can be communicated via the Store/Fetch facility. For example, an application subsystem is activated by an input message from a terminal, performs initial processing, and saves the input message and resultant data for the next related subsystem. The first subsystem makes some entries in a table, accesses files, sets switches, updates counters, etc., depending upon the content of the input message. A second subsystem then analyzes the output of the first subsystem and continues the processing to produce a second intermediate result. A third subsystem analyzes the output of the second subsystem and produces a final output message for the terminal.

Without Store/Fetch, this process would probably be implemented through the use of the File Handler to save intermediate results. The overhead for calls to File Handler service routines would then be included in the processing time of the application subsystems. Developing logic to interface with the File Handler to perform all of these functions would add to the implementation time and the storage requirements for each application subsystem. The Store/Fetch facility simplifies the development of modular applications.

Figure 34 illustrates the logic flow of a message which is processed in turn by Subsystems A, B and C through the use of the Store/Fetch facility. The subsystems are related and share switches, counters and tables saved and retrieved via Store/Fetch.

12.3 CONVERSATIONAL SUBSYSTEM APPLICATIONS

Subsystems involved in conversational processing may use Store/Fetch to preserve their conversational environment. Input messages and/or related data may be saved through the use of STORE and retrieved through the use of FETCH until the conversation is complete. Conversational processing using Store/Fetch is discussed in the Intercomm programmers guides (COBOL, PL/1, and Assembler Language).

12.4 OTHER ON-LINE APPLICATIONS

Subsystems which develop any type of data to be used by other subsystems can save and retrieve the data through Store/Fetch. The data may be transient, for example, cumulative totals for one Intercomm execution, and it might be necessary to preserve it only from Intercomm startup to closedown. The overhead of preserving the data on a data set can be eliminated by using Store/Fetch.

A subsystem may create an output message which it wishes to save for later transmission. There may be a certain set of conditions which must be met before the message should be transmitted. For example, a subsystem might expect to receive a feedback message from the Front End (via the Front End Control Message facility) to indicate one type of terminal output is complete before it would wish to transmit this message. Store/Fetch may be used to save a terminal-oriented message for later transmission.

12.5 THE MULTIREGION ENVIRONMENT

Store/Fetch routines are eligible for Link Pack Area residence. Thus, no additional storage overhead per region is encountered in a multiregion environment. All I/O associated with Store/Fetch is overlapped with all processing regions as well as within the particular Intercomm region.

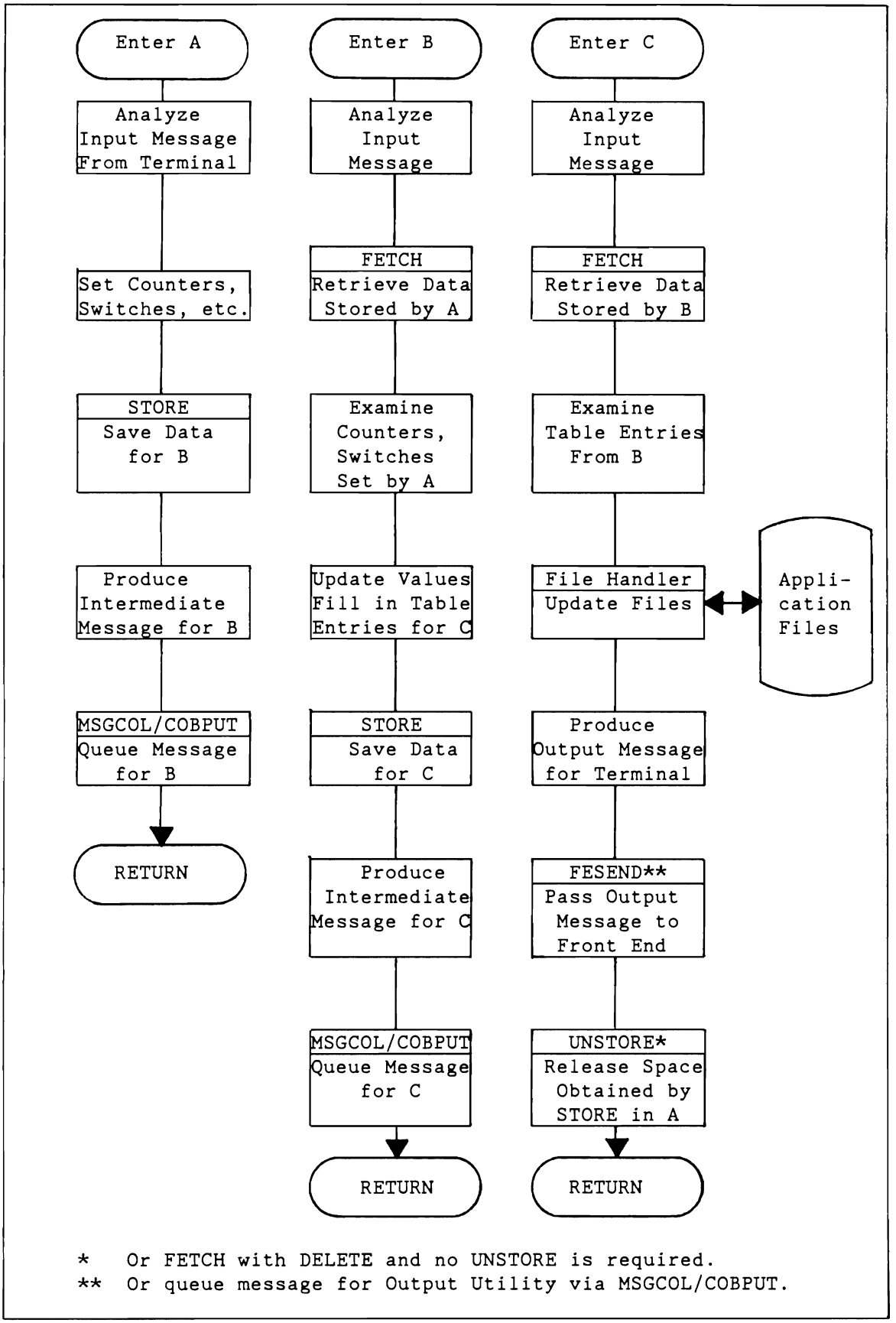
Store/Fetch data sets must be unique to each region. The multiregion concept of independent decentralized application-oriented regions is maintained.

12.6 BATCH MODE OPERATIONS

The Store/Fetch facility may also be used by batch programs to save and later retrieve data within the same or a different program. Off-line "utility" programs might create/update Store/Fetch data for on-line program access. Store/Fetch data can not be shared by Intercomm and batch application programs operating concurrently.

12.7 STORE/FETCH DATA SETS

Store/Fetch direct access storage requirement consists of up to ten formatted Keyed BDAM data sets. Store/Fetch employs internal data spanning if data strings exceed blocksize. Multiple data sets can be employed to keep blocksizes and string sizes consistent for efficiency.



* Or FETCH with DELETE and no UNSTORE is required.
** Or queue message for Output Utility via MSGCOL/COBPUT.

Figure 34. Using the Store/Fetch Facility

Chapter 13

DYNAMIC DATA QUEUING FACILITY

13.1 DATA QUEUES

The Dynamic Data Queuing (DDQ) facility (a Special Feature) provides application programs with the ability to dynamically create, retrieve, and delete logical data sets and/or queues of messages on a single BDAM data set. This eliminates the need to define separate data sets for small, transient groups of data records and/or messages. The term "queue" or "dynamic data queue" refers to any logical sequence of records, regardless of record size or content.

13.2 DDQ UTILIZATION

The DDQ facility provides on-line system designers with a powerful tool. Some typical uses of DDQ are:

- Segmented Messages

  Full multithreading of Intercomm segmented input and output messages to/from application subsystems can be achieved. Since each dynamic queue is unique to a message thread, no interleaving of unrelated message segments can occur; this eliminates the need to "lock out" a terminal until a subsystem has read/written the final segment of a multisegment message.

- Data Collection

  Application subsystems may desire to accumulate data for a period of time before processing that data, either on-line or in batch mode. They can do this by building a dynamic data queue and adding to the queue when necessary.

- Message Collection

  Application subsystems may wish to delay processing or outputting of messages until a time period has elapsed or a certain event has occurred.

- Data Switching

  If large amounts of data are to be passed from one subsystem to another, the normal Intercomm message switching scheme may be unwieldy. DDQ enables the user to pass the data by building a queue of records and then forwarding the queue name in a message to the destination subsystem.

- Batch/On-line Communications

  Queues created by an on-line program can be accessed via a batch program and vice versa.

13.3 DDQ FEATURES

The DDQ facility provides the following features:

- Dynamic queues may be created for use on-line with Intercomm or in batch mode. Records written on dynamic queues may be of any size up to the 32K hardware limitation of System/370.

- Either message or non-message data may be contained in queue records.

- Queues may be created, updated, added to, read, and deleted via DDQ service routines requested via application program logic.

- Queues may optionally be "saved" for any period of time. Queues that are saved can be retrieved at any time via user-specified queue identifiers.

- Queues may be created on one or more BDAM data sets. If multiple BDAM data sets are available, the user can optionally force the creation of a queue on a particular data set, or use a default data set.

- Records on queues may be blocked or unblocked. Automatic blocking/deblocking of blocked records is provided.

- Automatic handling of variable-length records is provided, including handling of records larger than the physical blocksize of the data set. This is true for both blocked and unblocked queues.

- Queues may be shared between on-line and batch programs.

- When used with Intercomm, the space allocated to a queue for a subsystem message processing thread is optionally recovered after subsystem failure, via the Intercomm Resource Management audit and purge facility.

- All queues are optionally preserved if Intercomm is restarted using the standard Intercomm Restart facility.

- Queues may be specified as single-retrieval or multiple-retrieval. Multiple-retrieval queues can be read nondestructively as many times as desired.

- If a subsystem uses MMU to format output messages (such as a printer report) to be put on a DDQ, the call to MAPEND (after all output is created) may request that the output messages be placed on a DDQ rather than being queued as multiple messages for the terminal, thus relieving main storage space.

- A DDQ containing output messages may be passed to a terminal via a Front End Control Message (FECM) which contains only the queue identifier. MAPEND automatically generates the FECM for a DDQ output request.

DDQ is further described in Dynamic Data Queuing Facility.

Chapter 14

PAGE BROWSING FACILTIY

14.1 PAGE BROWSING USE

The Intercomm Page Browsing Special Feature (PAGE) is a service program which allows a terminal operator at a display (CRT) terminal to browse multipage reports. This is accomplished via entry of any of the preprogrammed PAGE commands which cause the display of the first page, last page, next page, a specific numbered page, etc. The pages are displayed from a BDAM data set called the Page Data Set. The report through which the operator may browse consists of application-generated messages which fill more than one CRT screen. The application program which generates the messages does not participate in what appears to the terminal operator to be a conversation. The application program generates the output messages (report) in response to an input message received from the CRT, writes them to the Page Data Set, and then returns control to Intercomm. The Page Facility returns the first message to the terminal operator as response to the input message, no message is directly returned by the subsystem. If MMU is used to format the output messages, the program may request MAPEND to store the messages on the Page Data Set, rather than calling Page. If the Output Utility is used, the PAGE command processing subsystem will pass the output message to the Output Utility for formatting before transmission to the terminal, however the subsystem must directly pass the prepared messages to Page.

In addition to its browse capability, PAGE includes a command for saving a report for later reference. This capability enables the operator to request additional detail regarding a particular part of a report without losing the entire report. For example, the operator might have entered an input message requesting the name and employee-ID of all employees with more than 20 years of service with the company. The output report might contain hundreds of names and span numerous CRT pages. The operator might note a particular name and ID from page 2 of the report, input the command to save the report, input another message requesting a display of the employee's entire record, verify additional details from the employee's record, input another PAGE command to view page 2 of the report again, and continue until all relevant information is gathered.

PAGE increases programmer productivity since each programmer does not have to develop the logic for multipage report processing. PAGE also contributes to Intercomm efficiency since as a multithreaded subsystem, it will concurrently process multiple unrelated page requests from multiple operators. Without PAGE, this might require multiple conversational application programs, each of which would occupy storage in order to respond to operator requests for information. Furthur details are provided in Page Facility.

14.2 PAGE BROWSING OPERATION

Figure 35 illustrates the message flow using the Page Browsing facility in the Intercomm system. The numbers correspond to the numbers in the illustration:

1. The operator enters an inquiry (internally identified, for example, by message No. 501).

2. The appropriate user subsystem is initiated; it processes the message and

3. passes the output message response(s) for No. 501 to the Page Control routine.

4. The Page Browsing facility saves the responses on the Page Data Set,

5. enters the necessary control information in the Page Table, that is, message number, address of first message on page queue, and number of pages, then

6. passes the first page of output response No. 501 to the Front End for eventual transmission to the terminal operator.

The terminal operator examines the first page of response 501, then, by entering different PAGE commands, can effectively browse through the report on the Page Data Set.

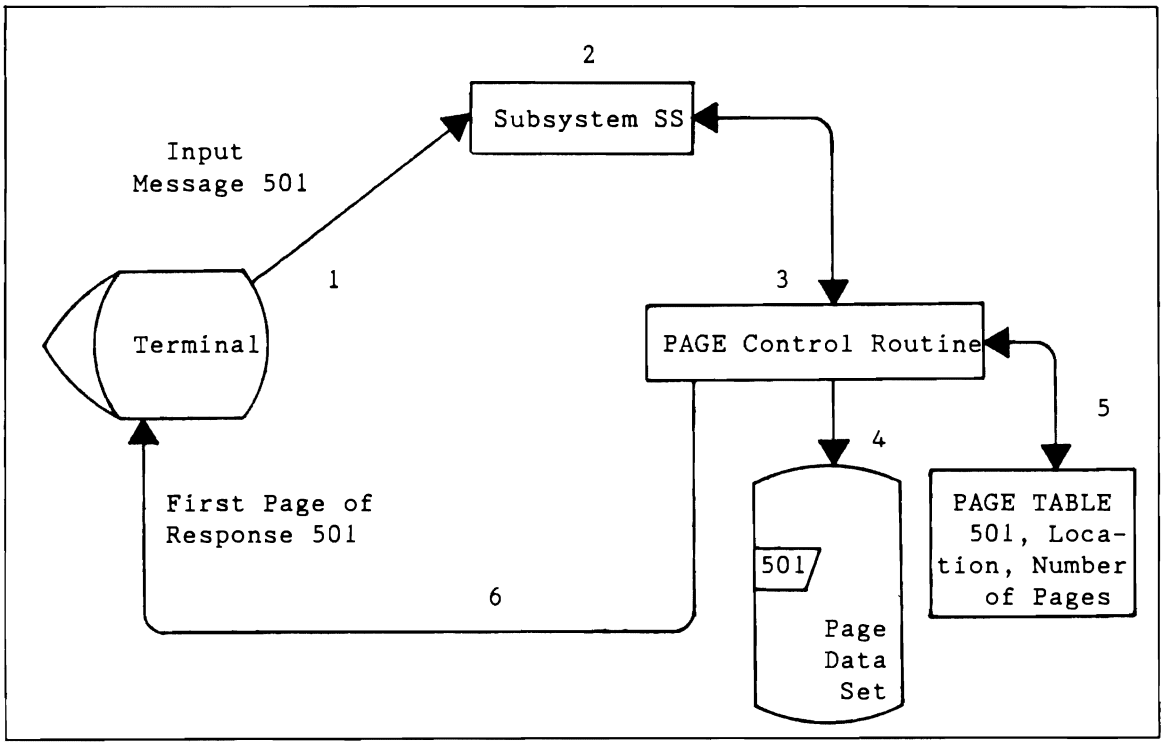


Figure 35. Message Flow Using Intercomm Page Browsing Facility

Figure 36 illustrates the message flow when the terminal operator issues a PAGE command. The numbers below refer to the numbers in the illustration:

1. The terminal operator enters a PAGE command, e.g., to get the next page of the report, which is passed directly to the Intercomm Page subsystem (PAGEMSG).

2. PAGEMSG retrieves the requested page from the Page Data Set and

3. passes the message to the Front End for eventual transmission to the terminal.

If the operator completes viewing response 501, but requires it for future reference, the SAVE command can be entered to save the report on the data set. Otherwise, when the Page subsystem recognizes a new message number, it assumes the previous report is no longer required and deletes it from the Page Table.

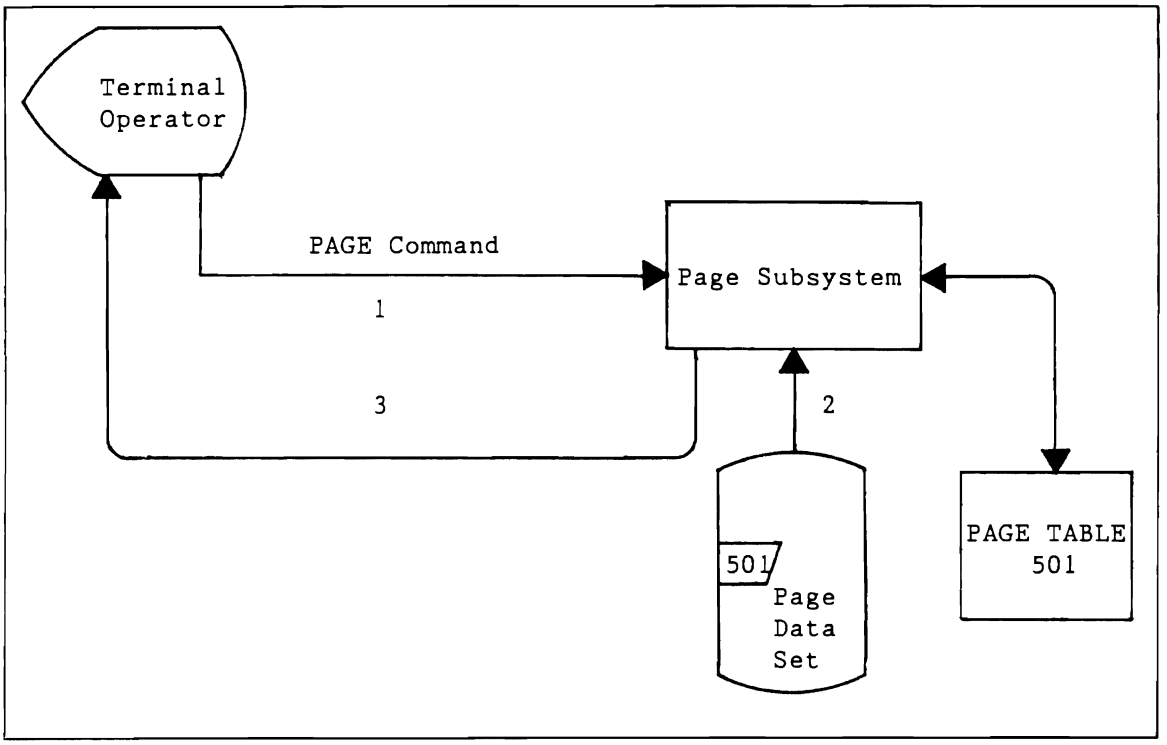


Figure 36. Terminal Operator/PAGE Communication

Chapter 15

MULTIREGION SUPPORT FACILITY (MRS)

15.1 MULTIREGION CONCEPTS

The Intercomm Multiregion Support facility (MRS), a Special Feature, allows groups of application subsystems to execute in separate OS/VS regions or partitions. One region is designated as the Control Region; the others as Satellite Regions. (See Figure 37).

The Control Region consists of a complete Intercomm system and, optionally, application subsystems. The primary function of the Control Region is to handle all terminal and interregion message traffic, that is, it controls the routing of messages to and from the terminals and the Satellite Regions. Additionally, the Control Region can accept messages from batch application programs.

Satellite Regions consist of application subsystems and an Intercomm Back End; they communicate only with the Control Region, never with each other. Subsystems which conform to standard Intercomm coding conventions can execute unmodified in either the Satellite or Control Regions.

An optional feature of the Multiregion Support facility is the ability to service a Satellite Region's logging requests in the Control Region and to have its log records written to the Control Region's log data set. This is called the Single Log feature. Intercomm Message Restart does not support the Multiregion Single Log feature; thus, if the Restart/Recovery facility is required in a particular Satellite Region, that region must have its own log data set.

15.2 MULTIREGION FEATURES

The salient features of Intercomm Multiregion Support include:

- Terminal I/O centralized in one region
- High subsystem reliability, due to separation of applications
- Lower storage requirement than running multiple copies of Intercomm
- Ability to start/stop any region when desired
- Separate logging in each region. Optionally, all logging may be to a single INTERLOG data set through the Control Region. (Restart capabilities are lost with this option.)

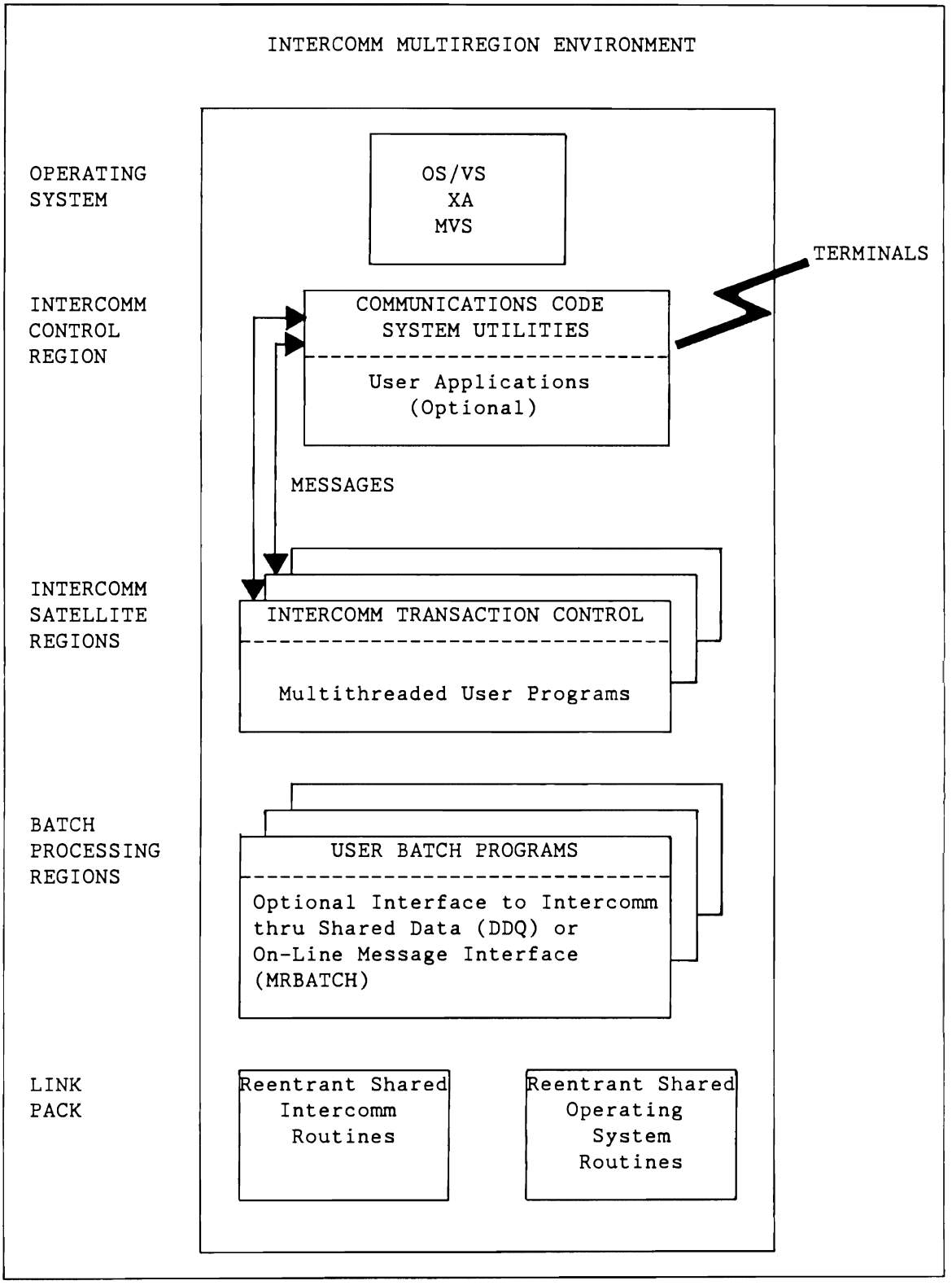


Figure 37. Intercomm Multiregion Environment

- No currently executing subsystem need be modified to run in a multiregion environment.
- Automatic handling of region ABENDs
- Messages to a subsystem in a particular region may, if the region is inactive, be flushed, queued, or sent to an alternate region.
- Batch programs may communicate with the on-line environment by sending messages to an Intercomm subsystem.
- Security can be implemented via table-oriented Region Associated Processing (RAP) which restricts terminals to the use of transactions associated with specific regions.

Additional performance efficiency is realized in a VS environment because page faults in an Intercomm region are overlapped with page faults in other Intercomm regions.

Implementation, features, and control processing are described in Multiregion Support Facility.

Chapter 16

MODEL SYSTEM GENERATOR

16.1 SYSTEM PERFORMANCE QUESTIONS

Users are constantly asked the following types of questions about the Intercomm system:

- "What kind of response time will I get?"
- "What are the throughput capabilities of the system?"
- "How much storage must be allocated?"
- "Should I increase my channel capacity?"
- "How much overhead does Intercomm take?"
- "How much disk space is required?"
- "How much real storage will be required under VS?"

The answer to the above and many similar questions is always "installation-dependent."

After numerous studies of different Intercomm users' systems, the following variances among the systems have been found:

- Region size ranges from 600K to over 6 meg.
- Response time averages range from less than one second to greater than 5 seconds.
- CPU utilization ranges from less than 5% to greater than 50%.
- Throughput capacities vary from several thousand an hour to over 200,000 an hour.

These variations occur as a result of many installation-dependent factors including concurrent batch requirements, CPU size, channel capacity and separation, network configuration, etc. However, system performance is dependent mostly on the resource demands of the application programs. This factor is totally installation-dependent and subject to wide variations.

16.2 SYSTEM PERFORMANCE MODELING

The Model System Generator (MSG), a Special Feature, is designed to answer questions on resource requirements, throughput capabilities and response time estimates. MSG creates an individualized model for one or more Intercomm application subsystems. MSG models the key elements in the user's Intercomm system, i.e., application software and message flow. Thus, it is possible, when using MSG, to accurately answer all of the above questions about a future system long before any application programs have been written.

MSG is beneficial to the user who contemplates new Intercomm applications. MSG can operate in conjunction with an existing "live" system and thus, not only model requirements and capabilities of the new system, but also accurately measure the impact of the existing system.

MSG is described in Model System Generator.

Statistics on resource utiltizations and service requests for existing applications may also be gathered via SAM (System Accounting and Measurement) as described in the Operating Reference Manual.

Chapter 17

DATA ENTRY FACILITY

17.1 DATA ENTRY OPERATION

The Intercomm Data Entry System Special Feature provides general purpose data entry and verification capabilities to facilitate keypunch-like operations from IBM 3270 local and remote terminals. Data Entry is designed to provide equivalent capabilities to the IBM Program Product Video/370.

Data Entry is a reentrant Assembler Language subsystem that executes under Intercomm. It requires approximately 24K plus 776 bytes of storage per active Entry mode operator. An off-line extract program is provided to remove data after entry.

Data Entry uses the screen formatting capabilities of the IBM 3270 to provide fill-in-the-blank prompting templates for data submission. Data is maintained in one or more centralized files, according to document type and batch number attributes which are specified at entry time. Operators can ENTER, VERIFY, CORRECT and SCAN data.

17.2 DATA ENTRY IMPLEMENTATION

Data Entry applications can be quickly implemented, since only table entries are required from the user. User exits are provided and user exit code may take advantage of all Intercomm capabilities.

Data Entry is described in the Data Entry Installation Guide and the Data Entry Terminal Operators Guide.

Chapter 18

AUTOGEN

18.1 INTRODUCTION

The Autogen facility (a Special Feature) is an extension to the Message Mapping Utilities (MMU). Autogen facilitates screen format specification from an IBM 3270 CRT. The user enters field-oriented data according to prompting screens supplied by Autogen. MMU map definitions are generated as a result of the terminal session.

18.2 MAP DEFINITIONS WITH AUTOGEN

The Autogen Special Feature provides a direct method of creating Map Definitions, speeding implementation of on-line applications by specification of terminal screen layout through the terminal itself. Autogen operates as an Intercomm application program (subsystem).

Using Autogen, the application programmer enters a model of the screen layout desired, then submits this model to the Autogen facility. Autogen, through a series of prompting screens, requests additional information to complete the screen specifications and produces the Assembler Language macros required for an MMU map definition.

After the general screen layout is defined, that layout is displayed to the user for detailed specification of names of variable fields and field attributes. This process completes the definition of one screen. The user may then continue to define additional screens, or terminate the session.

A correction facility exists within Autogen which enables the user to return to a previous point within the prompting sequence for a map definition in order to change previously specified items. This correction facility is referred to as Revise Mode.

18.3 AUTOGEN CAPABILITIES

Autogen has been designed to provide automatic specification for a subset of MMU capabilities; that subset has been chosen to represent the majority of anticipated MMU uses. Autogen is designed for simplicity and ease of use.

MMU provides an extensive array of features that offer a diversity of complex formatting options, many of which produce an output format whose characteristics vary dynamically according to data content. In such complex formatting situations, direct assembler macro specification may still be required.

Some of the advantages of the Autogen Special Feature are:

- Proposed screen layouts can be analyzed and critiqued at the terminal to quickly correct design flaws such as clutter or awkward field positioning.
- Proposed screen layouts can be created by analysts or programmers in conjunction with prospective system users to obtain immediate user reaction and feedback.
- Programmer requirement to understand MMU macro coding and other training requirements are decreased.
- Programmer productivity is increased; programmer error rates can be decreased.

Autogen uses existing system facilities and requires negligible system overhead. Depending on installation requirements, it can be defined as a high-priority resident subsystem or a lower priority dynamically loaded (or overlay region) subsystem.

Autogen supports the IBM 3270 Model 2 CRTs (1920-character screen) or hardware compatible devices, and Dataspeed 40 terminals.

Autogen implementation and use are furthur described in Autogen Facility.

Chapter 19

DYNAMIC FILE ALLOCATION

19.1 INTRODUCTION

The Dynamic File Allocation (DFA) Special Feature provides a means by which program threads utilizing the File Handler (either Intercomm subsystems or batch programs) may access already existing data sets or create new data sets without each data set being explicitly defined via JCL. It is only required that a DD card be present in the Intercomm execution JCL which defines the disk pack containing (or, to contain) the data set. Only sequential data sets residing on disk are supported as described in Dynamic File Allocation. DFA functions are provided by two File Handler service routines: ALLOCATE and ACCESS.

19.2 ALLOCATE SERVICE ROUTINE

The ALLOCATE service routine allows application programs to create sequential data sets in any format (fixed, variable or undefined) on any disk pack for which a DD card is present. Dynamically allocated data sets always have an implied disposition of NEW, in which case calls may subsequently be made only to WRITE or PUT; calls to GET or READ are invalid. As an option, dynamically allocated data sets may be cataloged at the time they are created.

Dynamically allocated data sets are spun-off for immediate use by other programs, such as batch print programs, or they could be used as input to assemblers or compilers; that is, as data sets are created using ALLOCATE, they are normal data sets, no longer hooked to Intercomm, and can be used as any data set can be used. This fact requires some caution in analyzing the programs that use dynamically allocated or accessed data sets because they are not subject to the normal exclusive control provided by the operating system.

19.3 ACCESS SERVICE ROUTINE

ACCESS provides the ability to access existing data sets without preplanning. It allows, for instance, a subsystem to retrieve data from any number of sequential data sets upon request by a terminal operator. Only the data set name (DSN) must be known. These data sets need not have been created prior to the execution of Intercomm; they must be created at any time prior to being accessed. In addition to being accessed, these data sets may be extended or updated; that is, dynamically accessed data sets may have a disposition of OLD, which allows retrieval or updating, or a disposition of MOD, which allows extension of the data set.

Chapter 20

SNA TERMINAL SUPPORT

20.1 ELEMENTS OF A SYSTEM USING SNA TERMINALS WITH INTERCOMM

The newest IBM telecommunication system organization is called Systems Network Architecture (SNA). Intercomm's support for this comprehensive environment is a Special Feature.

SNA systems have a nodal structure with processing capabilities distributed among the nodes. Connections between the nodes may be changed without individual node involvement; network resources may be shared by many nodes. SNA defines the protocols used to communicate between nodes and the structure of the processing system of the nodes: transmission layer, function management layer, and application layer.

The implementation of SNA uses the Virtual Storage Telecommunications Access Method (VTAM) in the host computer node (System/370 with VS). Remote telecommunication lines are serviced by the Network Control Program (NCP) executing in a 37**xx** Communications Controller. Some terminal nodes may have processing capabilities in a SNA Cluster Controller, which executes SNA Controller application programs. One or more VTAM application programs are in the host computer. Intercomm is one VTAM application program. It controls the execution of application subsystems which process transactions originating from SNA terminals.

Figure 38 summarizes these system elements.

20.2 INTERCOMM SUBSYSTEMS AND LOGICAL UNITS

The only nodes of a SNA network of interest when designing applications are:

- Intercomm (a VTAM application program) and user-written application subsystems executing under Intercomm control.

- Logical Units (LU)--addressable units of logic in a remote SDLC SNA Controller, or remote bisync and local 3270 terminals. An SDLC LU is an executing controller application program communicating with VTAM application programs (i.e., Intercomm) using resources of the SNA Cluster Controller--storage, processor cycles and external terminals connected to the controller.

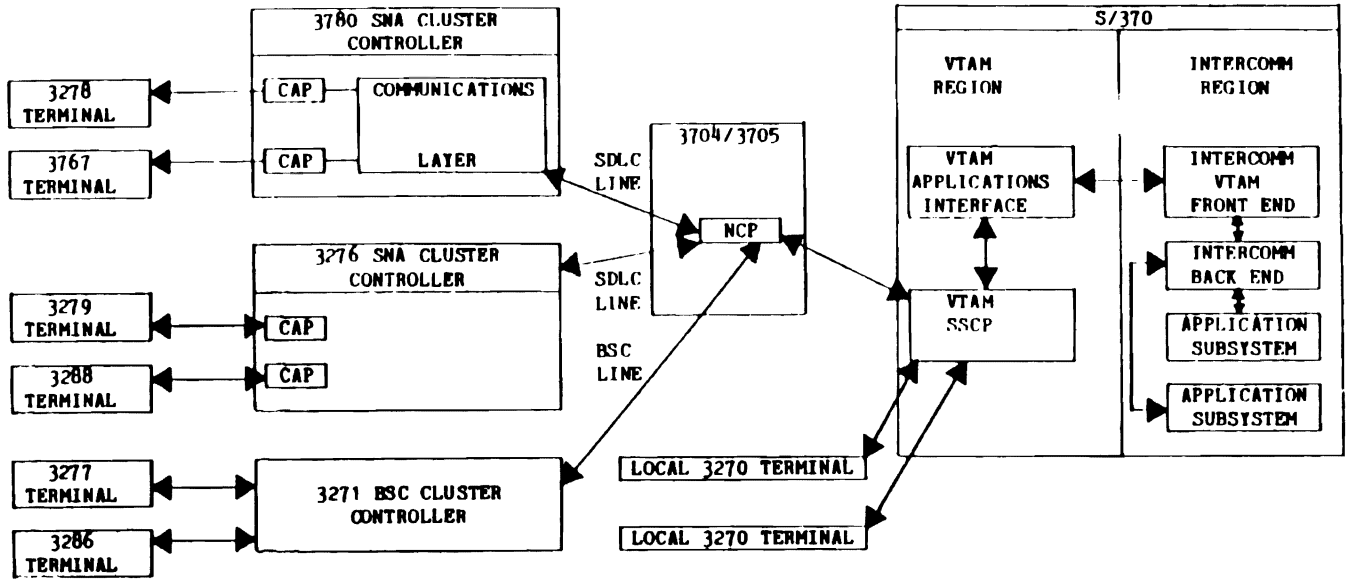


Figure 38. Elements of a Communications System Using VTAM with Intercomm

TERMINALS: Device to communicate with outside world, for example, CRT and Keyboard, Printer.
Examples: 3604, 3618 (3600 System)
3277, 3767 (3790 System)
327x, 328x (BSC/SNA 3270 System)

SNA CLUSTER CONTROLLER: Special purpose small computer that executes user-written application programs that communicate with terminals, perform local processing of data, and communicate with application programs in S/370. May also be microcoded, having no user code (for example, 3276)
Examples: 3601, 3791, 3270 system.

3704/3705: Telecommunications computer that executes the Network Control Program (NCP) to manage lines between SNA controllers and VTAM.

S/370: Host computer executing VS1/MVS with regions:

VTAM Region: Virtual Telecommunications Access Method (VTAM) allocates network resources and routes data through network and to application systems.

Intercomm Region: Intercomm application system utilizes system routines to schedule Intercomm subsystems when input is received from SNA controller programs and sends output from these subsystems back to the SNA controller programs. User-written Intercomm subsystems process the input, access the data base, and create the output.

To an Intercomm subsystem, a logical unit appears as one or more logical unit components. Each component is assigned a unique Intercomm terminal identifier. The subsystem will receive messages from components, will process them, and will create reply output messages, usually directed back to the originating component. In general, the coding requirements for an Intercomm subsystem communicating with logical units are independent of VTAM or SNA considerations.

Subsystems do not need to be changed when converting from BTAM or TCAM to a VTAM Intercomm Front End. Logical units send messages to Intercomm subsystems based on transaction codes in the input transactions.

20.3 SNA TERMINALS SUPPORTED BY INTERCOMM

SNA terminal support is described as a list of VTAM/SNA facilities supported by Intercomm. This method of description is possible because of the uniformity of SNA communication protocol. Any programmable SNA terminal that uses only this subset of VTAM/SNA facilities is supportable by Intercomm. The IBM 3270 Display System (SDLC, remote bisync and local) and the 3600 and 3790 systems are currently supported. Packet networks which conform to SNA protocol may be treated in Intercomm as 3270 LUs. The IBM Network Terminal Option (NTO) special feature is not supported. Users may employ protocol converters (to 3270 protocol) to support remote switched (dial) devices.

The VTAM/SNA facilities in Intercomm are:

- Connection may be initiated by the logical unit or by Intercomm. Session parameters are not checked by Intercomm; Intercomm is controlled by its own tables describing the logical unit. Bindareas and Logmode tables may be defined via the network table. Automatic reconnect after a user-specified interval is supported.

- Orderly shutdown or immediate disconnection of a logical unit can be initiated by the logical unit or by Intercomm.

- Data messages from the logical unit to Intercomm may be single segment or chained and may request definite, exception, or no response protocol.

- Data messages from Intercomm to the logical unit may be single segment or chained, and responses may be requested (forced for 3270-system printers), according to table definitions.

- Message sequence numbers may optionally be reset to zero on connection, state error, or request for recovery (RQR) command from the logical unit. The Set and Test Sequence Number (STSN) command is not sent by Intercomm. BTAM input messages sequencing may be requested for VTAM LUs (required for message restart).

- Quiesce by Intercomm of the logical unit may be requested in order to deactivate the LU in Intercomm and refuse new logons.

- Logical units may send the Signal command. Intercomm invokes a user exit routine to act upon the Signal command.

Any VTAM/SNA features not listed above are not supported.

20.4 INTERCOMM FRONT END FACILITIES SUPPORTED BY THE VTAM FRONT END

Where appropriate, facilities of the BTAM Front End are implemented in the VTAM Front End. A BTAM Front End may coexist with VTAM to support non-VTAM devices. New facilites unique to VTAM have also been implemented. The existing and new facilities include:

- Verbs in the input data message text are delimited by the system separator character or end of the message; short verbs are allowed. (The system separator character is defined via the Intercomm SPALIST macro in the SPA table.)

- A logical unit (component) may be locked to a verb via table coding or the LOCK system control command, and to a Multiregion satellite region via table coding or the LOKR command.

- All transaction code definition options are supported with the same meaning as for BTAM input.

- System control commands may be entered from a logical unit for BTAM or VTAM terminals, including current status display or a disconnect (Logoff) request for the subject terminal.

- Fast message switch messages may be sent from a logical unit to any VTAM component or BTAM terminal.

- Input chains from a logical unit may be either queued for a subsystem as individual segments or accumulated into a single message to be sent to a subsystem.

- Each logical unit component has its own dedicated output queue, which may have main storage, disk and priority queue specifications.

- Only full messages may be sent to components by a subsystem. Segmentation into output message chains is done within the Front End based on maximum segment size or by a user exit routine. Front End Control Messages may be used.

- A component may be defined as a CRT to obtain one output message per input message processing logic. Conversational terminal processing for CRTs is also supported.

- Standard Intercomm message recovery is provided when the message is scheduled for output by VTAM, that is, when the message is on the LU output queue.

- With Release 9.0, the Intercomm control terminal may be any LU. It may also be the system console. In the event of VTAM problems, the operator may continue to communicate with Intercomm via the console. Global WTO routing is also available.

- 3270 CRT copy processing is supported from the requesting terminal only and may be to a BTAM or VTAM 3270 device.

- 3270 CRT AID Key processing may be requested.

- The Intercomm VTAM interface may be shutdown and restarted by command or restarted automatically after an elapsed time interval.

- Support is provided to automatically share printers between Intercomms or with other TP applications.

- Basic Security processing (see Chapter 6) and ESS are supported for VTAM LUs.

Facilities of the BTAM Front End not listed above are not supported. Refer to the BTAM Terminal Support Guide and System Control Commands for additional description of the above facilities and commands. Details on VTAM support, implementation and control are provided in the SNA Terminal Support Guide, along with descriptions of optional user exits for further control of the network and message processing.