INTERCOMM

D B M S
USERS GUIDE

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

LICENSE: INTERCOMM TELEPROCESSING MONITOR

SPR 232 12/88

Data Base Management System Users Guide

Publishing History

Publication	Date	Remarks
First Edition	December 1976	This manual corresponds to Intercomm Release 7.0.
IPN 112	April 1977	General revisions and updates.
IPN 127	June 1977	General revisions and updates.
2nd Printing	July 1977	Incorporating IPNs 112 and 127.
SPR 173	June 1980	Revisions, corresponding to Intercomm Release 8.0.
3rd Printing	August 1980	Incorporating SPR 173.
SPR 188	December 1980	Revisions, particularly to Chapter 9, "Installing IDMS Support."
4th Printing	December 1980	Incorporating SPR 188.
SPR 214	June 1983	Preliminary documentation of support for TOTAL Release 8 under Intercomm Release 9 (revisions to Chapter 7).
SPR 232	December 1988	Revisions to Chapter 7, "Installing TOTAL Support" containing Intercomm Release 9 updates and Intercomm Release 10 additions (Automated Restart).

PREFACE

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

Intercomm Data Base Management System Support is offered as a Special Feature to the basic Intercomm system. It provides both on-line and batch application programs with the ability to access data via the facilities of a particular Data Base Management System (DBMS), and includes DBMS coordinated checkpoint/restart capabilities to maintain data integrity in the event of system failure. The following DBMS are supported:

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

In addition, a Generalized Data Base (GDB) Interface facility is available to which user routines may be added to support a user-coded DBMS or a proprietary DBMS not directly supported by Intercomm. Each DBMS Interface Support is a separate Special Feature.

DBMS Support may be limited to certain operating systems. Consult the DBMS Vendor for possible restrictions.

Multiple DBMS can be concurrently supported by a single Intercomm if the Intercomm Multiregion Special Feature is utilized.

This document describes the DBMS Interface facility and its utilization from the application program point of view, and details implementation techniques for the Intercomm System Manager.

It is assumed the reader is familiar with the DBMS in use and the associated documentation supplied by the vendor. Therefore, only DBMS Interface specifications are given. The reader is referred to the following list of Intercomm publications in conjunction with the use of this document.

INTERCOMM PUBLICATIONS

GENERAL INFORMATION MANUALS

- Concepts and Facilities
- Planning Guide

APPLICATION PROGRAMMERS MANUALS

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

SYSTEM PROGRAMMERS MANUALS

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

CUSTOMER INFORMATION MANUALS

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

FEATURE IMPLEMENTATION MANUALS

- 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

EXTERNAL FEATURES MANUALS

- SNA LU6.2 Support Guide

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

1.1 DBMS INTERFACE OVERVIEW

In recent years, data processing has undergone a major technological advancement through the general availability of Data Base Management Systems (DBMS). These systems now provide a broad array of services and techniques for the management of file data items. The resultant benefits are a simplification of programming requirements, a high level of data structure independence for the programs, reliability maintenance schemes and the capability to readily find, interconnect and associate widely divergent items--in an encompassing set of data items. Concomitant with the advancement of DBMS technology has been the need for communications software to permit the efficient and effective on-line utilization of data base services. This on-line facility must also provide for an integration of communication and transaction-oriented recovery techniques with reliability. The INTERCOMM DBMS interfaces provide a powerful, efficient and complete DB/DC environment that permits high volume access, rapid response time, a broad array of utility services, simplified application programming and an integrated restart/recovery.

Many software components are required to effect the operation of support. Support varies according to the specific DBMS as to which, if any, components are supplied by Informatics Inc. and which, if any, components are supplied by the DBMS vendor. The user is advised to consult the DBMS vendor as to separate charges that may be applicable for the supplied DBMS interface.

INTERCOMM Data Base Management System (DBMS) interfaces are provided as an extension to the inherent capabilities of the INTERCOMM on-line system. Four different categories of DBMS support exist for the INTERCOMM system:

- Generalized Data Base Management System (GDB) Interface--INTERCOMM and user-supplied routines to interface to a user-written (or nonsupported) DBMS.

- Customized Data Base Management System Interface--specially developed support for a specific DBMS. This category applies to DL/I support (no vendor routines required) and TOTAL support (vendor routines required).

- Data Base Management System Interface via GDB--vendor developed support based around the INTERCOMM GDB support. This category applies to support for ADABAS, IDMS and Model 204. The DBMS vendor and/or INTERCOMM supply all interface routines.

- Data Base Management System Interface provided totally by the DBMS Vendor. System 2000 support is in this category.

Figure 1-1 illustrates these categories of support. INTERCOMM DBMS interface facilities are each a Special Feature to the INTERCOMM system, available as separately contracted software products.

The specific techniques used to accomplish the DB/DC environment vary according to the specific DBMS. However, all the techniques provide equivalent capabilities. Generally, these various interfaces can be visualized as consisting of six areas of support:

- On-line initiation--where a communication path between INTERCOMM and the DBMS is established, requisite signing on procedures are performed. Contingency plans are executed if the communication path cannot be established.

- On-line termination--where an orderly disassociation is made of the DB/DC environment.

- Command processing--where requests for DB services made by application programs are executed.

- Contingency processing--where a program is executed in the event of a failure situation in INTERCOMM, or of its on-line applications (subsystems) and/or the DBMS software.

- Checkpoint--where a data base/transaction execution quiesce and equilibrium point is periodically established to provide for a common backup point in the event of catastrophic failure.

- Recovery--where a reconstruction of a valid transaction and DB environment is accomplished.

Before presentation of the elements of INTERCOMM DBMS support, the background against which data base management became an integral part of computer operations is briefly given here. The need for functional simplicity in interfacing a system with a multiplicity of DBMS types is reflected in INTERCOMM's

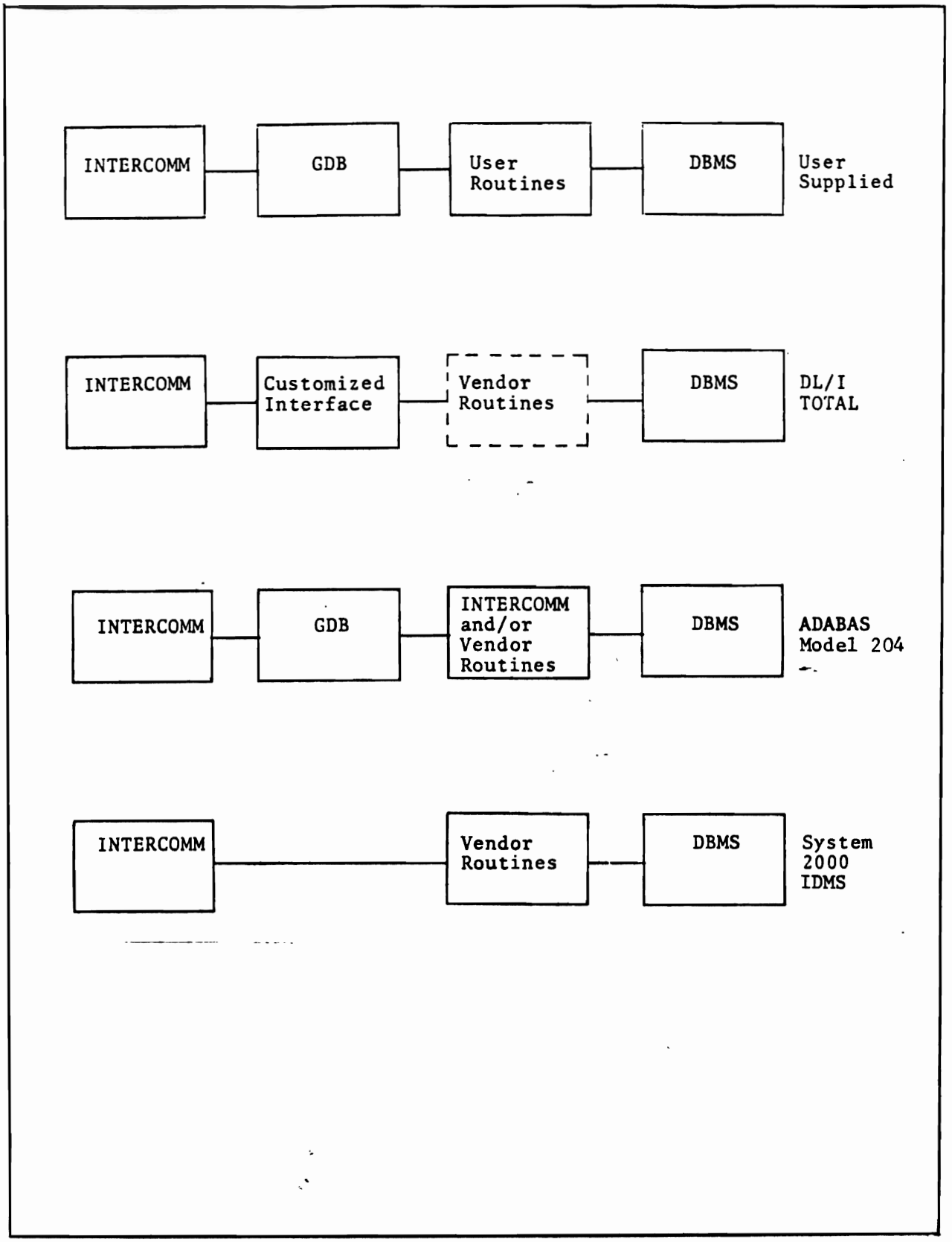

Figure 1-1. Categories of Interface Support

interface support capabilities. Therefore, we have provided a framework of interface concepts in Sections 2 and 3. Subsequent sections of this document further examine these implementation requirements, collectively where conceptual similarity of support elements exists, and then separately for implementation of INTERCOMM support for individual DBMS.

For details on the Data Base Management Systems themselves, consult the appropriate vendor's reference publications.

1.2 EVOLUTION OF DATA BASE MANAGEMENT SYSTEMS

At one time in the development of computer systems, data base management was nonexistent. Computer operations and user requirements had not yet mutually grown so enormous and multifaceted as to make centralization of data processing operations expedient for all users; i.e., a systems approach to all functional aspects. At best, programmers and systems people had to maintain control over their own data records and data files. The data was usually accessed on an application-by-application basis. Data was therefore application-bound. Such individual applications might have been inventory control, payroll, manufacturing, planning, etc. Since a data structure was designed specifically for an application, other applications had to either alter their approach to fit existing data structures or develop their own structures with similar data but disparate format and/or access techniques. Thus, the application program logic was subject to the organization of a particular data record(s) and file(s) or else required reorganization and programs were generally confined to usage of that data. Maintaining and updating data was, therefore, no small task.

Early data management was curtailed and generally oriented to one application principally because access methods themselves were as yet unsophisticated and based on single key access to the data. Data management in terms of file management and disk file organization did not provide the level of data supervision and application program independence needed as it was concerned only with the management and control of data on a file-by-file basis. (This is data management vs. data base management.)

The outcome of such a state of affairs was redundant and wasteful allocation of computer and programming resources; there existed repetitive data, inflexible storage capacity, a waste of storage space, needless use of computer time and high application programmer expenditures in maintenance and conversion procedures.

The lack of centralized data base informaton also limited the efficiency of on-line systems. To serve multiple applications in a real-time environment generally required a multiplicity of DASD files be mounted for prolonged time periods, a costly concern. With the emergence of the data base and the data base management concept, many costly and inefficient methods are now bypassed, such as redundant information in files, differing formats of each application file, updating difficulties, tailoring of data to specific physical devices and the change necessitated in application programs with the advent of any new data management techniques or devices. Today's data base can, therefore, be defined as a nonredundant assembly of interrelated data items which may be processed by one or more applications. Furthermore, the access and control of data information are now in the hands of Data Base Management Systems.

Today's Data Base Management System may be defined as a system which will manage virtually an unlimited number of data sets on an integrated basis and which will allow for entry and association of each of these data sets with any other data set in the data base, with provision for maintenance of data integrity. The objectives of such a system may be summarized as:

- Centralization and integration of data
- Flexibility in the selection, retrieval and modification of data
- Independent program and data base structure
- System security and data integrity

On-line applications in a DB/DC environment must be provided capabilities consistent with these DBMS objectives. Most notably, the DC software must carefully concern itself with security and data integrity. Furthermore, the extensive data base capabilities provided by a DBMS package are often achieved at a cost of increased CPU overhead. It is, therefore, incumbent upon the DC package to provide efficient application execution, in particular for a high degree of overlap between parallel application execution (multithreading) concurrent with DBMS services.

Therefore, discussion in this document does not investigate DBMS in particular but rather the facilities for use of DBMS with the INTERCOMM system to achieve the complete DB/DC environment.

To utilize a particular DBMS in a multithreaded environment, it is requisite that interface programs and routines be supplied. Such support of a DBMS should not modify the implementation procedures as prescribed by the DBMS reference manuals. Furthermore, all the facilities of the particular DBMS should be available to the user of any interface facility. Interface logic should not interfere with that of DBMS operation. Understandably, the guidelines for generation of an interface facility and maintenance of data integrity should be care and simplicity.

1.3 DBMS INTERFACE ENVIRONMENT

The on-line systems environment with INTERCOMM and a Data Base Management System consists of:

- The INTERCOMM region and associated user application programs requesting access of the data base
- The DBMS region controlling all data base access (in some cases the DBMS may be attached as an INTERCOMM subtask)
- Batch regions requesting access of the data base

All Data Base Management System functions are thus centralized and isolated within a region separate from the user (INTERCOMM or batch) jobs. The DBMS region should support concurrent use of its facilities by each user job, whether these separate jobs access the same or a different data base(s).

As depicted in Figure 1-2, INTERCOMM DBMS support consists of interface routines in each of the regions just described to accomplish communication and control functions between regions. Note that if the INTERCOMM Multiregion Support Facility (a Special Feature) is in use, 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 Multiregion Facility Users Guide. The multiregion version is required to support multiple Data Base Management Systems.

The INTERCOMM system is one of many users of the DBMS, each of which resides in its own individual region. Thus, the INTERCOMM region consists of the INTERCOMM monitor along with all on-line application programs including those which are using the DBMS for their file access techniques. The other DBMS users would be batch application programs, each residing in a separate region. All logic required for the data base access and control (exclusive control logic, data base

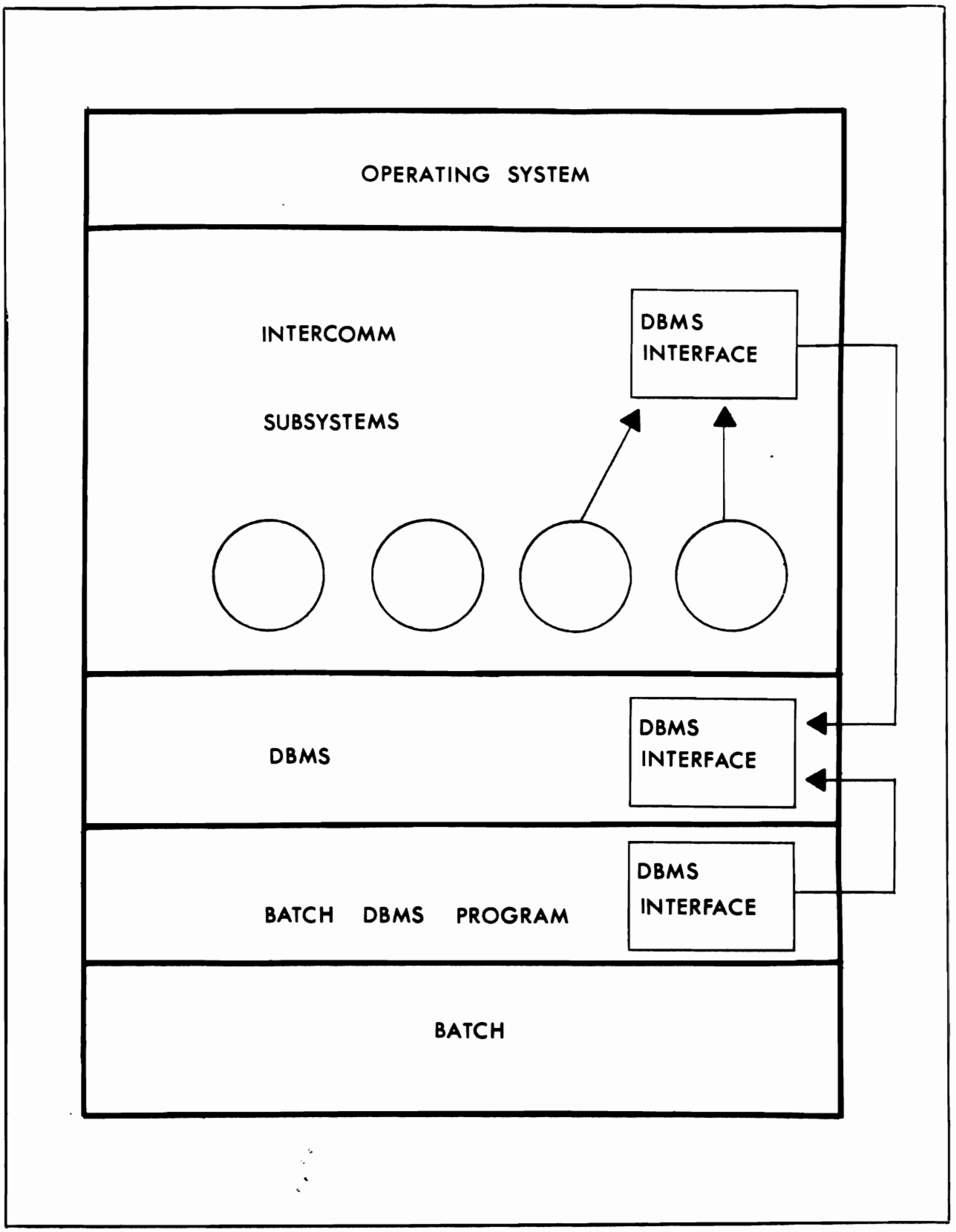

Figure 1-2. INTERCOMM and DBMS Environment (Single Region Version of INTERCOMM)

logging logic, etc.) is contained in the DBMS region; thus, there is no duplication of code in the user regions. The only logic required in each user region is the interface logic for communication of user data base requests to the DBMS region.

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 existing in its own region; the request is usually passed on to the DBMS via an interregion SVC (Supervisor Call). Where a system only performs inquiries to the data base, the required interface components are simple. Since the data base is not being altered, the only recovery procedures necessary are those of a dump/restore facility to be used in the case of total destruction of the data base.

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

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

- The data base access logic modules for performing the I/O activity 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, multi-threading provisions and recovery provisions.

1.4 DBMS INTERFACE COMPONENTS

The following interface facilities are common to most INTERCOMM/DBMS environments:

- An SVC Routine:

  Provides for interregion communication (except when the DBMS executes in the INTERCOMM region as an attached subtask).

- Initialization Processing Routine:

  Contains the initialization logic necessary to establish communication between INTERCOMM and the DBMS.

- Closedown Processing Routine:

  Contains logic necessary to notify the separately executing DBMS that the INTERCOMM region is closing down.

- Data Base Service Request Handling - On-Line Processing Routine, Batch Processing Routine:

  Contains logic necessary for communicating data base access requests from the user region to the DBMS region.

- DBMS Region Multitasking Routine:

  Provides capability to service concurrent requests within the DBMS region.

- Data Base Checkpoint and Restart Processing:

  Initiation of checkpoint processing is controlled by the time interval specified in the System Parameter List (SPA), activities of which are:

  - Initialization and continuation of interregion checkpoint.
  - Quiescing of data base activity within INTERCOMM at checkpoint time.

- Abend Processing Routines:

  Special logic necessary to retain data integrity if either INTERCOMM or the DBMS should abnormally terminate.

The following discussion further details the characteristics of the INTERCOMM interface facility for specific DBMS support categories.

1.5 GENERALIZED DBMS INTERFACE FACILITY

The INTERCOMM Generalized Data Base Management System Interface (referenced as GDB) consists of a series of programs which allow data base access from multiple regions or partitions while providing data integrity across program and system failure.

All but the specific data base logic necessary to provide any data base access is supplied by GDB in conjunction with user-supplied interface routines. Most application programs execute under the control of the INTERCOMM monitor. However, programs not under INTERCOMM's control (i.e., batch programs) may require concurrent and/or overlapping use of the Data Base Management System. Therefore, the GDB Interface to a DBMS also includes a provision for utilization of the DBMS by batch programs as well as on-line INTERCOMM programs.

In this situation where the specific data base logic is not supplied, all the control programs requisite to the DBMS interface for effectual use of the DBMS by on-line or batch programs are supplied by INTERCOMM, whether pertaining to startup, closedown, or restart/recovery procedures. In a sense, the INTERCOMM GDB 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. Thus, 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 time of the multithreaded TP region.
- At initiation of a data base request.
- At termination of a data base subsystem thread (normal and abnormal) for control of possible "lockout" situations.

In addition, full message restart processing is supplied to coordinate with data base recovery.

In the sections following which discuss general operating logic, reference is made to vendor-supplied modules. In the case of a user-written or nonsupported DBMS, this term then implies user-supplied modules.

1.6 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/I (IMS/DB), supplied by IBM
- TOTAL 5/6, supplied by Cincom Systems
- System 2000, supplied by MRI Systems

1.7 DL/I (IMS/DB) SUPPORT

INTERCOMM and its Data Language/I interface software execute as an IMS batch job. Users of the DL/I facility may be on-line INTERCOMM subsystems residing in the INTERCOMM region or other batch IMS application programs residing in separate regions. The reader is advised to refer to IMS vendor documentation for implications and possible restrictions when executing multiple batch IMS regions. The only logic required in the on-line region is, simply, the INTERCOMM-supplied interface logic for communication of user DL/I data base requests to the DL/I region.

Data Language/I is characterized by application program independence from access methods, from physical storage organizations and from characteristics of the device upon which the application data is stored. (This is due to a common symbolic program linkage and data base descriptions external to the application program.) Furthermore, DL/I provides sharing of common data for elimination of redundant storage which can include physical structuring of data over more than one data base. DL/I organization is based upon hierarchical structure. When operating against a DL/I data base, only the data predefined as sensitive is available for use in a particular application. Each application using the data base can be sensitive to its unique subset of data. Where an application has defined "sensitivity" to a subset of the data within a data base, modification and addition of nonsensitive data do

not affect the processing capability of the application. In addition, any application can be restricted as to types of data base requests made against its sensitive data. User-constructed standard IMS blocks define the scope access of each batch and on-line program.

From a user design point of view, the INTERCOMM support of DL/I provides for a high level of flexibility. The INTERCOMM interface provides for:

- Concurrent inquiry request to the same DL/I data base (single-threading of DB processing subsystems)

- Ability of on-line inquiry and/or update programs to execute concurrently with batch inquiry and/or update program(s) against the same or different DL/I data base(s). Considerations are noted in DL/I Installation, Section 6.

1.8 TOTAL SUPPORT

Support for Cincom Systems' TOTAL System (TOTAL 5/6) is provided INTERCOMM users through the INTERCOMM/TOTAL Interface Facility. The TOTAL spectrum of DBMS capabilities includes: elimination of reprogramming due to application/data base expansion, elimination of index areas and separate addition areas (reducing direct access storage requirements) and provision for direct access of all records. TOTAL systems are designed and implemented with the real-time concept of operational versus analytical processing. Operational processing involves the immediate application of the transaction pool to all the data base parts, thereby reflecting the current status of business through inquiry and exception reporting. Analytical processing is the periodic extraction and reporting of data to support effective management planning. All the facilities of TOTAL are secured by the INTERCOMM/TOTAL interface.

Because TOTAL operates at the call level with the host programs (i.e., COBOL, PL/I or BAL programs) and as a logical direct access file management program with the operating system, the user is empowered to readily convert old programs to a data base approach. Thus, the host programming language functions are specified by the application program.

Two modes of INTERCOMM and TOTAL operation exist. INTERCOMM and TOTAL 5/6 may operate in the same region (or partition) as a main task and subtask, respectively. With this method, the ATTACH option must be SYSGENed. 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 the interregion SVC, supplied by TOTAL. Coordinated INTERCOMM/TOTAL support provides data base integrity and affords restart/recovery procedures.

When 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.

1.9 SYSTEM 2000 SUPPORT

Support for MRI's System 2000 is provided for both the Natural Language Interface (NLI) and Procedural Language Interface (PLI). An MRI-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.

1.10 DBMS INTERFACES VIA GDB

Many 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 fasion are:

- ADABAS--a product of Software A.G.
- IDMS--a product of Cullinane Corporation
- Model 204--a product of Computer Corporation of America

DBMS implemented via GDB offer to the user all the GDB Interface facilities previously described in addition to the salient features of the individual DBMS as discussed in the following pages.

1.11 ADABAS SUPPORT

ADABAS, a product of Software A.G., is a DBMS utilizing inverted file structure. The interface is comprised of INTERCOMM's Generalized Data Base Management System and a special INTERCOMM/ADABAS interface program. ADABAS operates in a separate partition or region and may 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, are used for all data base activity against ADABAS files.

ADABAS itself consists of an operational nucleus which performs the most commonly used ADABAS functions. ADABAS also includes a number of utilities used for special, infrequently requested functions such as: loading, scratching, dumping and reorganizing files.

ADABAS operation can be categorized as follows: initial definition and loading, definition modifications, data update, data query, data read, checkpoint/restart, save/restore and data base maintenance. All aspects of these data base services and the advantages of ADABAS are made available via the INTERCOMM interface.

1.12 IDMS SUPPORT

The Integrated Database Management System (IDMS) is a software product marketed by the Cullinane Corporation. A comprehensive DB/DC environment is provided for IDMS via INTERCOMM and Cullinane-provided interface routines. IDMS is developed on the basis of the standards formulated by the CODASYL committee. Characterized by a network type file structure as defined by the CODASYL Data Base Task Group (DBTG), IDMS makes use of the data management language syntax defined in the CODASYL 1971 DBTG report.

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 may be called from one or more batch regions while INTERCOMM is operational. INTERCOMM support does not modify IDMS requirements as specified in the IDMS Database Design and Definition Guide. Intercomm message restarting may be used in conjunction with the data base, which provides checkpointing and recovery functions.

1.13 MODEL 204 SUPPORT

Model 204 is a Data Base Management System that utilizes the efficiency and comprehensiveness of an inverted file organization. Operating under IBM's 360/370, Model 204 runs in a separate region or partition and is usable for application programs coded in BAL, COBOL or PL/I, either batch or on-line subsystems. Model 204 may be utilized by one or more batch regions concurrent with INTERCOMM. Model 204's standard calling sequences are not altered by INTERCOMM support. Interfacing between INTERCOMM and Model 204 consists of INTERCOMM's Generalized Data Base Management System supplemented by an INTERCOMM/Model 204 Interface.

Simply, 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 that of INTERCOMM, the data base is fully recoverable from failure situations in which it was either physically or logically destroyed.

The inverted file is a collection of records to which access is directly provided via values of one or more keys. The functional advantages of the inverted file structure to which Model 204 adheres are as follows. First, the final structure allows for extremely rapid access to variable-length, variable-format records based on the properties of one or many different key fields in a file. In fact, all fields can be key fields if desired.

Additionally, fields may also be of variable length providing for file compression as well as simple text processing capability. A Model 204 file can contain a set of over 16 million variable-length, variable-format records. There is no limit on the number of key fields per record. No storage space is taken for fields which do not appear in a given record. The Model 204 file organization technique assures transparency and lack of redundancy to the application programs. The dynamic nature of the Model 204 data base also obviates the need for reorganization when fields are added or deleted.

Data retrieval, based on Boolean combinations of key field values, provides desired flexibility in retrieval and modification of data within the data base. Model 204 files may be organized and/or accessed in sorted sequence or with direct access through a hash key technique.

Application programs may be written in COBOL, PL/I, BAL and FORTRAN. Additionally, Model 204 provides a powerful, easy to use natural language access. Multiple batch and on-line applications written in different languages can access the data base simultaneously.

2. SERVICING DATA BASE USER REQUESTS

2.1 GENERAL

An INTERCOMM DBMS interface facility performs many functions in the course of servicing data base user requests, regardless of the actual DBMS in use. This section provides a conceptual description of system operation for establishing communication between the DBMS and user regions, for processing data base requests from the user regions and for the orderly closedown procedures of the DBMS and user regions.

In this section, concepts and program names are used which pertain to the INTERCOMM Generalized DBMS Interface Facility for convenience and consistency of reference. Reference is made to "programs;" the actual support routines may be separate load modules, entry points within one load module or CSECTs of a load module. The reader is referred to the individual sections which describe particular DBMS implementation for the specific techniques in use for a particular DBMS.

The relationship of the DBMS interface programs referenced in this section is illustrated in Figure 2-1. With the exception of checkpoint processing and restart/recovery functions (see Section 3), the following discussion provides the reader with an understanding of the logic involved in servicing data base requests from user regions. It is assumed the reader is generally familiar with the functions of the various INTERCOMM system components, in particular the multitasking Dispatcher.

2.2 STARTUP PROCESSING

Because a DBMS is usually operated as a separate job, it is activated prior to any jobs which use the DBMS facilities (unless it is operating as an INTERCOMM subtask). All data base access takes place from this region. All Job Control Language (JCL) relative to the data base(s) is placed in the job stream for this region. All control blocks and buffers relative to data base activity are defined and included as part of this region.

Startup processing of the DBMS region initializes the DBMS for receiving all user access requests. All programs required to be in core are loaded, data base and/or program control blocks are loaded or initialized as required and ABEND exits initialized. These functions are performed by the INTERCOMM/DBMS facilities and are unique to the INTERCOMM/DBMS environment as they are functions pertaining to multithread operation and are not performed in a non-INTERCOMM/DBMS environment.

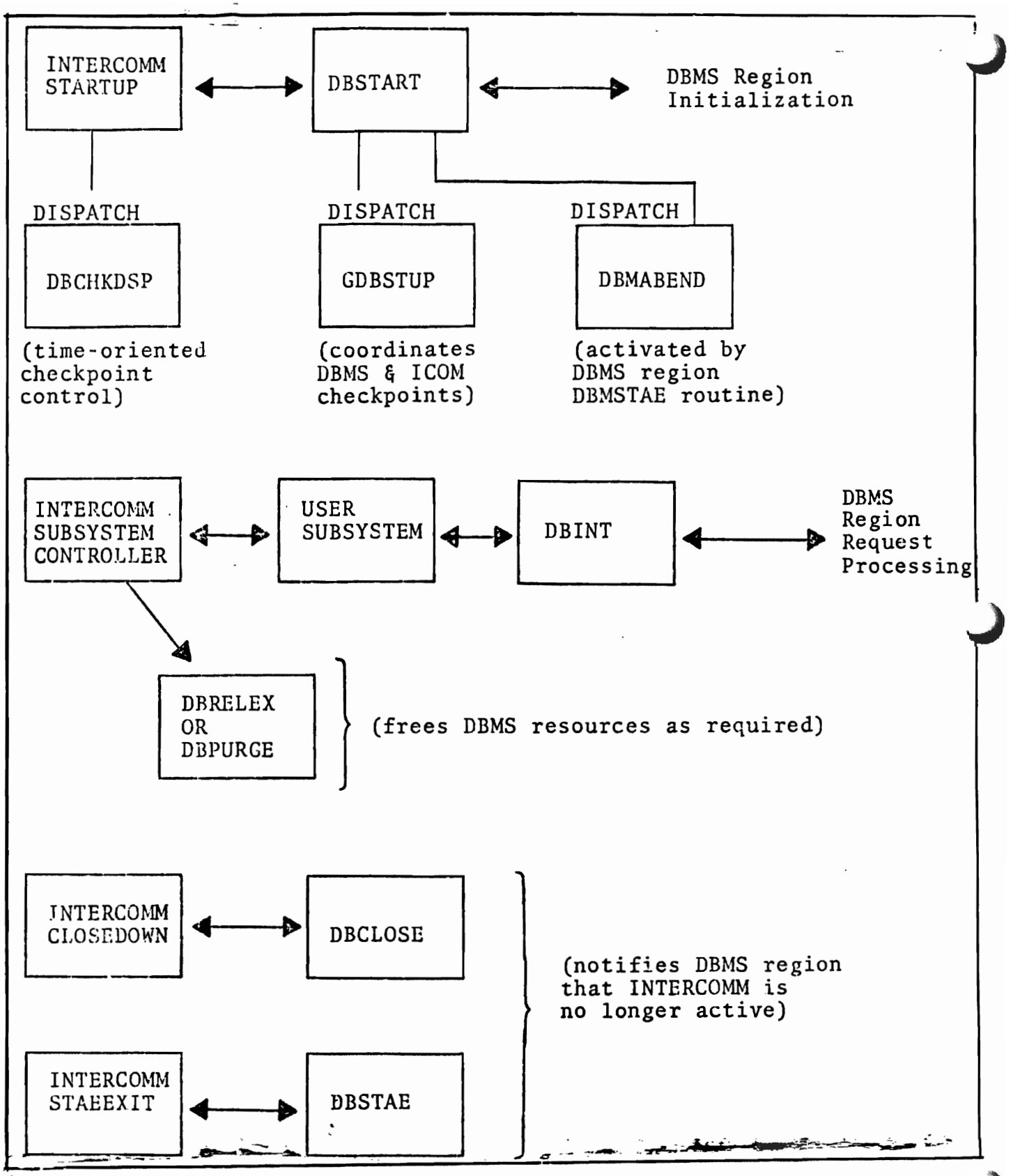

Figure 2-1. DBMS Interface Routine Summary

A user program accessing data bases often requires description by a DBMS control block or table as do the areas of the data base which the program will be using. After the DBMS region is started, any DBMS control blocks for on-line programs are, typically, automatically loaded into the DBMS region at INTERCOMM startup time. Any control tables for INTERCOMM application subsystems exist in the INTERCOMM region.

The on-line INTERCOMM job may have multiple DBMS control blocks associated with it such as one control block per message processing thread. A batch program has only one associated control block. When a batch program is started, its control block is generally loaded dynamically; thus, all control blocks for all possible batch programs do not constantly reside in core.

As a consequence of activation prior to the starting of user jobs, the DBMS region must have a way of knowing when a user region is activated. To provide such communication, the INTERCOMM interface facility places in its Task Control Block (TCB) the address of an Event Control Block (ECB) followed by an identifier of the DBMS region. When either an INTERCOMM on-line job or any batch job which uses the DBMS is begun, the startup logic of these jobs searches the TCB chain for the DBMS region. This search is performed by checking for the identifier of the region. When the DBMS region is found, the ECB is posted, thereby indicating to the INTERCOMM/DBMS interface facility that a new user region is starting. In identifying itself, the user region supplies a communication path to be used on all service requests from that user region to the INTERCOMM/DBMS interface facility.

During INTERCOMM startup processing, the basic steps required to prepare for communication with the Data Base Management System are performed by the startup program (DBSTART), called to perform all the initialization logic necessary to establish communication between INTERCOMM and the DBMS.

After INTERCOMM becomes active, periodic checkpoint processing is performed by the checkpoint program (DBCHKDSP) which is dispatched at startup on a time interval specified by the TCHP operand of the System Parameter List (SPA). (Detailed discussion in Section 3.)

DBSTART may reside in the INTERCOMM startup overlay region. The functions that typically are included in DBSTART are as follows:

- The ATTACH is done if the DBMS is to operate as a subtask of INTERCOMM.

- The existence of the DBMS, if it resides in another region, is determined. If that region is not operational, some action is performed, such as writing a message to the operator directing him to "bring up" the region. If it cannot be activated and the on-line system is needed for other than data base access, all subsystems that access data base files are marked non-schedulable but the on-line system may continue to execute.

- A communication path between INTERCOMM and the DBMS is established. The method typically used is to establish a pair of ECBs which are alternately posted and waited on by the two regions. This technique of interregion communication is typically referred to as a "software channel." These ECBs must reside in the INTERCOMM region since the INTERCOMM DISPATCH macro is used to accomplish the wait and the Dispatcher makes address validity checks on all ECBs being waited on. When notified that INTERCOMM is operational, the DBMS interface will then trigger any initialization necessary in the DBMS region (i.e., loading control blocks, logging a startup record, etc.).

- At this time, provision is made for a potential abnormal failure of the DBMS region. The typical method is to dispatch in the INTERCOMM region a small resident program (DBMABEND) which waits on an ECB to be posted by the DBMS region's STAEEXIT program (DBMSTAE).

- If interregion checkpoint synchronization is necessary (for data base integrity) a program for checkpointing (GDBSTUP) is dispatched in the INTERCOMM region to initialize interregion checkpoint control. A DBMS checkpoint workfile may be utilized to ensure that INTERCOMM startup will be capable of determining the necessary coincidental checkpoint time for the data base restoration process.

- The DBMS region, upon receiving a sign-on of INTERCOMM, makes note of the event on its log to provide the data base backout program with this information.

- If any data sets must be opened for data base access, this is done via the appropriate call to the vendor-supplied module. Often, the data sets are identified by a data file table in the INTERCOMM region.

2.3 USER REGION DBMS REQUEST PROCESSING

Whenever a request for DBMS services is made by an on-line or batch user program, the request is passed to the DBMS region through the interregion communication mechanism. To effect this communication, a user SVC routine must be included in the user's operating system (see Section 4 for implementation particulars). For the on-line INTERCOMM region, many concurrent requests may be outstanding at a given time; for a batch region, only one request is outstanding at any time.

The DBMS region accepts requests for service as they are issued by the application program. It is not always possible, however, to immediately service a request when it is received. There are several reasons a DBMS may not be able to immediately satisfy a request. These reasons include a busy condition on a single-threaded DMBS, a data conflict lockout situation and/or a specified limit of the maximum number of concurrently serviced requests. If the request cannot be immediately serviced then the DBMS must provide the facility to queue the request in proper sequence and execute it when possible.

From the user region viewpoint, all service requests are handled by the DBMS interface program (DBINT). This interface accepts the call from the user program and, utilizing an interregion SVC, communicates the request to the DBMS region. The coding conventions for the user's call are based on the programming language being used and the parameter requirements of the DBMS. Certain DBMS supply a precompiler that will generate appropriate calling sequences. The DBMS to be supported may require calling different entry points for each function it is requested to accomplish. In this case, the special entry point (DBINT) must still be present although the application request may use other entry points additionally or exclusively.

Since Data Base Management Systems vary greatly in the manner of their interface with application programs, no further description of these interface particulars will be attempted here; each DBMS is described individually in subsequent sections in this respect.

2.3.1 INTERCOMM Subsystem DBMS Requests

The INTERCOMM region subsystem DBMS request program is a reentrant program which transfers application data base requests from the on-line region to the data base region. There may be one or more entry points necessary in this interface program depending on the conventions of the DBMS being

supported. The entry point names normally coincide with the standard entry point names for the DBMS. Again, a DBMS precomplier, if used, must generate appropriate calls. Also, INTERCOMM language-dependent multithreading interface routines, such as COBREENT, must be utilized where required as DBMS calls are analogous to INTERCOMM File Handler calls.

The INTERCOMM message processing control routine will call DBPURGE (if INCLUDEd) at the completion of each message processing thread to ensure all DBMS resources are freed, whether or not the thread terminated normally.

2.3.2 Batch Program DBMS Requests

Batch programs are usually able to run concurrently with INTERCOMM on-line programs while accessing and updating the same data base. The DBMS region has the responsibility for handling the processing of many batch jobs concurrently insofar as core permits. (If the DBMS lacks multithreading capabilities, it must stack concurrent service requests then process them serially. Thus, these Data Base Management Systems provide apparent, not actual, concurrent processing.) Batch application programs which are inquiry-only, in general, require no change beyond that required for interregion DB communication. However, programs which perform updates to the data base must have the ability to participate in a synchronized checkpoint between INTERCOMM and DBMS regions. Each DBMS provides this function in a distinct manner. These techniques are described individually in subsequent sections pertinent to specific DB systems.

2.4 CLOSEDOWN PROCESSING

During normal INTERCOMM closedown processing, the DBMS closedown program (DBCLOSE) is called. This program notifies the DMBS region that INTERCOMM is ceasing processing, specifying whether termination is normal or abnormal (i.e., an ABEND). DBCLOSE includes these functions:

- Freeing of any control blocks, if relevant, which have been used for interfacing with the DBMS.
- Detach of the DBMS if executing as a subtask in the INTERCOMM region.

The nature and extent of processing which must be performed at closedown greatly depends on the specific DBMS.

When the DBMS is notified by DBCLOSE of the on-line job completion, data bases used by Intercomm on-line programs may be closed by the DBMS if these data bases are not in concurrent use by a batch program.

2.4.1 Batch Program Completion

The DBMS region, at notification of a batch job completion, closes all data bases and, if control blocks are used, purges all those related to the particular program. In this manner the DBMS region can handle the processing of many batch jobs concurrently although it may be limited by available core.

2.4.2 DBMS Region Closedown

The DBMS region will close down only when there are no on-line or batch users of the facility. Typically, closedown is effected through a computer operator response to a WTOR. If the DBMS region terminates abnormally while some user regions are operating, each user region is notified. No further use of the DBMS region is possible at this point. Within the Intercomm region, DBINT will continue to accept calls for DBMS; however, every such call is completed by providing an error status indicating the DBMS is no longer operational.

3. MAINTAINING DATA BASE INTEGRITY

3.1 GENERAL

INTERCOMM DBMS support generally allows on-line and batch programs to execute concurrently while accessing and updating the same data base. Use of an INTERCOMM/DBMS interface includes provision for data base integrity which is maintained across system and program failures. Restart/recovery procedures consist of standard DBMS recovery utilities and INTERCOMM system logic. The INTERCOMM operation basically consists of logging and checkpoint procedures, INTERCOMM and DBMS checkpoint synchronization and the INTERCOMM checkpoint subsystem processing.

The primary technique used to maintain data integrity is to quiesce all data base updating across all regions at appropriate intervals during the day. At each of these intervals a simultaneous checkpoint is taken of the INTERCOMM, DBMS and DBMS batch regions. In the event of a failure, it is then possible to recreate the data base, the INTERCOMM region and the batch region(s) as they existed at the same checkpoint time. Basically, this capability is based on both systems (the DBMS and INTERCOMM) backing up to the last checkpoint and restarting all processing affecting the data base initiated since that time by utilizing the log records created by both systems. All data base update activity completed between the checkpoint and the failure may be restarted to bring the files up to the point of failure. Synchronization of INTERCOMM and the DBMS checkpoint is accomplished by allowing INTERCOMM to initiate all checkpointing for both tasks while the DBMS and INTERCOMM simultaneously operate. Should the DBMS lack "data reversal from log" capability, then INTERCOMM can utilize the last system dump/restore point, reprocessing all transactions subsequent to this point.

For any data base failure, the logic required across that failure typically consists of either running a standard DBMS utility or running INTERCOMM/DBMS restart along with a standard DBMS utility (or utilities). With INTERCOMM DBMS support, if a data base is lost or destroyed in any manner, the DBMS user is guaranteed reconstruction. This is accomplished through a DBMS utility-restore followed by a forward reconstruction. If a batch DB update user of INTERCOMM itself fails then the reversal to checkpoint technique is used.

This section will examine these failure conditions, logic for maintaining data base integrity and the procedures for checkpoint, abend and restart processing.

3.2 DATA BASE FAILURE CONDITIONS

There are four types of data base failure conditions which could be encountered. These are:

- Physical or logical loss of (all or part of) a data base
- Failure of a batch program using the DBMS in update mode
- Failure of the on-line INTERCOMM system using the DBMS in update mode
- Any combination of the other three conditions

In the following discussion, these conditions are examined and action to rectify each is presented.

3.2.1 Loss of Data Base(s)

The first failure condition includes any situation in which a data base was either stolen from an installation or partly or wholly destroyed (e.g., a head crash). Any file structure which can be updated requires periodic dumping (backup) of the entire file contents for recovery and/or historical purposes. This might be performed weekly, monthly or on any scheduled basis determined by the installation. Recovery of the data base requires a restore of the data base from the last backup copy taken.

Since restoration of the data base is only useful for recovery purposes up to the time of the most recent backup, a "restored" data base is not ordinarily up-to-date. Recovery for information processed against the data base, subsequent to the last backup, must include a facility capable of reapplying all updates occurring subsequent to the restore point. This is usually performed by log after-images of updates. These after-images can be applied in a forward chronological sequence to bring a restored data base to a current condition.

3.2.2 Failure of Batch Program(s)

The second condition mentioned previously is the failure of a batch program using the DBMS. When this occurs, all updates performed by the batch program must first be reversed. A utility to eliminate all update records from a specified time onward must be provided. The specified time may be chosen from a runoff sheet which indicates all the checkpoint

times. When the INTERCOMM and the DBMS are running simultaneously, checkpoint time is usually determined by a backout program.

If the failure is due to a software error in the user's batch program, recovery is still possible. However, recovery consists of reversing the data base updates performed by the batch program. The program error must then be corrected and the complete job must be rerun using the corrected program. The reversal of the data base activity accomplished by the program is performed by the Data Base Backout Utility program.

If the DBMS lacks reversal capability, as in System 2000, the provisions for data base loss conditions must be carried out.

3.2.3 Failure of the On-Line System

The third failure conditon encountered is the failure of the on-line INTERCOMM system which uses the DBMS and the INTERCOMM interface. Although more complex, this failure condition is functionally very similar to that previously given for batch programs. The complexity is introduced by the multithreaded environment existing during execution of the INTERCOMM on-line system. In essence, many programs are operating in parallel as INTERCOMM is running. At the time of the failure, one or more programs may be in the midst of execution. This is contrary to the batch program failure condition where only data base activity is being performed by the batch program following on the time of failure. To handle data base integrity during on-line system failure, the computer operator must usually invoke backout of the data base via the DBMS vendor-supplied backout utility program and then must restart INTERCOMM. (If the DBMS is still executing, it must be quiesced.) In restart mode, INTERCOMM will reverse OS and VS file updates to the point of the last common DB/DC checkpoint. The execution of the data base backout program should select the last checkpoint or the operator may check the console sheet and select any other appropriate time. Since both the DBMS and INTERCOMM are restoring to the point of the last valid checkpoint, any data base processing program which updates files must be reprocessed. For this reason, INTERCOMM restart logic restarts update messages that had previously been executed subsequent to the checkpoint time as if these messages had not been processed. Message processing programs not using the DBMS or DBMS inquiry programs will not have to be reprocessed since such programs are in no way affected by the recovery logic. All of the processing described is accomplished automatically without any user programming required. (Restart logic treats reprocessing or DBMS updating programs

identical to OS or VS file updating programs. In fact, DBMS and other file updates can be combined in a single program. The File Recovery Special Feature is required for OS/VS file integrity.

A failure situation requiring message reprocessing in no way changes standard INTERCOMM procedures for message integrity and recovery: any message that had not processed prior to the INTERCOMM on-line system failure or was in process at failure time will be recovered and processed. The process described above merely provides additional logic for messages that:

- Were processed prior to the failure of system
- Use the DBMS or OS or VS files
- Update data base files or OS or VS files

Considering the conditions possible after a failure of the total INTERCOMM on-line system, the various states of processing in which a message could exist are:

- Message 1

  Message has been processed prior to failure and prior to last valid checkpoint.

- Message 2

  Message had been processed prior to failure but after the last valid checkpoint.

- Message 3

  Message was in process of execution during system failure.

- Message 4

  Message was received by INTERCOMM prior to failure but no processing performed (message was on an input queue).

Any message in the system at the time of failure must be in one of these four states. Let us examine the recovery process from the viewpoint of the individual message and observe how INTERCOMM restart generally treats each case.

In the first case cited, the message would not be recovered after a failure since all processing had been completed.

The second type of message (processing completed but subsequent to the last valid checkpoint) must be separated into two distinct categories as follows:

- The message is processed by a program that does not use the DBMS or by a program using the data base in an inquiry (read-only) mode. In this instance, INTERCOMM recovery performs the same functions described under Message 1 above. A user option allows the on-line program to indicate to the system during execution (via its return code) that such a message is always to be reprocessed. Normally, however, reprocessing its inquiry-only messages would be superfluous; therefore, it is avoided.

- Alternatively, this message is processed by a program that accessed the data base in an update mode. In recovery mode INTERCOMM would mark this message for reprocessing and place it on the INTERCOMM queue in its original sequence relative to other messages. Again, a user option can reverse this standard INTERCOMM logic by using a special return code from the data base update subsystem during its normal operation. (One possible use of this special return code option might be to inhibit processing of a message which did not perform any data base updates. That is, the subsystem is marked in the INTERCOMM SYCTTBL as a data base update program but in execution of this particular message no updates were performed.)

The user is cautioned to be careful in using special return codes to overrride INTERCOMM recovery logic. In all failure cases, the DBMS will have reversed all updates to checkpoint. Only if all updating messages are reprocessed (as would occur in standard INTERCOMM recovery logic) will data files be identical after a failure and full data integrity be assured.

In the third message type (message processing was in progress during system failure), the message is marked for reprocessing and placed back on the INTERCOMM queue in its original queuing sequence relative to other messages. As described for Message 2 above, any data base updates performed by the program processing this message prior to the failure are reversed and backed out of the data base (by the DBMS utilities).

For the fourth message type (the message was on an input queue at failure time, hence no processing transpired), the message is merely requeued in its original sequence relative to other messages.

For a summary of the various processing actions performed by INTERCOMM recovery relative to messages and to their status at the time of failure, see Figure 3-1.

Message Status	INTERCOMM Restart	Default Procedure		Option
		Queue Message	Discard Message	Return Code Option Allowed to Reverse Default
Message processed prior to failure and prior to last valid checkpoint.	Message not recovered after failure because processing was completed.	NO	YES	NO
Message processed prior to failure but after last valid checkpoint.	a. Processed by a program not using DBMS or using DBMS in an inquiry (read-only) mode.	NO	YES	YES
	b. Processed by a program that used DBMS in update mode.	YES	NO	YES
Message in process of execution during system failure.	Marked for reprocessing and placed back on INTERCOMM queue in its original sequence reltive to other messages.	YES	NO	NO
Message received by INTERCOMM prior to failure, but no processing performed on message prior to failure	Message will be marked for processing & placed back on the INTERCOMM queue in its original sequence relative to other messages.	YES	NO	NO

Figure 3-1. INTERCOMM/DBMS Restart and Recovery

The contents of the queues following a recovery is as follows:

- Previously completed messages being reprocessed
- Messages that were in process at failure time
- Messages that were on the queue at failure time
- New message traffic arriving after restart

DBMS region data base recovery is always coordinated with INTERCOMM message recovery. This coordination insures that the checkpoint used by the DBMS for data base recovery is

the same checkpoint as that used by INTERCOMM for message recovery. DBMS action for failure of the on-line INTERCOMM system is the same relative to the data base(s) as for failure of a batch program described earlier. All DBMS data bases will be restored to their status at the point of the last valid checkpoint. As previously mentioned, the INTERCOMM File Recovery Special Feature provides capabilities analogous to a DBMS to ensure OS and VS file integrity. File Recovery procedures are integrated with DBMS recovery procedures.

3.2.4 Combinations of Failure Conditions

In any situation encompassing multiple failures of the nature described earlier, recovery logic is easily defined using the facilities already discussed. The critical element is to understand the nature of the recovery mechanisms and to apply these tools in the appropriate sequence.

As a simple example, assume that a head crash occurs while a batch program is running. In this situation, recovery would consist of the application of two utilities already described. Initially, the data base must be restored to the moment of the combined failure and this is done by using the loss of data base logic described. Once the data base has been recovered to the most recent moment of operation, the situation remaining is the normal case of failure of a batch program.

In a more complicated situation, let us assume that an OS failure occurred while a batch program was executing and the on-line INTERCOMM system operating. When this occurs, the basis for recovery is the last checkpoint taken of the entire environment. An INTERCOMM/DBMS interface backout utility would be used to back out data base updates to the last checkpoint. No activity, whether batch or on-line, having affected the data base(s) after this last valid checkpoint would be reflected once this recovery has occurred. To ensure proper reprocessing of any activity in the batch and on-line systems, both systems must be restarted. In regard to the on-line INTERCOMM system, a restart of the INTERCOMM job will affect reprocessing any lost data base activity and continuation of the on-line activity. The data base backout program brings the data base back to the last checkpoint; then, the DBMS region would be brought up and any batch jobs specified by that program rerun.

Under the topic "Failure of a Batch Program," it was mentioned that were the failure due to a program error, all data base activity would have to be reversed back to the beginning of the job. With more than one job running concurrently, the

implication of doing such is that the checkpoint needed for reversal must be one coincidental with totally quiesced batch updating.

These are the various stages of processing when abnormal termination occurs, the conditions necessitating such termination and the logic behind restart and recovery with respect to a data base(s) in an INTERCOMM/DBMS environment, whether for the operating system or for batch or on-line programs. The maintenance of data base integrity for the INTERCOMM/DBMS user is assured by INTERCOMM-provided procedures in conjunction with programs provided by the user or vendor of a particular DBMS. The operational concepts for the checkpoint, restart and abend programs are described in the remainder of this section.

3.3 CHECKPOINT PROCESSING

Maintaining data base integrity is based upon the availability of the INTERCOMM System Log and an activity log (provided by the DBMS) recording data base update activity. Checkpoints in the INTERCOMM/DBMS environment consist of recording only pertinent table data, not a "core" checkpoint in the normal operating system sense. This section presents a conceptual description of the checkpoint processing including the technique used to synchronize the INTERCOMM and DBMS region checkpoint. Note again that concepts and program names correspond to the GDB Interface Facility; refer to the following DBMS-dependent sections for specific detail regarding a particular DBMS.

Synchronous checkpoints of the INTRCOMM and DBMS regions are performed at the following points in time:

- At INTERCOMM startup
- At a periodic interval specified by the user in the INTERCOMM System Parameter List
- At Batch Program startup and closedown (for those batch programs that perform updates)
- At INTERCOMM closedown

INTERCOMM will, when operating, initiate all checkpoints within its region. Under certain circumstances checkpoint processing can also be initiated from the DBMS region. However, to maintain data integrity when utilizing INTERCOMM/DBMS support, the checkpoint facilities of both must be synchronized.

This is accomplished by allowing Intercomm to initiate all checkpointing for both tasks while the DBMS and Intercomm simultaneously operate. The time interval to be used for checkpoint is specified as a system parameter of the Intercomm region. DBMS checkpoints may also be controlled based on the number of updates when Intercomm is not running. For some DBMS, a checkpoint request may be initiated via a terminal.

A complication in synchronizing Intercomm and DBMS checkpoints arises when batch region(s) which update the data base are active at checkpoint time. This situation necessitates a checkpoint request to the DBMS region when it is time to perform an Intercomm checkpoint. However, the Intercomm region may only perform a valid checkpoint when the DBMS region notifies it that all batch program update activity has quiesced. When Intercomm is active, it initiates all checkpoint requests. The DBMS honors such requests at its discretion.

The following programs are involved with checkpoint processing in the Intercomm region:

- DBCHKDSP

  Triggered at startup by a time-oriented DISPATCH for the interval specified by the TCHP parameter of the SPALIST macro (contains replacement CSECT CHECKPT for that in CHECKPT3)

- DBCKPREP

  Dispatched by DBCHKDSP to request acknowledgement that a checkpoint of the DBMS is possible

- GDBSTUP

  Triggered at startup by an event-oriented DISPATCH to wait for the DBMS region to signal checkpoint preparation (quiescing of batch programs) is complete

- CHCKPTSS

  The checkpoint subsystem which receives a message from GDBSTUP, quiesces all data base update subsystems and then takes the INTERCOMM checkpoint

- DBCHKCOM

  Called by CHCKPTSS to notify the DBMS that Intercomm checkpoint is complete and request a checkpoint of the data base region

Checkpoint functions in the DBMS region are in general controlled by the DBMS logic activated by DBCKPREP and DBCHKCOM.

3.3.1 Checkpoint Logic Flow

The basic flow of checkpoint processing is shown in Figure 3-2. The numbers therein correspond to the following brief descriptions of each step within the overall checkpoint processing flow.

(1) At INTERCOMM startup time, DBSTART dispatches GDBSTUP. Via ECBs, GDBSTUP will wait until it receives a signal which indicates it is time to take an INTERCOMM checkpoint.

(2) INTERCOMM, based upon a specified time interval in the SPA, periodically dispatches DBCHKDSP. On receiving control, DBCHKDSP dispatches DBCKPREP.

(3) DBCKPREP then sends a "checkpoint prepare" command to the Data Base Management System.

(4) The Data Base Management System now interrogates all involved batch regions to determine if it is possible to take a valid checkpoint (i.e., no active batch update regions).

(5) When all checkpoint criteria have been met the DBMS calls DBMCHECK which resides within its region.

(6) GDBSTUP is activated by SEXTOECB in the INTERCOMM region being posted with a code of 8, either directly by the DBMS or by response to the checkpoint prepare command.

(7) GDBSTUP then sends a message to the checkpoint subsystem and proceeds to wait until all checkpointing is complete.

(8) The checkpoint subsystem quiesces all data base update subsystems, logs a checkpoint record, takes the INTERCOMM checkpoint and calls DBCHKCOM.

(9) DBCHKCOM subsequently sends a checkpoint command to the DBMS. Upon completion of the DBMS, checkpoint SEXTOECB in the INTERCOMM region is posted with a code of 12.

(10) The Data Base Management System then calls DBMCHECK to update a checkpoint completion count, i.e., the number of regions requiring checkpoints and to initiate the data base region checkpoint. NOTE: DBMCHECK is hypothetically a program which verifies that all checkpoint criteria have been met, i.e., batch programs have responded. Dependent on the data base management system, these functions may automatically be provided as part of the checkpoint or checkpoint prepare command.

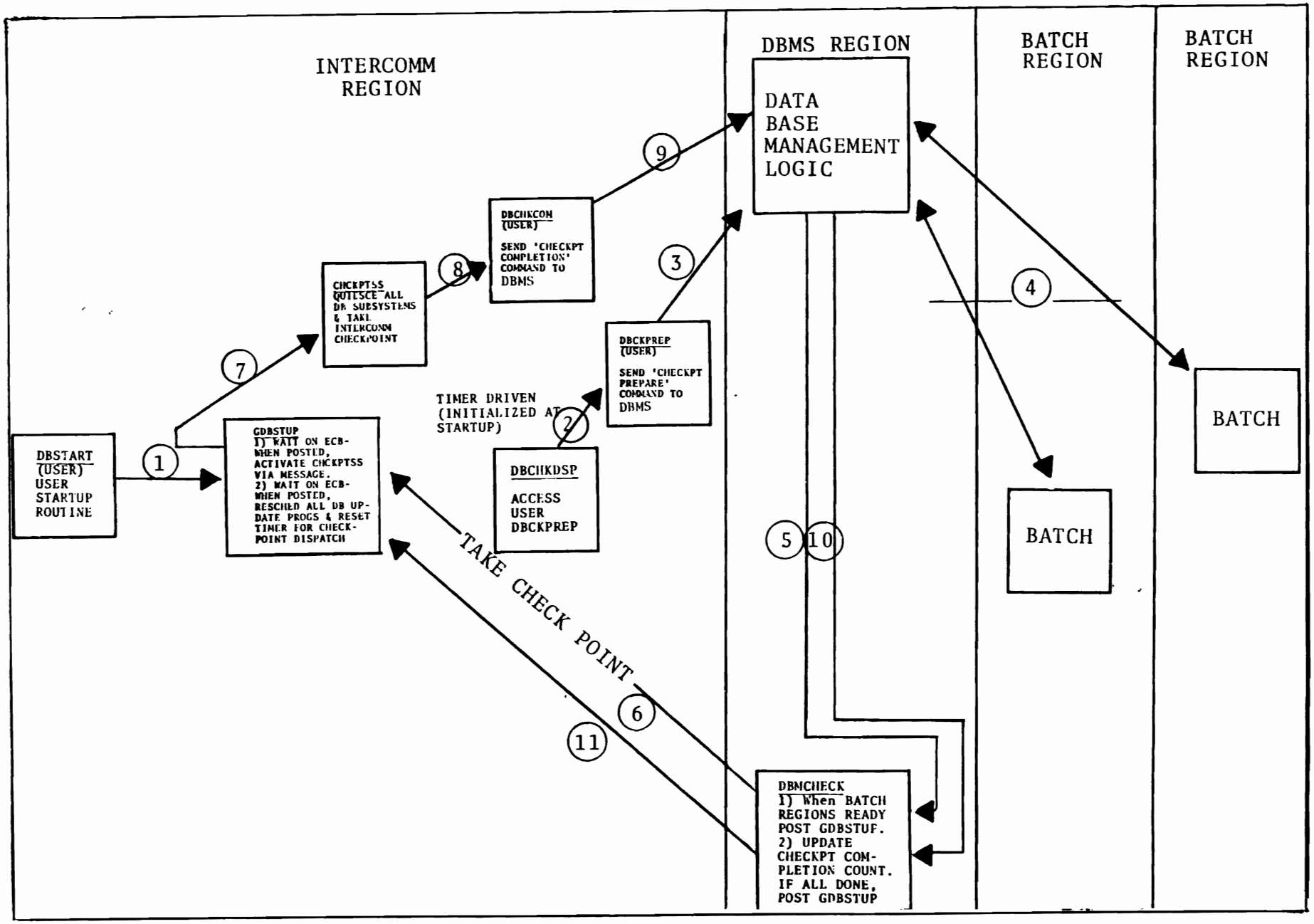

Figure 3-2. Checkpoint Processing Flow

(11) DBMCHECK, once checkpointing has completed, will also reactivate GDBSTUP by posting SEXTOECB with a post code of 12 in the INTERCOMM region. GDBSTUP now logs a checkpoint record, marks data base subsystems as schedulable and redispatches DBCHKDSP on a timer interval.

3.3.2 Checkpoint Subsystem Logic Flow

The following describes the specific actions of the checkpoint subsystem, CHCKPTSS.

This program operates as an INTERCOMM subsystem (SSC=Q) and ensures that all data base updating programs are quiesced by marking them as nonschedulable and ensuring that current messages in progress have completed before allowing INTERCOMM checkpoint to take place. The checkpoint subsystem also takes the INTERCOMM checkpoints.

Upon receiving control, the checkpoint subsystem takes no action until all message processing for programs updating DBMS files has been completed. It then uses the following method to quiesce these programs. There are bits which are defined in the Subsystem Control Table and indicate whether or not a subsystem accesses DBMS files and whether or not it may perform data base updates. To quiesce update processing, CHCKPTSS scans through all the SCTs and sets a special "unschedulable" bit in all entries having both DBMS file access and DB update activity bits on. This prevents any new update-type messages from being started by the Subsystem Controller.

CHCKPTSS then does another scan of all the SCTs. If both the data base access and update activity bits are on and a message is in progress for the subsystem, CHCKPTSS must wait on an ECB defined in the SPALIST. This ECB is posted by the Subsystem Controller when any message completes for a DBMS updating program. Until all DBMS updating programs are quiesced, the checkpoint subsystem will continue to receive control to scan the SCTs and repeat the above sequence.

CHCKPTSS requests a checkpoint of the DBMS via the module DBCHKCOM and calls the INTERCOMM checkpoint routine. When the INTERCOMM checkpoint is taken, the checkpoint subsystem writes a checkpoint indicator record to INTERLOG. This record has a message number of zero and contains that time stamp of the checkpoint just taken in the message text portion of the record, plus a number of system statistics. To insure that this record is written to the log immediately, the LSYNCH=YES parameter should be coded in the Subsystem Control Table entry defining this CHCKPTSS.

The checkpoint subsystem completes processing by turning off the unschedulable bit it had turned on for update systems when quiescing activity. It also checks if there are any messages queued for those subsystems marked unschedulable. If messages are queued for any resident subsystem which is using the ECB Wait option, the corresponding ECB is posted to "wake up" the Subsystem Controller for that program. In addition, the SPACTIVE ECB is posted to activate the Subsystem Controller in the event that any new work has been queued for overlay programs while checkpointing was in progress. (The rescheduling of subsystems is performed by GDBSTUP when using the GDB support.)

The checkpoint subsystem performs analogous operations for file updating programs if the File Recovery Special Feature is being utilized.

3.4 ABEND PROCESSING

Special processing is necessary to insure data integrity if either INTERCOMM or the DBMS should abnormally terminate. When INTERCOMM and the DBMS execute in two seaprate regions, if either job goes down, the one remaining must also be shut down and restart procedures initiated. When either INTERCOMM or the DBMS abnormally terminates, it is necessary to close all DBMS data sets. To accomplish this, special procedures have been incorporated into INTERCOMM's STAEEXIT routine via the module DBSTAE. The DBMS should have comparable capabilities.

In particular, STAE routine processing is needed to inform the regions communicating with each other of the abnormal termination of any of the other regions. There are three considerations: the DBMS region, an INTERCOMM region and a batch region abending. If any of these jobs goes down, the remaining regions must also be cancelled and restart procedures initiated to maintain an up-to-date status of the data base.

The following procedures are typically followed. Each time a new region which is to access the data base is initiated, it locates the Data Base Management region to inform it of its presence. At this time, a block with two ECBs is defined for the new region. One of the ECBs is waited upon by the application region for notification by the DBMS of an abend condition; the other ECB should be waited upon by the DBMS for notification of the application programs abend.

The DBMS interface routine of the INTERCOMM region should contain an entry point to utilize the STAE routine (DBSTAE).

If supplied, INTERCOMM STAEEXIT calls DBSTAE after determining that the INTERCOMM task cannot be reinstated. This provides the user with the ability to notify the DBMS region that the INTERCOMM task has abended. The DBMS should, upon receiving notification of abnormal termination, purge all threads from INTERCOMM in the process of being serviced.

There are several important ramifications of this purging activity. First, it is important that any portion of the data base "locked out" be now freed up. Second, it is very necessary that the DBMS take no further action in the core owned by INTERCOMM (such as moving back data or posting) since the implication of an abend is that INTERCOMM will give up ownership of the core.

To secure data integrity of the DBMS region abends, there is a routine (DBMABEND) waiting for notification from the DBMS of an abend at all times. This routine, when it receives control, typically requests the operator via a WTOR to indicate whether INTERCOMM processing should continue without data base subsystems or if the region should be cancelled to initiate restart procedures. The details of abend processing for each DBMS may be found in subsequent sections.

DBMABEND is dispatched via an ECB by DBSTART when the DBMS is:

- Operating in a separate region
- Specified in the EXTR parameter of the ATTACH; i.e., operating as a subtask of INTERCOMM

A DBMS interface routine may additionally employ a routine where a check for a particular character string at a particular location is made at a frequent periodic interval. This strategy is intended to find failing DBMS user regions that were unable to execute STAEEXIT at failure time, an unlikely but possible condition. The need for such additional error precautions increases if the DBMS has interregion data-moving code within it. (INTERCOMM's Multiregion Facility uses this time-driven checking logic.)

3.5 RESTART/RECOVERY OFF-LINE UTILITIES

The INTERCOMM DBMS restart capability is based on backing up both systems to the last checkpoint and restarting all the processing which affects the data base initiated since then.

With utilization of the utilities discussed here and the INTERCOMM provided programs for checkpoint and restart, all data base and system failure conditions are foreseen. Possible

data base failures, for which a return to the prefailure state is prescribed by the restart/recovery procedures herein, are simply and effectively handled without extensive programming on the user's part.

For the on-line INTERCOMM region, all restart processing is supplied by the INTERCOMM DBMS Restart facility. The INTERCOMM/DBMS user must have the following DBMS utility programs:

- A data base dump/restore facility to yield a backup copy of the data base to be used for recovery when part or all of the data base is lost.

- A data base recovery utility which applies the after-images from a series of data base log tapes to an old (restored) copy of a data base. This utility will be used to bring a data base up-to-date when it has been lost and the last backup taken is not up-to-date.

- A data base backout (reversal) utility to be invoked when the system has failed during execution of a data base update program (INTERCOMM or a batch run). This utility removes any updates done to a data base up to the last checkpoint time or to the time specified by the operator.

In line with the ability to back out all updates performed on given data bases up to a given checkpoint, it is suggested that the Data Base Backout program make available the following options:

- The operator may specify a time which will cause this program to back out all updates to that checkpoint time.

- The program may arbitrarily pick the last valid checkpoint by reading the data base log backward.

- The program should be able to back out to the last INTERCOMM checkpoint taken.

- If an INTERCOMM startup record is placed on the data base log, it should be recognized as the last valid on-line checkpoint. This might be done when INTERCOMM performs the sign-on function to the DBMS. If the operator has selected a time previous to the INTERCOMM startup, the program should inform the operator that INTERCOMM backup is complete and continue to the checkpoint taken at the indicated time.

- The Data Base Backout program may have the ability of not backing out any updates by a specific job step, if update by job step identity is available in the data base log records. This facility could be of assistance to installations using the DBMS with batch programs that update data bases not being updated either by on-line system or by other batch programs. If an INTERCOMM restart must be done after such a batch data base update job has completed, the Data Base Backout Utility need not be rerun.

3.6 RESTART PROCESSING

As referenced previously, the INTERCOMM/DBMS restart capability involves:

- Use of backout facility to recover both systems from the time of a checkpoint (usually the last) until the failure condition.

- A restart of all processing which affected the data base during this backup time.

Prior to INTERCOMM restart, the DBMS is first backed out via the DBMS Backout Utility. Typically, the user may specify restoration to other than the last checkpoint by supplying the checkpoint time in the PARM field of the Backout Utility EXEC card or in response to a WTOR. In any case, the utility will inform the operator of the checkpoint time to which the data base was backed out.

In most cases, INTERCOMM restart may be executed concurrently with the backout utility. During restart processing, INTERCOMM calls the Restore routine to recreate the subsystem tables as they existed at the last checkpoint. The INTERCOMM program LOGPROC is then called to process the INTERCOMM log.

LOGPROC has the responsibility for selecting messages for restart. This is accomplished by reading the log tape backward until a checkpoint record is encountered. All messages processed by programs using the DBMS in update mode will be restarted unless indicated otherwise by the application. (A return code of 64(X'40') supplied by a data base processing program specifies a message should not be recycled at restart time. This override option applies only to those messages which completed since the last checkpoint.) Any messages which were in process at system failure time or were

on the queues at failure time are restarted under all circumstances. The following log codes (character) will be put by LOGPROC in the message headers of restarted messages:

P = message was completely processed but is being restarted for DB recreation purposes

R = message was in process at time of system failure

2 = message was on the queue (never started) at time of failure

NOTE: These log codes appear on INTERLOG as translated to hexadecimal values X'01', X'03', X'02', respectively.

When the last checkpoint message has been located, INTERCOMM (via the 'ENTER CHECKPOINT TIME' WTOR) requests from the operator the time to which the DBMS has backed up.

If the checkpoint ID supplied by the operator is not equal to the last INTERCOMM checkpoint then LOGPROC continues processing its log file backwards to the desired checkpoint. The technique of having LOGPROC immediately proceed to the first checkpoint before knowing which checkpoint is actually to be used is for speed purposes. Generally, the last checkpoint is always used. Thus, INTERCOMM is usually waiting to immediately proceed after the DBMS Backout Utility completes.

4. INSTALLING DBMS SUPPORT--GENERAL REQUIREMENTS

4.1 GENERAL

This section details specifications required to install the DBMS support which is common to all DBMS. DBMS support is considered primarily with preparation of the INTERCOMM system to execute in conjunction with a DBMS region. The following areas can be generalized:

- Conditional Assembly Specifications via SETGLOBE
- Preparation for Use of the Interregion SVC
- Coding of the System Parameter List (SPA)
- Coding of Subsystem Control Table (SCT) Entries
- Preparation for Checkpoint (CHEKPTFL)
- Execution Procedures--Normal Startup
- Execution Procedures--Restart/Recovery
- Subsystem Design Considerations

The INTERCOMM linkedit will include DBMS interface modules. Unless the DBMS is operating as an INTERCOMM subtask or the DBMS requires a workfile in the INTERCOMM region, execution JCL for the INTERCOMM Job Step is standard and requires no additional consideration for use of the DBMS.

Additional DBMS-dependent table considerations, conditional assemblies, linkedit and JCL requirements are described in subsequent sections along with pertinent detail for application programming functions.

4.2 CONDITIONAL ASSEMBLY SPECIFICATIONS

Two members of the INTERCOMM Release Library PMI.SYMREL are utilized for conditional assembly specifications:

- INTGLOBE defines system globals
- SETGLOBE specifies system globals

These two members control assembly of the System Parameter List (SPA CSECT) and SPA Extension (SPAEXT CSECT) and many INTERCOMM system routines. Certain globals control specifications for DBMS-dependent modules. INTGLOBE and SETGLOBE, as released, are illustrated in Figures 4-1 and 4-2. Certain of these globals also control assembly of DBMS-dependent modules (discussed in subsequent sections).

Typical JCL for creating a tailored SETGLOBE for a particular installation is illustrated here.

```
//EXEC  LIBE,Q=LIB,NAME=SETGLOBE
//SYSIN DD *
./ REPL NAME=SETGLOBE
./ NUMBER NEW 1=00000100,INCR=00000100
./
     :        installation-dependent settings for Data Base
              Support, Resource Management, Multiregion
              Support Facility, DDQ, etc.
```

4.3 PREPARATION FOR INTERREGION COMMUNICATION

The following steps must be performed to prepare for interregion communication:

- Create or identify an available Type I or II user SVC number. (The choice of type varies with DBMS. A Type I is more efficient and should be used if possible. Refer to DBMS-dependent sections of this manual.)

- Update the member SETGLOBE to specify the SVC number via &INTSVC SECTC 'nnn'. Update the member SETGLOBE to specify a VS2 operating system via &VS2 SETB 1.

- Reassemble and linkedit the member IGC250 and add it to the OS/VS nucleus. Linkedit parameters must be: LIST, LET, DC, REUS.

This SVC is not required under the following conditions:

- For TOTAL support, the module IGC250 (and the associated Type II SVC identified by &INTSVC) is required only if the INTERCOMM region is to coordinate checkpoints with batch regions updating the on-line data base. (See Section 7.)

- For ADABAS support, interregion communication is not accomplished via IGC250 but through Type III SVC provided by Software A.G. (See Section 8.)

```
GBLB  &MULTASK            OBSOLETE GLOBAL
GBLA  &LOGRECL            OBSOLETE GLOBAL
GBLB  &SPAEXT             OBSOLETE GLOBAL
      GENERAL SYSTEM FEATURES:
GBLB  &VSSYSTM            ON IF RUNNING UNDER VS1 OR VS2
GBLB  &SYS370             USE OF /370 INSTRUCTION SET
GBLC  &USRHI              HEX UPPER LIMIT FOR USER LOG CODES
GBLC  &USRLO              HEX LOWER LIMIT FOR USER LOG CODES
GBLA  &LOGINIM            LOG INPUT INTERVAL
GBLC  &INTSVC             INTERREGION SVC USED BY INTERCOMM
GBLB  &VS2                FOR VS2 PROT
      RESOURCE MANAGEMENT
GBLB  &RM                 RESOURCE AUDITING
GBLB  &RMSTATS            RM STATISTICS GATHERING.
GBLB  &RMACCT             BUCKET ACCOUNTING SWITCH.
GBLB  &RMPOOLS            SUPPORT USER POOLS.
GBLB  &RMINTEG            RESOURCE MGMNI CORE INTEGRITY CHCK.
      MESSAGE RESTART/RECOVERY:
GBLA  &FEMSGHI            FRONT END HI VALUE MSG ACCTG
GBLA  &BEMSGHI            BACK END HI VALUE MSG ACCTG
GBLA  &LOSPRCQ            LOGPROC Q ELEMENTS VI   ONLY
      DISPATCHER:
GBLA  &NUMWQES            NUMBER OF WORK QUEUE ELEMENTS
      FILE HANDLER:
GBLA  &RPTINTV            FILE STATISTICS REPORT INTERVAL
GBLB  &ISAM               ISAM FILES USED
GBLB  &AMIGOS             AMIGOS FILES USED
GBLB  &VSAM               VSAM FILES USED
GBLB  &VISAM              ISAM/VSAM COMPATIBILITY REQUIRED
      EDIT UTILITY:
GBLB  &DELCHNG            NO CORRECT/CHANGE FACILITY USED
GBLB  &EDERRS             NO MAXIMUM FOR EDIT ERRORS SENT
GBLA  &EDERMAX            MAXIMUM NUMBER OF EDIT ERRORS
                          (USED ONLY IF &EDERRS=0)
GBLB  &UPTRPT             SEND ERRORS FOR OPTIONAL PARMS
      OUTPUT UTILITY:
GBLB  &QTAM               FRONT END IS QTAM
GBLB  &TCAM               FRONT END IS TCAM ONLY
GBLB  &DDQBACK            DYNAMIC DATA Q'S - AUTO INPUT
GBLB  &BROAD              NO BROADCAST GROUP
GBLB  &RPIBLE             NO REPORTS TO TAPE
GBLB  &ALTRPT             NO ALTERNATE REPORTS
GBLB  &OUTEXIT            NO USER OUTPUT EXIT
GBLB  &ICAMOUT            TCAM OUTPUT-ONLY STATIONS IN USE
GBLB  &IASYN              TCAM OUTPUT-ONLY I/O DONE BY
                          A SUBTASK (TCAMASYN)
      DL/I SUPPORT:
GBLB  &DLI                DL/I
GBLC  PMAXTASK,&MASRES    OBSOLETE GLOBAL
GBLA  &MAXRESA            OBSOLETE GLOBAL
GBLB  &PSIREDU            OBSOLETE GLOBAL
      TOTAL SUPPORT:
GBLC  &IOTESC             TOTAL DATA BASE DESCRIPTOR
GBLC  &IOTSVC             TOTAL SVC NUMBER
      RJE FACILITY:
GBLB  &RJEWTO             INTR RJE TO INFORM OP OF EACH JOB
GBLC  &RJECLSA            OUTPUT CLASS TRANSFORMATION FOR CL A
GBLC  &RJECLSH            OUTPUT CLASS TRANSFORMATION FOR CL B
GBLB  &AUTORDR            ON IF RJE IS TO AUTO START A RDR
GBLC  &RDRNAME            READER NAME TO BE USED
GBLC  &RDRID              RJE RDR ID-('.5','.P2',ETC.)
GBLA  &NUMJOBS            JOB THRESHOLD FOR AUTO START
      MULTIREGION SUPPORT:
GBLB  &MULTREG            MULTI-REGION SUPPORT REQUESTED
GBLC  &MRSVC              MULTI-REGION SUPPORT SVC
      MODEL SYSTEMS GENERATION:
GBLC  &GENTERM            TERMINAL IDENTIFICATION
```

Figure 4-1. INTGLOBE Example

```
&MULTASK SETB  1                    OBSOLETE, REQUIRED SETTING
&LOGRECL SETA  500                  OBSOLETE, REQUIRED SETTING
               GENERAL SYSTEM FEATURES:
&VSSYSTM SETB  1                    DEFAULT TO VS
&SYS370  SETB  0                    DO NOT USE /370 INSTRUCTIONS
&USRHI   SETC  '6F'                 USER LOGCODE - HIGH VALUE
&USRLO   SETC  '40'                 USER LOGCODE - LOW VALUE
&LOGINTM SETA  3                    .3 SEC TO DISP LOGINPUT
&INTSVC  SETC  '013'                DEFAULT INTERCOMM INTER-REG SVC NUM
&VS2     SETB  0                    DEFAULT TO NOT VS2
               RESOURCE MANAGEMENT
&RM      SETB  1                    RESOURCE MANAGEMENT
&RMSTATS SETB  1                           STATISTICS
&RMACCT  SETB  1                           ACCOUNTING
&RMPOOLS SETB  1                           CORE POOLS
&RMINTEG SETB  0                            CORE POOL INTEGRITY CHECK
               MESSAGE RESTART/RECOVERY:
&FEMSGHI SETA  255                  MESSAGE ACCOUNTING - HIGH
&BEMSGHI SETA  255                  MESSAGE ACCOUNTING - LOW
&LOGPRCQ SETA  100                  VI + ONLY
               DISPATCHER:
&NUMWQES SETA  120                  NUMBER OF WORK QUEUE ELEMENTS
               FILE HANDLER:
&RPTINTV SETA  600*300              600 SECS = 10 MINS
&ISAM    SETB  1                    ISAM FILES USED
&AMIGOS  SETB  0                    AMIGOS FILES NOT USED
&ISAM    SETB  (&ISAM OR &AMIGOS)   AMIGOS REQUIRES ISAM
&VSISAM  SETB  1                    ISAM/VSAM COMPATIBILITY
&VSAM    SETB  0                    VSAM FILES NOT USED
&VSAM    SETB  (&VSAM OR&VSISAM)    FORCE VSAM IF ISAM/VSAM IS USED
               EDIT UTILITY:
&DELCHNG SETB  1                    NO CORRECT/CHANGE FACILITY
&EDERRS  SETB  0                SEND NO MORE THAN &EDERMAX ERRORS/MSG
&EDERMAX SETA  5                    MAXIMUM NUMBER OF ERRORS/MESSAGE
&OPTRPT  SETB  0                SUPPRESS ERROR MSG IF PARM IS OPTIONAL
               OUTPUT UTILITY:
&QTAM    SETB  0                    FRONT END IS NOT QTAM
&TCAM    SETB  0                    FRONT END IS NOT TCAM-ONLY
&DDQBACK SETB  0                    DEFAULT TO NO DDQ AUTO INPUT
&BROAD   SETB  0                    BROADCAST GROUPS IN USE
&RPTBLE  SETB  0                    REPORTS TO TAPE IN USE
&ALTRPT  SETB  0                    ALTERNATE REPORTS IN USE
&OUTEXIT SETB  1                    NO USER OUTPUT EXIT
&TCAMOUT SETB  0                TCAM OUTPUT-ONLY STATIONS NOT USED
               DL/I SUPPORT:
&DLI     SETB  1                    DL/I IN USE
               TOTAL SUPPORT:
&TOTDESC SETC  'XXXXXX'             TOTAL DATA BASE DESCRIPTOR
&TOTSVC  SETC  'NUL'                NO INTERREGION COMM NECESSARY
               RJE FACILITY:
&RJECLSA SETC  'M'                  DEFAULT TRANFRMATION FOR CLASS A
&RJECLSB SETC  'N'                  DEFAULT TRANSFORMATION FOR CLASS B
&RJEWTO  SETB  1                    DEFAULT
&RDRNAME SETC  'RJERDR'             DEFAULT
&RDRID   SETC  '.S'                 DEFAULT
&NUMJOBS SETA  10                   DEFAULT
               MULTIREGION SUPPORT:
&MRSVC   SETC  '013'                MULTI-REGION SVC NOT PRESENT
&MULTREG SETB  1                    MULTI-REGION SUPPORT REQUESTED
               MODEL SYSTEMS GENERATOR:
&GENTERM SETC  '$$$$$'
```

Figure 4-2. SETGLOBE Example

- IDMS (see Section 9).
- For Model 204, interregion communication is through a Type IV SVC provided by CCA as of release 3.16. (See Section 10.)
- System 2000 (see Section 11).

4.4 CODING THE SYSTEM PARAMETER LIST

The System Parameter List (SPA CSECT) and SPA Extension (SPAEXT CSECT) are coded via the INTERCOMM-supplied SPALIST macro, described in detail in the INTERCOMM System Macros Manual. Specifications pertinent to DBMS interface are:

CHKPTLIM=nn	*coordinates with CHCKPTSS SYCTTBL TCTV value.*
TCHP=nnn	*specifies estimated checkpoint time interval, in seconds. Default is 120.*
TOTATT= YES / NO	*for TOTAL only; specifies whether or not TOTAL is to be attached as an INTERCOMM subtask. Default is YES.*
GENSW=OO	*permits checkpointing onto CHEKPTFL DD file.*

Typical JCL to assemble and linkedit is illustrated in Figure 4-3.

4.5 CODING SUBSYSTEM CONTROL TABLE ENTRIES

In order to inform INTERCOMM that a subsystem accesses a data base, the Subsystem Control Table (SCT) entry for each such subsystem must specify whether it performs inquiries only or data base updates. The SCT is generated via the INTERCOMM-supplied SYCTTBL macro, documented in detail in the System Macros Manual. The operand specifying DBMS requirements is:

DBASE= { DB / UDB }	*where DB signifies inquiry-only access; UDB signifies updates performed by the subsystem. No data base access is specified by omitting this operand.*
	(TOTAL, UTOTAL, DLI, UDLI, ADA, UADA are also allowable specification for the DBASE= operand as described in the System Macros Manual.

Other operands define logging requirements and message restart requirements. This macro must be coded carefully in conjunction with study of Sections 3 and 12 in the Operating Reference Manual (ORM).

```
//SPA EXEC LIBELINK,Q=LIB,NAME=DBSPA,LMOD=DBSPA
//LIB.SYSIN DD *
./ ADD NAME=DBSPA
./ NUMBER NEW1=100,INCR=100
**SYSTEM PARAMETER LIST
SPA CSECT
 SPALIST TCHP=120,              Checkpoint interval
         TOTATT=NO,             No attached TOTAL
         GENSW=00               CHECKPTFL Spec.
         :                      Other operands as required

**SUBSYSTEM CONTROL TABLE ENTRIES
         SYCTTBL
         .
         .                      Entries as required for
         .                      INTERCOMM and user subsystems
         SYCTTBL
         .
         .
         .
         SYCTTBL
         .
         .
         .
         GENINDEX               Generate SCT index
         END
//SPAEXT EXEC LIBELINK,Q=LIB,NAME=DBSPAEXT,LMOD=DBSPAEXT
//LIB.SYSIN DD *
./ ADD NAME=DBSPAEXT
./ NUMBER NEW1=100,INCR=100
**SYSTEM PARAMETER LIST EXTENSION
SPAEXT CSECT
 SPALIST EXTONLY=YES,           Extension only
         TCHP=120,
         TOTATT=NO,
         .                      Identical operands
         .                      as above
         .
END
```

Figure 4-3. SPA and SPAEXT Creation

If INTERCOMM DBMS Restart/Recovery procedures are to be invoked, the checkpoint subsystem, CHCKPTSS, must be defined by a Subsystem Control Table entry as illustrated in Figure 4-4.

```
[SYMBOL]  SYCTTBL  SUBC=Q,          SUBSYSTEM CODE ENTRY POINT
                   SBSP=CHCKPTSS,
                   LANG=NBAL,       NON-REENTRANT BAL
                   MNCL=1,            SINGLE-THREAD
                   NUMCL=1,           CORE QUEUE
                   LSYNCH=YES,      FORCE IMMEDIATE LOG ENTRY
                   RESTART=NO,      NO RESTART
                   TCTV=nnn,        MAXIMUM TIME OUT
                   OVLY=n,          NONRESIDENT OR RESIDENT
                   PRTY=0,          HIGHEST PRIORITY
                   .
                   .                OTHER USER-SPECIFIED PARAMETERS
                   .                AS DESIRED
```

Figure 4-4. Checkpoint Subsystem SCT

The checkpoint subsystem may be resident or nonresident at the user's option, using the following criteria:

- If resident, allows non-update resident or overlay subsystems to continue processing with maximum concurrency while update subsystems are quiesced.

- If assigned to Overlay Region A, the time to quiesce update activity may be minimized. All other Overlay Region A Subsystems will be inactive during the checkpoint processing interval. Generally, no other subsystem should be included in the same overlay as CHCKPTSS. Thus, only currently active resident or any dynamically loaded update subsystems will delay the checkpoint subsystem, CHCKPTSS.

In other words, the time to quiesce update activity to perform a checkpoint is dependent upon activity in all subsystems in core and active at the time checkpoint preparation begins.

4.6 PREPARATION FOR CHECKPOINT

A BDAM data set is used by the INTERCOMM checkpoint routines and is required if data base recovery is to be performed. Checkpoint data, consisting of pertinent table information, is written to this data set. In addition, an entry is made at checkpoint time on the INTERCOMM log signifying "checkpoint synchronization with a DBMS."

The checkpoint data set (ddname CHEKPTFL) must be preformatted prior to execution via the Intercomm off-line utility, CREATEGF. The minimum allowable specification is 50 blocks of 56 bytes. (A block size of at lease 200 bytes is recommended.) Please consult the Operating Reference Manual, to compute the actual number of blocks and most efficient block size for your installation's use.

The DD statement for CHEKPTFL must be contained in the Intercomm region JCL. Omission of this DD statement will effectively cause checkpoint processing to be bypassed. The computer operator will be notified of this no-checkpointing condition but unless he takes action the system will continue processing. The user can ensure that a no-checkpoint situation never occurs by adding code in USRSTRT1 which selects CHEKPTFL and abends on a nonzero return code.

4.7 EXECUTION PROCEDURES--COLD STARTUP

The typical sequence of events for system startup with Intercomm and a DBMS executing in separate regions is as follows:

1. Initiate the DBMS region.
2. Initiate the Intercomm region in startup mode.
3. Initiate any required batch programs.

Should Intercomm be initiated prior to the DBMS region, the console operator will typically receive a WTOR indicating the DBMS region is not active. Three responses are possible:

- Cancel DBMS subsystems (those marked by DBASE in their SCT) and allow other subsystems to continue operation.
- Abend Intercomm region.
- Retry DBMS startup functions (presumably the operator will start the DBMS before he responds).

Should a batch program be initiated prior to the DBMS region, a similar WTOR may be issued with possible replies of ABEND or RETRY. See Messages and Codes for specific WTORs for each DBMS.

Of course, when the DBMS is attached, initialization of both Intercomm and the DBMS proceeds concurrently.

4.8 EXECUTION PROCEDURES--WARM STARTUP

An alternative to the foregoing involves a warm startup in which no file or DB reversal is performed, nor are any messages reprocessed. However, INTERCOMM will analyze the previous run's log to recover input and output queued messages existing at last closedown. The procedures for this warm start mode are identical to the above excepting that INTERCOMM is initiated in restart mode.

4.9 EXECUTION PROCEDURES--RESTART/RECOVERY

In the event of an abend condition in either the DBMS, INTERCOMM or batch region(s) performing DBMS updates, an INTERCOMM restart must be performed. Restart of the INTERCOMM system generally necessitates backing out all data base updates to the time of the last coordinated INTERCOMM/DBMS checkpoint.

The following steps lead to system restart in the event of INTERCOMM abnormal termination:

A. Terminate batch jobs using the DBMS.

B. Execute the DBMS Backout Utility.

C. Execute INTERCOMM in restart mode.

D. Reinitiate batch processing.

Similarly, if the DBMS terminates abnormally, the following steps lead to system restart:

A. Close down INTERCOMM (NRCD or IMCD).

B. Terminate batch jobs using the DBMS.

C. Execute the DBMS Backout Utility.

D. Initiate the DBMS region (in some cases, may be done before C).

E. Execute INTERCOMM in restart mode.

F. Initiate any further batch processing.

If a batch region performing updates to on-line files terminates abnormally, these same procedures must be followed. Both the INTERCOMM and DBMS regions must be restarted, following execution of the Data Base Backout Utility.

4.10 SUBSYSTEM DESIGN CONSIDERATIONS

Application subsystems executing under INTERCOMM and using a DBMS must consider the amount of time required for file I/O. That this is often longer than for standard access methods should be accounted for when designing the number of I/O activities required.

Because of the requirements to quiesce all data base activity for checkpoint purposes, care should be exercised in planning the amount of I/O in any one program. If I/O in one program is excessive then, in checkpointing, all other programs remain quiesced waiting for this long running update program to complete. This may cause periodic serious degradations in response time. Also, pay particular attention to mixing large inquiry actions in the same program which does updates. If the SYCTTBL entry indicates update potential then the checkpoint subsystem will wait out this program regardless of whether or not it is actually doing updates. Long, sequential searches through various segments of the data base may slow down the entire system. Where possible, data base programs should be designed to be of short duration relative to other processing in the environment.

If different subsystems use the same data base, they may be accessing the same record. Thus, contention will be minimized if these subsystems are assigned to different overlay groups. Use of the RESOURCE macro with SYCTTBL macros is another consideration when planning active subsystem concurrency. Although desirable, this is not required, however, as the system will protect against concurrent use of the same data base by two or more subsystems. Subsystems using different data bases may perform more efficiently when assigned to the same overlay group. Subsystems that use the CONVERSE feature may only update a data base during processing of the last message of a conversation. The SYCTTBL entry for such a program must specify DBASE=UDB and CNVREST=YES. This insures that a checkpoint is not taken during its processing and that the conversation is restarted from the beginning. If CNVREST=NO is specified, the conversational program may not do data base updates directly. It may, however, send a message to another subsystem while processing the last message of a conversation.

A suggested good programming technique is as follows. The processing of any input transaction that will be processed by a number of different subsystems should be designed with all its update operations centralized in one subsystem. This will keep application restart logic to a minimum. It will also not slow checkpoint if some programs in this processing chain do inquiries, even lengthy ones.

A subsystem can identify messages that are being sent to it during recovery for INTERCOMM restart via the log code field (MSGHLOG) of the incoming message header as follows:

- C'2'

  Normal message; not previously processed.

- C'P'

  Restarted message; being reprocessed for data base recovery even though message processing had completed in previous run. Subsystem logic may bypass generation of response message(s) to terminals(s).

- C'R'

  Restarted message; had not completed its processing in the previous execution. Subsystem logic typically can not determine the extent of processing completed in the last run; thus, it is best to generate response message(s) to terminal(s) even though they may be duplicates.

Generally, the subsystems need not be concerned with any aspect of recovery. They can ignore these log codes and process the message as any other. Other subsystem design considerations concerned with message restart/data base recovery are:

- By using a Subsystem Controller return code of 64 (X'40'), the application subsystem specifies to INTERCOMM that a message processed by a DBMS update subsystem is never to be restarted. This could be used when subsystem logic performs only inquiries even though the SYCTTBL specifies an update subsystem; some messages processed may not update the data base. Inadvertent use of this return code can impair data integrity.

- By using a Subsystem Controller return code of 68 (X'44'), an application subsystem specifies to INTERCOMM that a message processed by a DBMS inquiry-only subsystem is to be restarted if processed since the last checkpoint.

5.1 INTRODUCTION

The INTERCOMM Generalized Data Base Management System Interface (GDB) consists of a series of programs which supply all but the specific data base logic necessary to provide an integrated DB/DC environment. GDB support logic is used to supply data base access from multiple regions (partitions) while providing data integrity across program and system failure. GDB support is not available for MVS.

Although most application programs will be run under control of the INTERCOMM monitor, programs not under INTERCOMM's control (i.e., batch programs) may require concurrent and/or overlapping use of the DBMS. Thus, INTERCOMM's interfaces have been generalized to provide for utilization of the DBMS by batch programs as well as on-line INTERCOMM programs.

The DBMS in use must meet the following requirements in order to be supported by the INTERCOMM system:

- Ability to operate as a separate task from those application programs utilizing it. This may be either as a subtask of INTERCOMM or as a main task in its own partition or region.

- Ability to force out buffers and store all position-relevant data in control blocks for possible future use during a restart.

- A logging technique must be provided to record each time an update, insert or delete to the data base is performed. Before- and after-images of data base records should be logged when the data base is altered in any way.

- To avoid loss of INTERCOMM updates, as well as those of batch programs operating concurrently in separate regions, an exclusive control facility must be present in the Data Base Management System.

- The level of exclusive control at the data base file or record level is dependent on the design of the Data Base Management System. Any restrictions on the ability to execute multiple update programs concurrently will be dependent on the level of the lockout which is possible.

With the above requirements met, the implementation of a number of interface routines to coordinate data base requests across regions is all that is required of the GDB user.

5.2 SUPPORT MODULES

The INTERCOMM GDB facility consists of the following requirements (as summarized in Figures 5-1 and 5-2):

- An INTERCOMM-supplied Type II SVC routine (IGC250) which provides for interregion communication.
- User-provided routines called by INTERCOMM to provide for the following:
  - Initialization processing (DBSTART)
  - Closedown processing (DBCLOSE)
  - Data base request handling (DBINT, DBRELEX, DBPURGE)
  - Data base checkpoint processing (DBCKPREP and DBCHKCOM)
- INTERCOMM-supplied routines which provide the following functions:
  - Initialization and continuation of interregion checkpoint (GDBSTUP, DBCHKDSP)
  - Quiescing of data base activity within INTERCOMM (CHCKPTSS)
  - Multitasking capability within the DBMS region (IJKDSP02)
  - Message restart processing (DBRSTRT, LOGPROC)
- Other suggested user-supplied routines:
  - Abend processing (DBMABEND, DBSTAE, DBMSTAE)
  - DBMS region checkpoint processing (DBMCHECK)
  - Batch region checkpoint requests

The previous discussion of conceptual system logic in Sections 2 and 3 relates directly to GDB support. In this section, a brief summary of implementation requirements where applicable follows recommendations for user routine logic.

The user has the option of providing all interface modules as CSECTs or entry points in one load module for the INTERCOMM region and one load module for the DBMS region, or separate load modules for each of the required functions. The decision is based upon ease of implementation considering the program structure of the DBMS to be supported.

Function	Member	CSECT or Entry Point	Residency
Interregion Type II SVC Routine	IGC250	IGCnnn	Nucleus: nnn is the user-assigned SVC number
Startup Processing	username	DBSTART	Startup Overlay
Data Base Request Handling	username username username	DBINT DBRELEX DBPURGE	Resident Resident Resident
Closedown Processing	username	DBCLOSE	Closedown Overlay
Abend Processing	username username	DBMABEND DBSTAE	Resident Resident
Checkpoint Processing*	username* CHCKPTSS* username* GDBSTUP* DBCHKDSP*	DBCKPREP CHCKPTSS DBCHKCOM GDBSTUP DBCHKDSP	Transient Overlay User-Assigned Resident Resident Resident
Restart Processing*	DBRSTRT* LOGPROC* READBACK* INTDBLOK*	DBRSTRT LOGPROC READBACK INTDBLOK	Startup Overlay Startup Overlay Startup Overlay Startup Overlay

* Required only for restart when on-line updates performed.

Figure 5-1. Interface Modules INTERCOMM Region

Function	Member	CSECT
Startup Processing	username	username
Data Base Request Handling	IJKDSP02	IJKDSP02
Closedown Processing	username	username
Checkpoint Processing	username	DBMCHECK
Abend Processing (STAE routine)	username	DBMSTAE
Restart Processing	username	username

Figure 5-2. Interface Modules——DBMS Region

5.3 DESIGN OF GDB SUPPORT MODULES

The INTERCOMM GDB support provides for user exits at key points in control of DBMS access from application programs. The conceptual logic and functional relationship of all programs involved in DBMS support is presented in Sections 2 and 3.

User-provided interface programs for a user DBMS should be designed to include facilities for:

- Interregion or intertask communication between the DBMS and INTERCOMM. The objective of such communication is a synchronized path between two asynchronously executing programs. This path is normally called a software channel. One program notifies the other when it requires attention and then waits to be recognized. After a request has been passed, the two activities continue asynchronously processing. Thus, the software channel usually "waits" until notification of a completion of one request.

- Transfer of data from the DBMS region to the INTERCOMM (or batch) region.

- Multithreaded operation of the DBMS region or at least a multithreaded interface that can stack requests for serial processing.

Two INTERCOMM-supplied routines may be utilized as service routines to assist in providing the above facilities: IGC250, a Type II SVC routine; and IJKDSP02, a program used in conjunction with the standard INTERCOMM Dispatcher (IJKDSP01) to effect multithreaded operation of the DBMS region.

Extensive use of the INTERCOMM DISPATCH macro is a requirement of the GDB Interface routine programmer. The macro format is illustrated in Figure 5-3. Please refer to the INTERCOMM Basic System Macros for further detail on this and other macros available to the INTERCOMM user. The remaining discussions in this section assume understanding of the DISPATCH macro for task scheduling, waiting for event completion or performing a time delay.

[symbol]	DISPATCH	for an execution, event, or timer queue request: {priority/overlay / 'S' / (r)} , {address / (r) / (0)} , {parameter / (r) / (1) / (13)} [,EXIT] [{ ,INTVL= {interval / (r)} / ,ECB= {event-control-block / (r)} }] [,LINK=(14)] [,SYS={YES / NO}] [,INTRNL={YES / NO}] for a cancellation request: [,EXIT,]CANCEL={wqe-address / (r)} [,LINK=(14)] for a termination request: EXIT [,LINK=(14)]

Figure 5-3. The DISPATCH Macro
(See Basic System Macros)

A typical method of accomplishing interregion communication is to define a pair of ECBs used as a communication channel between the two regions. Posting of one of the ECBs by the DBMS region serves to notify INTERCOMM that a DBMS activity is complete; posting of the second ECB by the INTERCOMM region indicates to the DBMS region that an INTERCOMM activity is complete. In this way, synchronization of the two regions can be accomplished. The ECB channel must be resident in the INTERCOMM region because the DISPATCH macro will be used to wait on the ECBs and the Dispatcher (resident in the INTERCOMM region) allows ECBs only in its own region. The DBMS region may accomplish wait and post functions for ECBs not in its region via the interregion SVC routine IGC250 or via special use of IJKDSP02 described below.

Required ECB channels for INTERCOMM programs are contained in the INTERCOMM SPA Extension (SEXT0ECB, SEXT2ECB). Any ECB channels required by user GDB Interface routines might be defined in the User Extension to the System Parameter List (USERSPA) or self-contained in interface programs. A user-defined table might be used for subsystem or thread-related ECBs when a variable number of channels is required.

Transfer of data from the DBMS region to the INTERCOMM region (i.e., the results of data base access being moved to an application program work area) may also be accomplished via the program IGC250. In this instance, its function is to allow operation in protect key zero.

Multithreading in the DBMS region may be accomplished by the use of the DISPATCH macro to perform task scheduling and control. All DBMS programs must of course be reentrant if DISPATCH is used to achieve multithreading. All operating system WAIT operations will be replaced by DISPATCH macros.

To make use of the Dispatcher in a multiregion mode, there is a special macro which may be issued when the Data Base Management System region comes up to signal the Dispatcher that interregion ECB waits will be executed.

This special macro is coded as:

```
DISPSET  OPTION=CROSSREG
```

The module IJKDSP02 contains a CSECT 'EXECWAIT' which is a replacement for the CSECT of the same name in IJKDSP01. The function of IJKDSP02 is to issue an interregion wait via the interregion Type II SVC rather than standard OS/VS WAIT.

5.4 INTERCOMM SVC ROUTINE (IGC250)

The member IGC250 on PMI.SYMREL is a Type II SVC routine which provides for interregion communications or general use in protect key zero. Before it may be used, the following steps must be taken.

A. The OS SYSGEN must have an available SVC number (Type II).

B. The Global &INTSVC must be set in the member SETGLOBE to SVC number which is to be assigned to IGC250. For example, SVC number 247 is available. The following code should be added to SETGLOBE:

```
&INTSVC SETC '247'
```

If operating with VS2, the &VS2 global should be SETB'd to 1 in SETGLOBE:

```
&VS2 SETB 1
```

C. IGC250 should then be assembled. Its CSECT name will be IGC247 after following the above example. (The released CSECT name is IGC&INTSVC).

NOTE: (Member SPAEXT must be re-assembled to incorporate &INTSVC.)

D. A re-linkedit of the OS nucleus must then be done including this new SVC routine.

This user SVC does the following:

- Posts an ECB in another region.
- Waits on an ECB in another region.
- Gives control to a user-supplied routine which must execute in protect key 0.

The input parameters are described in Figure 5-4.

A parameter list for the user routine may be passed to IGC250 by coding it immediately after the 'INTR' password DC statement. On entry to the user routine Register 0 still points to 'INTR'. The routine which executes must return to the address in Register 14. Its base is Register 15.

Register 0: always contains the address of a 4-character password 'INTR'		
Register 1: must contain one of the following:		
	BITS 0-7	BITS 8-31
WAIT routine	X'80'	ECB address
	X'C0'	ECB list address
POST routine	X'41' indicates a post code is supplied in the fullword following 'INTR' password. X'40' indicates no post code is supplied.	ECB address
user routine	X'00'	routine address

Figure 5-4. IGC250 Parameters

5.5 STARTUP PROCESSING (DBSTART)

The user must supply a module with CSECT name DBSTART to perform any initialization processing necessary in the Data Base Management System. This module will be executed when INTERCOMM is started. This CSECT may reside in the INTERCOMM startup overlay region. The functions that typically should be included in this program are illustrated in Figure 5-5 and are:

(1) If the Data Base Management System is to operate as a subtask of INTERCOMM, an attach of the DBMS should be done.

(2) The existence of the Data Base Management System, if it resides in another region, should be determined.

(3) A communication path between INTERCOMM and the Data Base Management System should be established. The DBMS, upon receiving notice that INTERCOMM is operational, should perform any initialization that may be necessary in its region (i.e., loading control blocks, logging a startup record on, etc.).

(4) Notification by the Data Base Management System to INTERCOMM of an abend of the DBMS region should be provided for at this time.

(5) If interregion checkpoint synchronization is neces: (data base integrity) the routine GDBSTUP must be

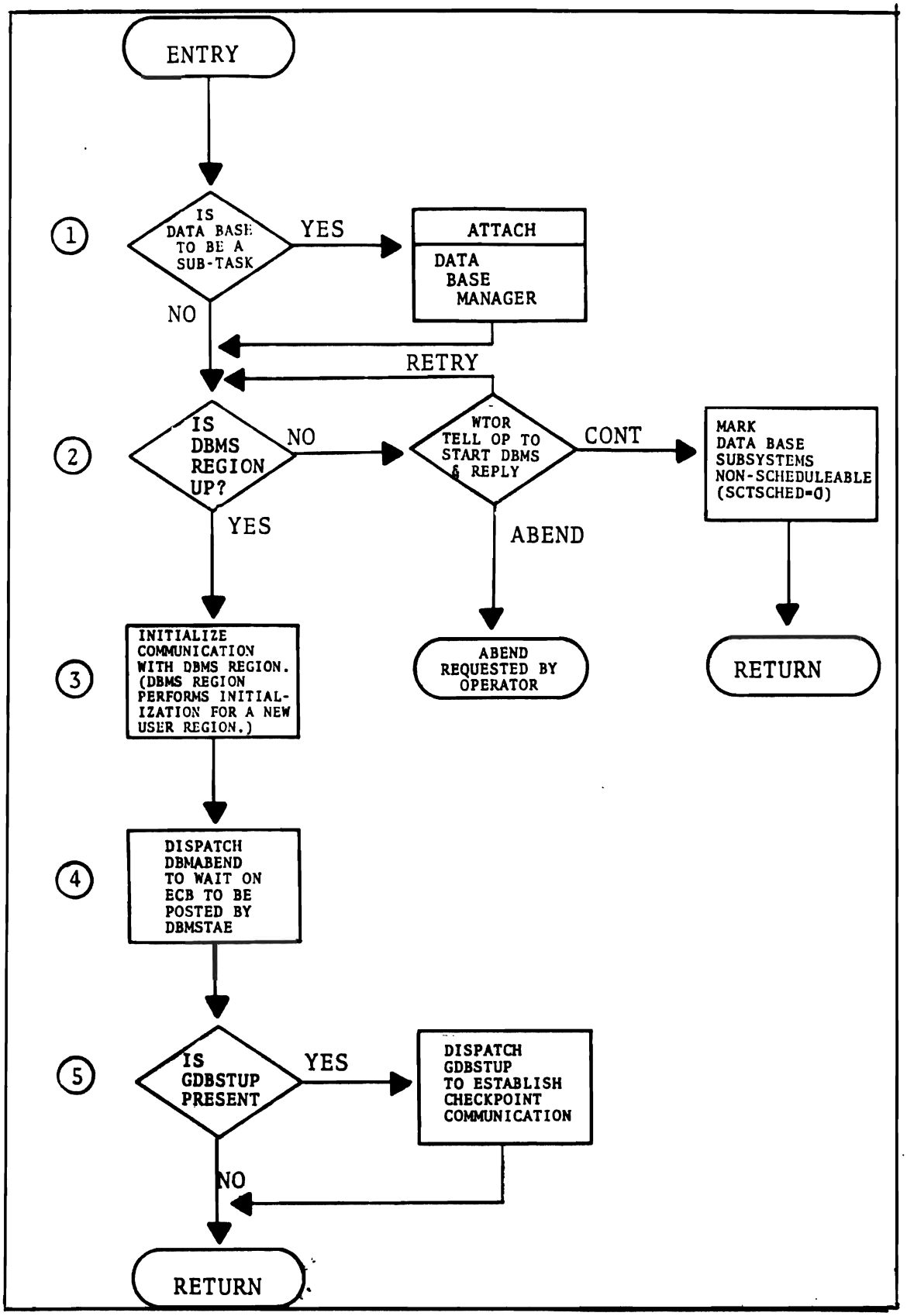

Figure 5-5. Suggested DBSTART Logic. (Encircled numbers correspond to list on previous page).

dispatched to initialize interregion checkpoint control. To accomplish this dispatch, DBSTART should include code as follows:

```
        L         R0,=V(GDBSTUP)
        LTR       R0,R0
        BZ        NODISP
        DISPATCH  'S',(0),(1),SYS=YES
NODISP  DS        0H
```

(Register 1 must be specified in the macro but is not relevant in this example.)

The Data Base Management System, upon receiving a sign-on of INTERCOMM through DBSTART logic, should make appropriate notation of INTERCOMM startup on its log. This will provide the data base backout program with information necessary for processing.

5.6 CHECKPOINT INITIALIZATION (GDBSTUP)

The INTERCOMM-supplied resident program GDBSTUP, if included in the linkedit of INTERCOMM, should be dispatched by the DBSTART program if restart/recovery procedures are to be used. This program establishes a path of communication between itself and the DBMS through the use of a pair of ECBs in the SPA Extension (SEXT2ECB and SECT0ECB). As illustrated by Figure 5-6, GDBSTUP performs the following functions:

(1) At initialization time GDBSTUP posts SEXT2ECB signifying it is ready to accept requests and dispatches itself waiting on SEXT0ECB waiting for a checkpoint request from the DBMS (the module DBMCHECK or a similar module).

(2) When a checkpoint request is received, it then formats a message for the checkpoint subsystem and queues it via Message Collection.

(3) GDBSTUP waits for an "end checkpoint command" from the DBMS. When it receives this command (code of 12 posted in SEXT0ECB), it sets schedulable those subsystems previously marked nonschedulable by the checkpoint subsystem.

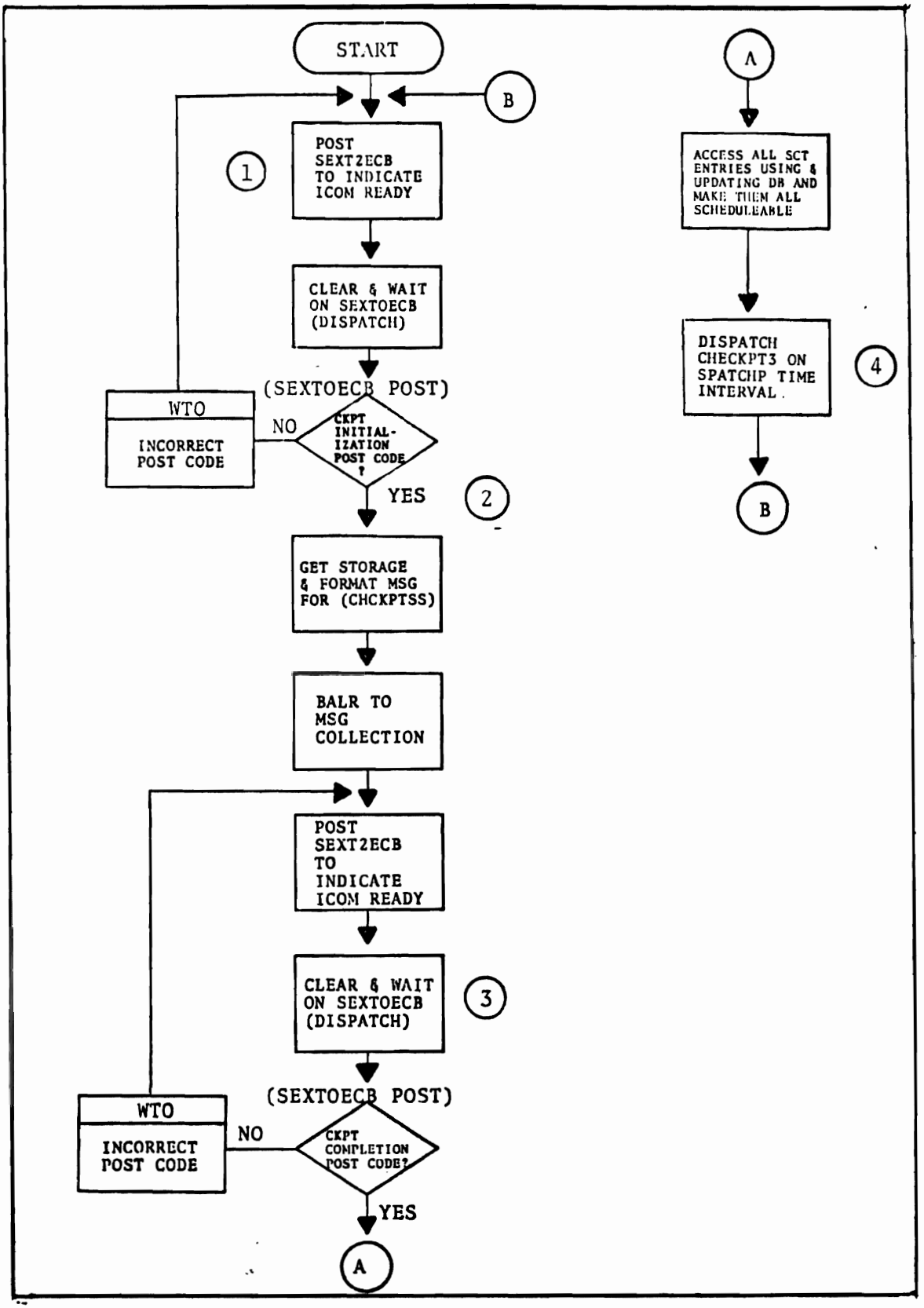

Figure 5-6. GDBSTUP Logic. (Encircled numbers correspond to list on previous page).

(4) It then resets the timer for the next checkpoint based on SPALIST parameter TCHP and waits for a checkpoint request from the DBMS. Note that the first word of the first save area (accessed via TCBFSA) is used to locate the SPA. At INTERCOMM startup time, the V-CON of the SPA is placed in this field. If the user intended to use that same location, the field SEXUSER in CSECT SPAEXT has been reserved for use instead. Label SPAEXTAD (SPA extension address) in CSECT SPA provides a V-CON for SPAEXT.

5.7 ON-LINE DATA BASE REQUEST HANDLING (DBINT)

The user must provide a reentrant program (with at least one entry point named DBINT) which will transfer application data base requests from the on-line region to the data base region. Depending on the conventions of the DBMS being supported, this may be the only entry point necessary.

If the normal data base calls are in the form:

```
CALL   DBRQ, (FUNC1, FUNC2, ....FUNCN)
```

where:

DBREQ = the single entry point which is used for all requests GET, PUT, etc.

FUNC1-FUNCN = the parameters which this entry point analyzes to determine the action it is to take.

the entry point DBINT may coincide with the normal entry point DBREQ. This entry is currently defined in REENTSBS (see COBOL and PL/I programmers manuals for details) with a code of 84. Therefore both reentrant COBOL and PL/I programs may call this program using this code.

The DBMS to be supported may require different entry points to be called for each function requested of it. In this case, the entry point DBINT must still be present although it may or may not be used by any application request.

The following techniques should be utilized by the on-line data base interface:

- No WAIT macros should be issued by this program since this will cause all other on-line functions to halt (i.e., polling of lines, processing of other application programs). Instead of WAIT the INTERCOMM DISPATCH macro must be used. Figure 5-3 describes the various parameters of this macro.

- The DBMS may need to have a unique identifier to distinguish concurrently processing threads. INTERCOMM maintains such an identifier, the thread identification or thread number.

  NOTE: Thread number = 1 to 255
  When DBPURGE (optional user exit) is called a thread is about to end.

  It is a one-byte field and may be accessed in the following manner:

```
L   Rx,=V(IJKTHRED)
MVC ID,3(Rx)
```

  where Rx is any general purpose register other than zero and ID is a one-byte area to receive the thread identifier.

  The subsystem's timeout limit value is specified by the SYCTTBL macro TCTV parameter. There are two INTERCOMM macros, DISABLE and ENABLE, which temporarily deactivate time-out purge processing for a subsystem. These should be used by the interface to prevent a subsystem which times out from having its resources purged (i.e., core) while an I/O is being performed within the DBMS. The DISABLE should be issued before the I/O request is made to the Data Base Management System task to deactivate time-out/purge processing. The ENABLE should be issued by the interface after the I/O is posted complete and all data has been transferred into the INTERCOMM region. Only after the ENABLE is issued will the thread be allowed to be purged. This coding technique should be utilized in conjunction with the DBPURGE exit routine (see below). These macros are normally for INTERCOMM use only; contact the SE-on-duty for assistance in their use.

Since Data Base Management Systems vary greatly in their interface with application programs, no further description of this interface will be attempted here. Each individual DBMS must be treated individually, based on user requirements.

DBRELEX is conditionally called when a subsystem ends normally or abnormally. This entry point should be supplied for purging threads (exclusive control, etc.).

Alternately, DBPURGE is another entry point available for use. It may be used for releasing exclusive control or purging threads which may still be marked as active to the Data Base Management System. The advantage of using this entry point rather than DBRELEX lies in the ability to utilize the Resource Management facility for delaying the purging of a thread until all outstanding I/O is complete. This entry point will only be called after any outstanding I/O has completed or after the ENABLE for the thread has been issued.

DBPURGE receives a two-word parameter list as follows:

```
DC  A(SCT)
DC  A(RETCODE)
```

The first word contains the address of the SCT entry for the terminating subsystem and the second word contains the fullword return code which was passed to the Subsystem Controller by the application subsystem on completion.

Abnormal termination by a subsystem is indicated by one of the following codes:

X'00FFFFFF'	*indicates the application program timed-out*
F'904'	*indicates the application program program checked*
F'908'	*indicates the application program program checked*

Any other code indicates normal completion of the subsystem.

5.8 CLOSEDOWN OF ON-LINE REGION (DBCLOSE)

A user-supplied program with the entry point DBCLOSE is called during INTERCOMM closedown processing. This program functions to notify the DBMS that INTERCOMM is no longer processing. The functions that should be included are:

- The freeing of any control blocks which may be used for interfacing with the DBMS.
- A detach of the DBMS if it is executing in the same region.

The DBMS, upon receiving notice of INTERCOMM closedown, might force out all buffers and write a checkpoint (or set an indicator to take a checkpoint as soon as possible) to note the

fact that INTERCOMM's closedown is complete. Again, most of the processing which must be performed is greatly dependent on the design of the DBMS and therefore must be determined by each individual user.

5.9 ABNORMAL TERMINATION PROCESSING (DBSTAE, DBMABEND)

There is a need for special STAE routine processing since the multiple regions communicating with each other must be informed of the abnormal termination of any of the regions with which they are processing. There are three abend conditions which must be provided for: the DBMS region, an INTERCOMM region and a batch region abend.

The following procedures are suggested. Each time a new region which is to access the data base is initiated it must locate the Data Base Management region to inform it of its presence. At this time a channel of two ECBs should be defined by that region. One of the ECBs should be waited upon by the application region for notification by the DBMS of an abend condition, the other should be waited upon by the DBMS for notification of the application program's abend. (Waits are performed by the DISPATCH macro.) The user may supply an interface routine within the INTERCOMM region with entry point DBSTAE. If it is supplied, INTERCOMM STAEEXIT will CALL DBSTAE after it has been determined that the INTERCOMM task cannot be reinstated. This supplies the user with the ability to notify the DBMS region that the INTERCOMM task has abended. The DBMS should, upon receiving notification of abnormal termination, purge all DBMS services and resources in process for the INTERCOMM region.

The DBMABEND routine in the INTERCOMM region should be dispatched at startup time by the DBSTART routine to wait on an ECB. The ECB will be posted by the DBMSTAE routine in the DBM region if that region abends. Since it may be posted at any time, DBMABEND must be resident.

Upon receiving control, DBMABEND should take the following recommended steps:

- Set an indicator to inform DBINT not to try to process further data base requests.
- Do a WTOR requesting the operator to reply ABEND, CONTINUE or WAIT.

- If ABEND is indicated, the program should abend the INTERCOMM region and restart procedures may then be initiated.

- If CONTINUE is indicated all data base subsystems should be marked nonschedulable and an exit should be done. INTERCOMM will continue processing non-data base programs.

- If WAIT is indicated, all data base subsystems should be marked nonschedulable then another WTOR done to which the operator will reply only after the DBMS region has been brought up again. As described previously, this technique should only be used in inquiry-only systems.

- When a reply is received, initialization processing should be repeated and DB subsystems marked schedulable.

5.10 DATA BASE MANAGEMENT SYSTEM CHECKPOINT REQUIREMENTS

The Data Base Management System may initiate checkpoints (based on time interval or number of updates) when INTERCOMM is not running. However, when INTERCOMM is operational it initiates checkpoints. The DBMS is first notified that a checkpoint time has been reached when it receives the checkpoint prepare command from the INTERCOMM region. Upon receiving this command the DBMS must determine if a checkpoint is now possible. One of the following criteria must be met:

- No batch programs which update on-line data base files may be running, or

- Any batch program which updates on-line data base files has either not performed any updates as yet or the batch program has a checkpoint facility which may be invoked by the DBMS when necessary.

If either of these two conditions are met, the DBMS must do the following:

- Quiesce batch update activity for the duration of the checkpointing procedures. A suggested action is to stop accepting any requests from batch programs except those which will complete an update operation in process.

- Respond to the checkpoint prepare command to indicate that a checkpoint command would now be accepted. If a checkpoint is not currently possible (that is, neither of the above two conditions can be satisfied), the DBMS must set a switch which will indicate to it that as soon as one of these conditions can be satisfied (probably when a batch region closes down), the checkpoint steps previously noted should then be initiated.

Thus, although Intercomm is initiating the checkpoint (through the PREPARE), the DBMS controls the checkpoint.

5.10.1 Checkpoint Processing (DBCKPREP, CHCKPTSS)

Intercomm, when operating, will initiate all checkpoints within its region. Checkpoints are initiated at regular intervals based on a given time field in the System Parameter List.

The program DBCKPREP is called by DBCHKDSP each time the checkpoint time interval specified in SPATCHP expires. Its only function is to send a checkpoint prepare command to the DBMS. (See Section 5.10, "Data Base Management System Checkpoint Requirements," for the actions which must be taken in the DBMS region.) This program is only needed periodically; therefore, it may be placed in the transient overlay region of Intercomm.

The Intercomm-supplied CHCKPTSS program operates as an Intercomm subsystem. It ensures that all data base updating programs are quiesced by marking them as nonschedulable before allowing the Intercomm checkpoint to take place. After the checkpoint has been taken, a checkpoint command is sent to the residing DBMS. When the DBMS indicates the checkpoint is complete, SEXTOECB is posted and GDBSTUP receives control to mark the data base update functions as schedulable.

5.11 CONDITIONAL ASSEMBLIES

As discussed in Section 4, the members INTGLOBE and SETGLOBE control conditional assembly of the System Parameter Area and SPA Extension. Globals specified in these members control conditional assembly for GDB support as follows:

Global	Module	Condition Defined
&INTSVC	IGC250 username for SPA and SPAEXT CSECTs	Interregion SVC Number
&VS2	IGC250	Indicate a VS/2 environment

A change to any global above necessitates reassembly and linkedit of the associated member.

5.12 INTERCOMM REGION TABLES

In addition to the System Parameter Area, SPA Extension and SCTs described in Section 4, the only additional table(s) required in the Intercomm region to implement GDB support are user-defined based upon user DBMS requirements. Typically, such tables would contain subsystem or thread-dependent control information relating to data base access.

If the DBMS support is to include the ability to initiate a checkpoint request from a terminal, an entry is required in the Verb Table to relate the user-defined verb to the checkpoint subsystem (SSC=Q).

```
[symbol]  BTVERB  VERB=CKPT,SSC=Q,EDIT=YES[,SECURE=YES]
```

EDIT=YES is coded to force the message header VMI to X'00' for internal processing. An Edit Control Table entry is not required. This command may only be used for TOTAL Data Base.

5.13 INTERCOMM REGION LINKEDIT

In addition to linkedit requirements for the non-DBMS functions of the Intercomm regions (Intercomm programs, tables, service routines, Intercomm and user subsystems, etc.), the linkage editior control cards listed in Figure 5-7 are required in the Intercomm region. ICOMLINK Data Base parameters are DBASE and DBLIBR.

```
INCLUDE SYSLIB (dbstart)
INCLUDE SYSLIB (GDBSTUP)
INCLUDE SYSLIB (dbmabend)
INCLUDE SYSLIB (CHCKPTSS) do not INCLUDE if dynamically
INCLUDE SYSLIB (dbint)    loaded
INCLUDE SYSLIB (dbclose)
INCLUDE SYSLIB (dbstate)
INCLUDE SYSLIB (DBCHKDSP) contains CSECT CHECKPT, must
INCLUDE SYSLIB (dbckprep) precede INCLUDE for CHECKPT3
INCLUDE SYSLIB (dbchkcom)
INCLUDE SYSLIB (RESTORE3)
INCLUDE SYSLIB (DBRSTRT)
INCLUDE SYSLIB (LOGPROC)
INCLUDE SYSLIB (READBACK,INTDBLOK)
  .
  .
  .
OVERLAY A                 Startup Overlay
        INSERT DBSTART
        INSERT DBRSTRT
        INSERT RESTORE
        INSERT LOGPROC
        INSERT READBACK
        INSERT INTDBLOK
OVERLAY A                 Closedown Overlay
        INSERT DBCLOSE
```

Figure 5-7. INTERCOMM Linkedit (lowercase letters indicate user-supplied modules)

5.14 INTERCOMM REGION DD CARDS

If the DBMS is operating in a region separate from INTERCOMM, all DD cards for GDB data bases are supplied as part of the JCL for the GDB region. If the DBMS is attached in the INTERCOMM region, the DD cards required by the DBMS must be included in the INTERCOMM execute deck. In the single region environment, if both the DBMS and INTERCOMM require the same ddnames then the Alias option of the File Handler (see OS and VS File Recovery Users Guide or Operating Reference Manual) can be used to alter INTERCOMM's required ddname(s).

Both the INTERCOMM startup and restart decks for the INTERCOMM/GDB support must include all the usual and necessary JCL to operate, irrespective of GDB as delineated in the Operating Reference Manual (see Sections 10 and 12 for complete detail). Additionally, the INTERCOMM restart deck with GDB must contain a DD card defining GDBWKFL. This file is used for coordinating INTERCOMM synchronized recovery compatible with the actions taken by the DBMS recovery. The DD card for GDBWKFL is pictured below.

```
//GDBWKFL   DD   DSN=GDBWKFL,VOL=SER=XXXXXX,UNIT=SYSDA,
                 DISP=SHR,DCB=(DSORG=DA,OPTCD=RF)
```

The data set must be preformatted by the off-line utility CREATEGF with a blocksize of 300 and a minimum of 2 blocks. GDBWKFL is also used by the backout utility (see recommended specifications for writing the Data Base Backout Utility).

5.15 RESTART PROCESSING AND DATA BASE BACKOUT UTILITY

There are three aspects of DB/DC recovery, namely: message recovery, file recovery and data base recovery. Informatics Inc. supplies code within INTERCOMM to recover OS/VS files (the File Recovery Special Feature), to recover messages and queues and to coordinate DB recovery. The user must supply the actual DB recovery programs. Further DB recovery programs must function compatibly with INTERCOMM's recovery.

The user-supplied DB recovery programs include: a dump/restore utility, a data base recovery utility and a backout utility. Only the backout utility need have special coding for INTERCOMM. The specifications for the backout utility are described below.

The Data Base Backout Utility must be able to back out all updates performed on selected data bases up to a specified checkpoint. It is suggested that at least some of the following options are available:

- A time may be specified by the operator which will cause this program to back out all updates to the checkpoint taken at the indicated time.
- The backout program may automatically pick the last checkpoint by reading the data base log backward.
- The program should be able to back out to the last coordinated INTERCOMM/DB checkpoint taken.
- If an INTERCOMM startup record is placed on the log (this might be done when INTERCOMM performs the sign-on function to the DBMS), it should be recognized as the first valid on-line checkpoint. If the operator has selected a time previous to the INTERCOMM startup, the program should inform the operator INTERCOMM backup is complete and continue to the checkpoint taken at the indicated time.

- It may have the ability of not backing up any updates by a specific job step if this information is available in the data base records. This facility may be useful to installations that are using the DBMS with batch programs which update data bases that are not being updated either by the on-line system or by other batch programs. In this way, if a restart must be done after a data base update job has completed, it need not be rerun.

Whichever of the above options is utilized, it is necessary for the backout utility to identify to INTERCOMM the specific checkpoint selected. This is accomplished by the backout utility setting information in the GDBWKFL file which will be processed in INTERCOMM recovery.

The file GDBWKFL is a BDAM data set consisting of 2 records (this data set is formatted by CREATEGF) and RBN0 will always be one of the following:

BYTE 0 - X'00'	This is an empty file, or a record indicating the restore finished successfully
X'FF'	This record contains the first 250 bytes of the last record backed out in this run (or in the previous run if we are in the initialization phase of this run).
BYTES 1 - 3	Unused
BYTES 4 - 7	The time of the checkpoint backed up to in the last restore run if X'00' in Byte 0. (If this field is zeroed, a startup record was reached in the last restore run if X'00' in Byte 0.
BYTES 8 - 299	The image of the last data base log record processed if X'FF' in Byte 0.

RBN1 contains the name(s) of all the step names that have been reversed in this run. The record has the following format:

BYTE 0	The number of step names that are recorded in this record
1 - 15	First step name to be rerun
32 - etc.	

Figures 5-8, 5-9 and 5-10 illustrate suggested program logic. There is a one-time switch to perform the iniatialization code. Initialization processing consists of:

- Writing a WTOR ((a) in Figure 5-8) to determine checkpoint time to be used.
- Writing WTORs ((b) and (c)) to determine step names to be ignored.
- Opening and reading RBN0 of GDBWKFL to determine if this is the second time this job has been run (system failure occured during original restore run) or initial time.
- If this is a rerun due to system failure, RBN1 of GDBWKFL will be read. This record will contain a list of step names that must be rerun after the backout has completed. After initialization has been completed, each log record retrieved will be checked for the following conditions before being reversed against the data base.
  - Is the record an after-image or a before-image?
  - Is it for a job other than any of those designated as not backoutable?
  - If initialization processing indicates this is a rerun of the backout, all log records read will be ignored until the last record causing an update is located. This should then cause a switch to be set so that the backout processing will result.
  - Each time a backout is successfully completed RBN0 of GDBWKFL will be rewritten to reflect this record as the last record processed thus far.

- The job step name (if this option is required) should always be compared against those listed in RBN1 of GDBWKFL. If it is a new one, its name should be entered in the record and written to the file.

- If an actual time is specified by the operator, each checkpoint record will be examined to check the time against that input by the operator to indicate processing has completed. If 'L' or 'L,ICOM' was input, the first appropriate checkpoint record found will indicate that processing has completed.

- When data base processing has completed, a message should be written to the operator specifying which jobs are to be restarted.

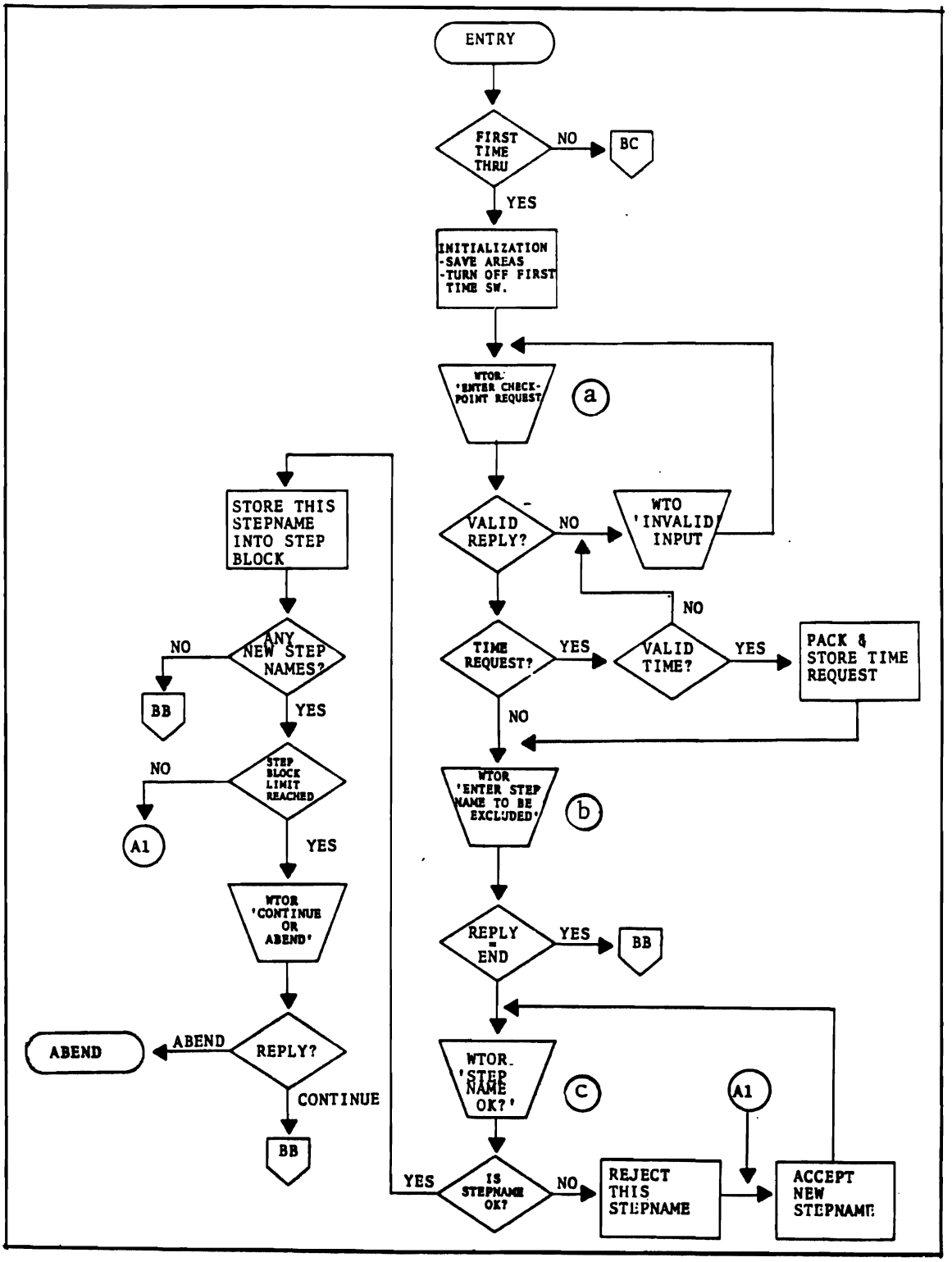

Figure 5-8. Data Base Backout First Time (Initialization)

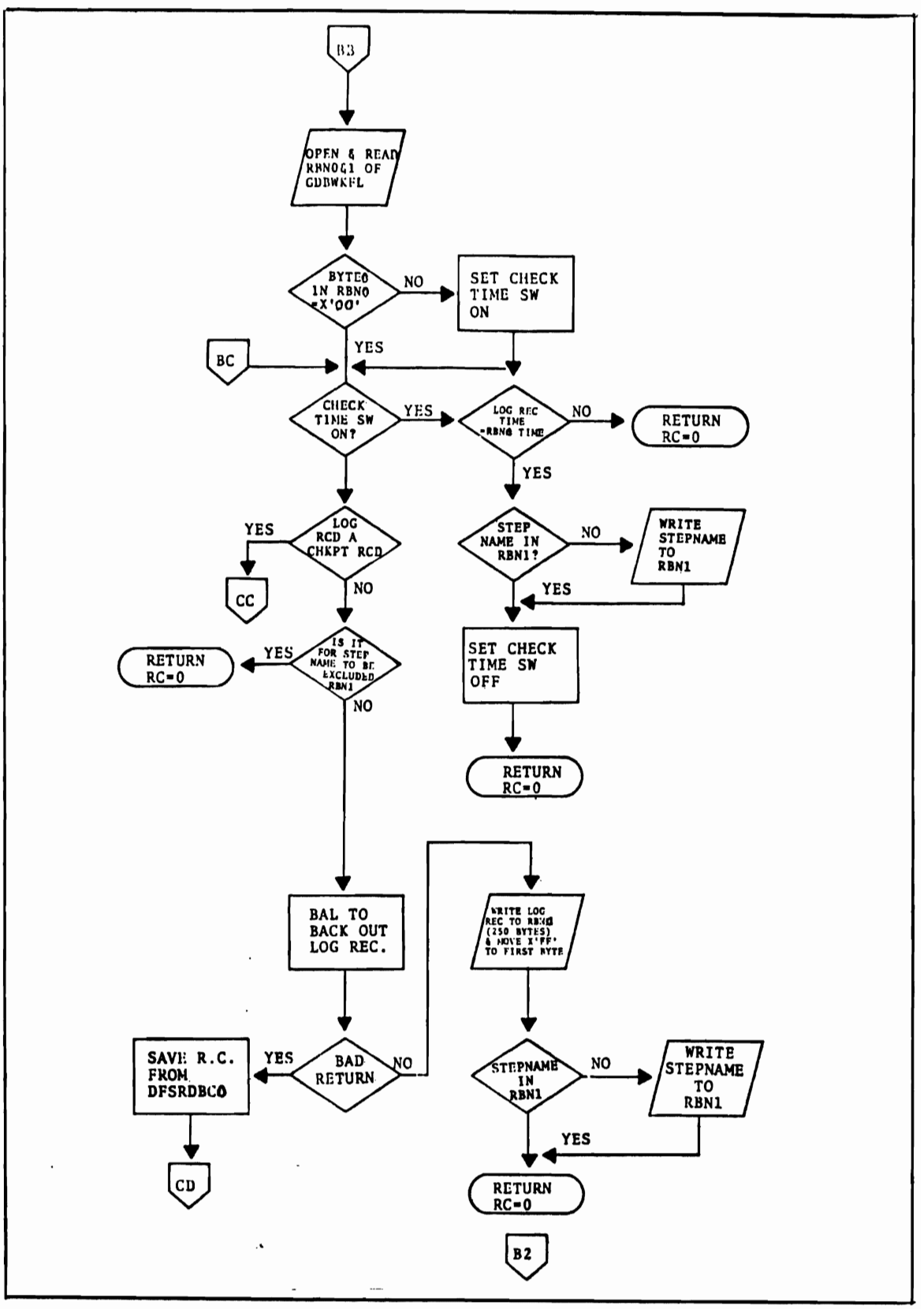

Figure 5-9. Data Base Backout Process

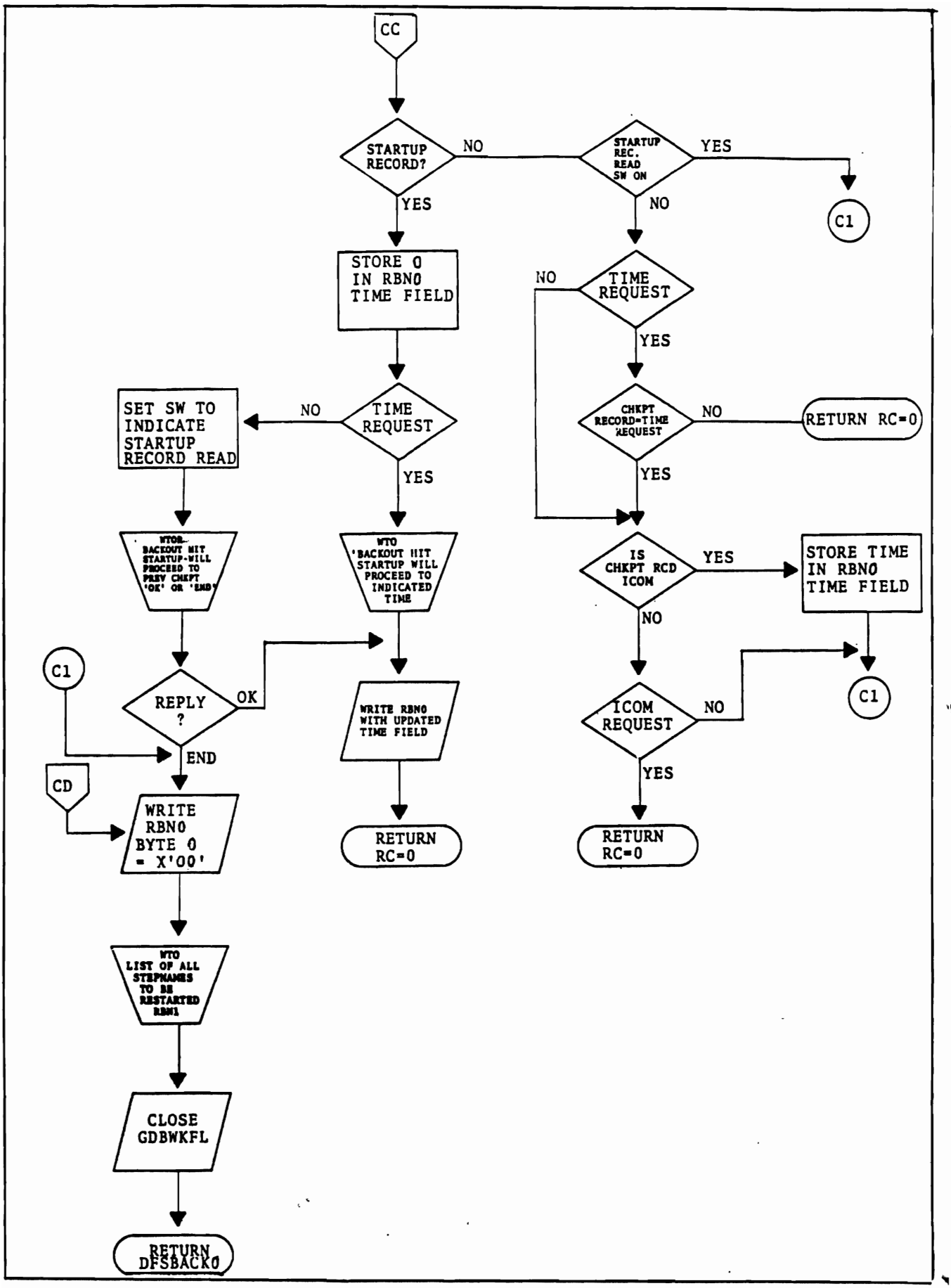

Figure 5-10. Data Base Backout Checkpoint Analysis

Section 6

INSTALLING DL/I SUPPORT

6.1 INTRODUCTION

The Data Language/I component of the IBM program product IMS/DB is referred to in this document as DL/I. The Data Language/I (DL/I) Data Base Management System is supported by Intercomm to allow application programs running on-line to use all facilities of DL/I. Intercomm executes as an IMS batch job.

From a user design point of view, the Intercomm support of DL/I provides a flexibility which includes the following capabilities:

- Inquiry requests to the DL/I data base executing with other Intercomm processing
- Update requests by on-line programs to the DL/I data base executing concurrently with other Intercomm processing
- Ability of on-line inquiry or update programs to execute concurrently with other IMS batch inquiry update program(s), against the same or different DL/I data base(s).

  NOTE: Intercomm functions as a batch DL/I job. As such, it may use either the batch DL/I facility (IMS/DB) or the on-line DL/I facility (IMS/DC). In the latter case Intercomm operates as an IMS Batch Message Program. Depending on the mode of DL/I operation, concurrent access to the same data base by Intercomm and other DL/I jobs might require application logic to serialize access, in accordance with batch DL/I restrictions.

While all of the above facilities are extended to the system designer, the Intercomm implementation of DL/I support requires no modifications to the standard DL/I calling sequence. Standard DL/I CALL statements as specified in the IBM reference literature are used for all data base activity against DL/I files. This compatibility holds for both batch as well as on-line programs. Throughout any on-line programs, standard Intercomm interfaces must be followed. DL/I support in no way changes the standard Intercomm environment or interface requirements.

6.2 INTERFACE SUPPORT MODULES

Before describing installation procedures, this section presents a brief overview of support modules for the DL/I Interface as summarized in Figures 6-1 and 6-2.

Function	Member	Csect	Residency/Comments
INTERCOMM-DL/I Interface Table	username	COBPCBTB	Resident
Startup Processing	STARTDLI	STARTIMS	Startup Overlay
Data Base Request Handling	TICDLICM	IDLISTRT	Resident
Subtask DL/I Processing	SBTSKDLI	SBTSK	Resident

Figure 6-1. Interface Modules--INTERCOMM Load Module

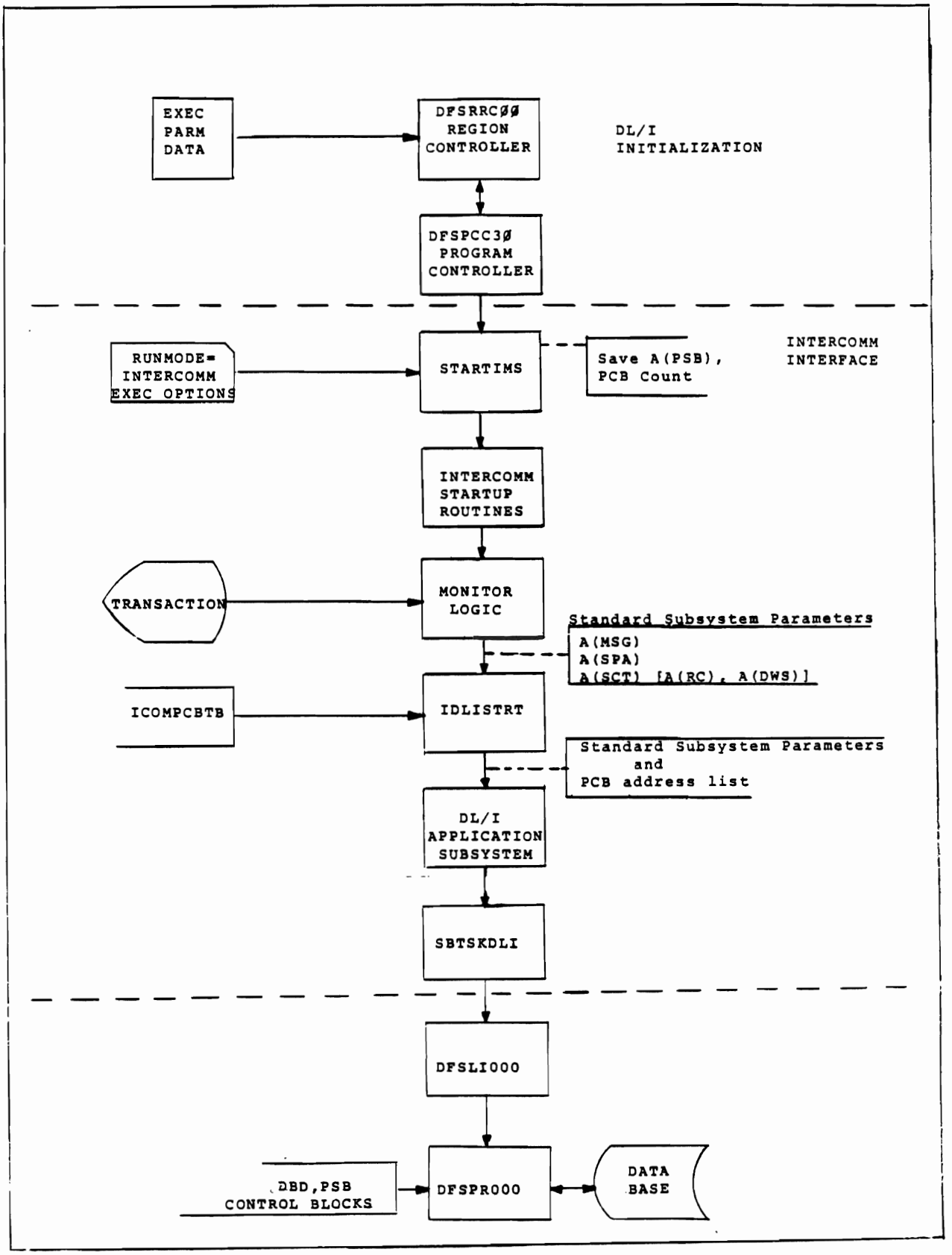

Figure 6-2. INTERCOMM-DL/I Interface Overview

6.2.1 The DL/I Interface Table (COBPCBTB)

This CSECT is a table created by the user via the ICOMPCB macro. Its function is to relate an on-line INTERCOMM application subsystem to its DL/I PCBs (Program Communication Blocks) defined within the DL/I PSB (Program Specification Block) used during on-line execution. Each table entry contains an application subsystem's code (identical to that in the subsystem's SYCTTBL macro) followed by its list of PCB locations within the PSB.

6.2.2 Startup Processing (STARTDLI, STARTIMS)

STARTIMS must be specified as the INTERCOMM load module entry point at linkedit stage. STARTIMS CSECT receives control from DL/I at execution time to begin INTERCOMM initialization. Its functions are to save the location and count of the PCB address list for use by INTERCOMM-DL/I interface software and to forward standard INTERCOMM execution PARM options to the INTERCOMM startup processing routine, PMISTUP. The location of the PCB address list and the count of PCBs are stored into labels PSBSAVE and NUMPCBS, respectively, in IDLISTRT CSECT. As an IMS batch job, INTERCOMM's execution PARM options are set by use of RUNMODE DD input.

6.2.3 On-Line Data Base Request Handling (TICDLICM, IDLISTRT)

All access to DL/I from on-line subsystems is processed by IDLISTRT CSECT. IDLISTRT is called by the INTERCOMM Subsystem Controller prior to passing control to a DL/I application subsystem in order to append its PCB parameters to the standard INTERCOMM subsystem parameters. The PCB parameters are those defined for the subsystem within the DL/I interface table, COBPCBTB. IDLISTRT processing is transparent to the DL/I application subsystem. The parameter list passed to DL/I by the subsystem is in standard DL/I format.

6.2.4 Subtask DL/I Processing (SBTSKDLI)

SBTSKDLI module intercepts calls to DL/I entry points by an application subsystem in order to provide overlapping with other processing within the INTERCOMM region. Its function is to subtask the DL/I processing at that point and does so with INTERCOMM's subtasking facility as described in the Operating Reference Manual. SBTSKDLI must be resident.

Some versions of IMS prohibit subtasking of PL/1 calls. The user must consult IMS documentation with respect to BMP mode and any subtasking restrictions. When no subtasking is permitted, SBTSKDLI must not be included in the linkedit, eliminating overlapping with non-DL/I processing. The CHANGE control cards in Figure 6-3 should be removed.

6.2.5 Checkpoint and Restart/Recovery

If an Intercomm thread or a subsystem thread terminates abnormally for any reason, then Intercomm message restart will requeue messages, placing an appropriate log code in the message header as described in Section 3.6 of this document. However, the current Intercomm DL/I support does not provide for DL/I checkpointing nor for synchronization of Intercomm restart with the DL/I Backout utility.

6.3 SYSGEN OF DL/I

To create the DL/I facility at an installation, the user must have access to the following:

- The IMS System Programming Manual (IBM document)
- The IBM PID Tape for the source and load modules of DL/I.

The System Programming Manual describes how to perform an 'IMSGEN' to create BATCH DL/I. The procedures in that manual must be followed exactly as presented in order to utilize INTERCOMM's DL/I Interface facility.

6.3.1 PSB and DBD Generation

PSB (Program Specification Block) defines PCBs (Program Communication Blocks) for a given batch IMS job. Both provide to DL/I software the descriptions of applications and their accessibility to data bases defined in a related Data Base Definition (DBD). A particular PSB is selected for a given job execution via PARM data in execution JCL. The user must consult IMS DL/I documentation for complete information and instruction on these control blocks.

NOTE: A language specification is noted in the PSB. The user should bear in mind that a high-level language specification in the PSB is downward compatible. That is, a PSB with a language specification for use by COBOL applications can be used by Assembler Language applications but the reverse is not true. A further restriction to PSB usage is that PL/I and non-PL/I applications cannot share a PSB.

6.4 CONDITIONAL ASSEMBLIES

As discussed in Section 4, the members INTGLOBE and SETGLOBE control conditional assembly of the System Parameter List (SPA) and SPA Extension (SPAEXT).

A change to any global necessitates reassembly and linkedit. Refer to the INTERCOMM Operating Reference Manual (ORM) for further information on SPA and SPAEXT.

The &DLI global indicates DL/I is being used at the installation. This global should be set to 1.

6.5 DEFINING DL/I SUBSYSTEMS TO INTERCOMM

Every application subsystem monitored by Intercomm requires an entry in the Subsystem Control Table (SCT) as described in the Intercomm Operating Reference Manual. A subsystem entry is defined via the Intercomm SYCTTBL macro. For DL/I subsystems, the SYCTTBL parameter DBASE must be coded with DL1 or UDL1 for read-only or update activity, respectively.

6.5.1 Use of the Resource Enqueuing Facility

If a DL/I data base is accessed concurrently by more than one subsystem thread, use of the Resource Enqueuing Facility may be required in order to protect data base integrity. This would be the case if any subsystems must maintain sequential positioning during their processing and/or must perform data base updates. Subsystem threads which perform inquiries only and do not need to maintain sequential positioning may process concurrently without the use of resource enqueuing.

The Resource Enqueuing Facility is documented in the Operating Reference Manual.

6.6 CONSTRUCTING THE INTERCOMM-DL/I INTERFACE TABLE (COBPCBTB)

This table, having CSECT COBPCBTB, is constructed by coding an Intercomm ICOMPCB macro for each on-line DL/I application subsystem executing under Intercomm. Any changes to the PSB (Program Specification Block) referenced during Intercomm-DL/I execution must be reflected within COBPCBTB. Similarly, any subsystem changes incolving its PCB (Program Communication Block) parameters must be accounted for in its entry within COBPCBTB.

6.6.1 Coding the ICOMPCB Macro

The ICOMPCB macro instruction is used to create COBPCBTB, the Intercomm-DL/I Interface Table. The purpose of the PCB Index Table is to provide TICDLICM with the relative offset into the PSB of all the PCBs used by a subsystem. The first occurrence of the macro generates a COBPCBTB CSECT statement.

The form of the ICOMPCB macro instruction is as follows:

(symbol)	ICOMPCB	SSCH=	High Order Subsystem Code
		,SSC=	Low Order Subsystem Code
		,PCBINDX=	(Offset ,[Offset],...)
		[,STOP]	

SSCH=

Specifies the high order byte of the subsystem code. This corresponds to the SUBH operand of the SYCTTBL macro. This operand must be coded.

SSC=

Specifies the low order byte of the subsystem code. This corresponds to the SUBC operand of the SYCTTBL macro. This operand must be coded.

PCBINDX=
Specifies a list of offsets into the PSB of the desired PCBs. The PCB offsets are relative to one. This is a required operand with no default. Multiple offsets must be enclosed by parentheses.

STOP=
Coding STOP creates a fullword of X'FF' to signify the end of the PCB Index Table. This is optional with no default. If STOP is not coded, the user must follow the last ICOMPCB macro with PMISTOP macro.

6.7 INTERCOMM MODULE LINKEDIT

The ENTRY statement must specify STARTIMS. In addition to linkedit requirements for the non-DBMS functions of the INTERCOMM region (INTERCOMM programs, tables, service routines, INTERCOMM and user subsystems, etc.), the following linkage editor control cards are required in the INTERCOMM region:

```
ENTRY STARTIMS                    REPLACES ENTRY PMISTUP
*THE FOLLOWING MEMBERS FOR BOTH INQUIRY AND UPDATE
    INCLUDE SYSLIB(TICDLICM)
    INCLUDE SYSLIB(COBPCBTB)
    INCLUDE SYSLIB(STARTDLI)
    INCLUDE SYSLIB(SBTSKDLI)
    CHANGE CBLTDLI(SBTSKCBL)
    CHANGE ASMTDLI(SBTSKASM)
    CHANGE PLITDLI(SBTSKPLI)
    INCLUDE SYSLIB(DFSLI000)
```

Figure 6-3. INTERCOMM Module Linkedit

The following additional revisions must be made to the linkedit input:

```
CHANGE DBINT(SBTSKDLI)

INCLUDE SYSLIB(REENTSBS)
```

6.8 EXECUTION JCL

The standard INTERCOMM execution JCL as described in Section 10 in the INTERCOMM Operating Reference Manual provides the base for these additions and changes:

. Additions

- All DD statements for the DL/I data base files referenced during job execution are required.

- A DD statement having the ddname RUNMODE is required. RUNMODE provides standard INTERCOMM execution PARM data, as described in Section 10 in the Operating Reference Manual. RUNMODE input is supplied via a SYSIN data stream. It is in the format nnmode, where nn is the two-byte binary length of the standard PARM data beginning in position 1, and mode may be STARTUP, RESTART, etc., followed by additional execution time PARMs as required. For example:

```
//RUNMODE DD *
08TEST,WTO
```

- STEPLIB DD concatenation to standard INTERCOMM execution procedure JCL of the following data sets:

```
For IMS/2.3.1: concatenate in order,
// DD DSN=PMI.MODREL,DISP=SHR
// DD DSN=IMS2.VSRES231,DISP=SHR

For IMS/VS:
// DD DSN=PMI.MODREL,DISP=SHR
// DD DSN=IMSVS.RESLIB,DISP=SHR
```

- DDNAME=IMS is required to specify DBD and PSB libraries,

```
For IMS/2.3.1:
//IMS DD DSN=IMS2.DBDLIB,DISP=SHR
//    DD DSN=IMS2.PSBLIB,DISP=SHR

For IMS/VS:
//IMS DD DSN=IMSVS.DBDLIB,DISP=SHR
//    DD DSN=IMSVS.PSBLIB,DISP=SHR
```

- Changes

  The EXEC statement must specify PGM=DFSRRC00, with PARM=recommend as follows:

  ```
  For IMS/2.3.1:
  PARM='DLI,INTCOMM,&PSB'
  where INTCOMM is the load module name
  for INTERCOMM, and &PSB is the PSB name
  for the region.

  For IMS/VS:
  PARM='DLI,INTCOMM,&PSB'
  ```

  Consult IMS literature for description and usage of other execution parameters.

6.9 CODING ON-LINE SUBSYSTEMS

Reentrancy can be effected across multiple subsystems using DL/I running under INTERCOMM.

Each subsystem processing a message is invoked by the INTERCOMM monitor. On entry, both standard INTERCOMM and required DL/I parameters are passed to the subsystem. The parameter list received by each subsystem provides the INTERCOMM parameters first, followed by the DL/I parameters (PCBs used by program). The parameter list is of variable length; however, each program receives the same number of parameters each time it is entered. The sequence of the parameter list is as follows:

```
MSG,SPA,SCT[,RETCD[,DYN-CORE]][[,PCBNAME[,PCBNAME],---]
```

where:

MSG = Address of the message

SPA = Address of the SPA

SCT = Address of the SCT entry for this subsystem

RETCD = Address of return code is a high-level language (present only if language is COBOL or PL/I)

DYN-CORE = Address of dynamic work area if reentrant COBOL is used (not present for other languages)

PCBNAME = Address of a PCB to be used by the subsystem as specified in COBPCBTB CSECT

Calls to the DL/I region are issued using the same syntax as ordinary use of batch DL/I. Reentrant ANS COBOL and PL/I subsystems use the REENTSBS offset code 84 to allow access to the entry point DBINT:

```
CALL 'COBREENT' USING CODE-84, standard DL/I parameters
```

Chapter 7

INSTALLING TOTAL SUPPORT

7.1 INTRODUCTION

TOTAL 8 (8.0, 8.1 and 8.2) is supported by Intercomm in two modes of operation. In one mode, Intercomm and TOTAL operate in the same region or address space as a main task and subtask, respectively. In the second mode of operation, Intercomm and TOTAL operate in two separate regions or address spaces. In this case TOTAL must be executing when Intercomm is brought up, and all communication between the two regions must be initiated through the use of an interregion SVC routine supplied by Cincom Systems.

In both modes of operation, the facilities of the TOTAL Data Base Management System may be utilized by one or more batch region(s). These off-line programs may access and update the on-line data base while Intercomm is up and data integrity will still be maintained through the use of procedures provided by Intercomm and TOTAL.

In addition to the general requirements for DBMS installation common to all systems as discussed in Chapter 4, the following steps are required to utilize Intercomm support for TOTAL:

- SYSGEN of TOTAL
- Conditional Assembly of the TOTAL File Table
- Intercomm Region Installation:
    - Tables
    - Linkedit Considerations
    - JCL Considerations
- Batch Region(s) Installation:
    - Linkedit Considerations
    - JCL Considerations
- Restart/Recovery Procedures
- Data Base Backout Utility
- Coding On-Line Subsystems
- Coding Batch Application Programs

Function	Member	CSECT (Entry)	Residency-Comments
Startup Processing	TOTSTART CPLUNCSS	TOTSTART(DBSTART) CPLUNCSS	Resident. Resident--COPT command.
Data Base Request Handling	CSTAMVxx* TOTINT* PDATBASE	DATBAS(see note) TOTINT(see note) PDATBASE(DBINT) (DBSTAE) DBPURGE	Cincom-supplied. Cincom-supplied. Resident-subsystem interface to TOTAL. Resident-purge outstanding TOTAL ENQs.
Data Base File Table	username (must have alias of TOTFILE if PMITOTRS is used)	TOTFILE	Resident
Macro	TOTFLGEN		Generate TOTFILE entries.
Closedown Processing	TOTCLOSE CPLUNCSS	TOTCLOSE(DBCLOSE) CPLUNCSS	Resident. Resident--UNPT command.
Abend Processing	ABTOTEND	ABTOTEND ABTOTFIX (Rel 10) TOTABRTS (Rel 9)	Resident-used for Attach Mode only.
Checkpoint Processing**	CHCKPTSS DBCHKDSP CHECKPT3 TOTCHKPT	CHCKPTSS CHECKPT CHECKPTO TOTCHKPT	Subsystem with user assigned residency. Resident. Resident. Resident--used only if coordination with Batch Region updates needed.
Restart Processing**	DBRSTRT RESTORE3 LOGPROC REQONDDQ AUTORCVR	DBRSTRT RESTORE LOGPROC AUTORCVR (AUTORSTU) (AUTORCLD) (AUTORDBR) (AUTORREQ)	Resident. Resident. Resident. Resident-serial restart. Resident--Auto-restart. Called by STARTUP3. Called by CLOSDWN3. Called by DBRSTRT. Called by REQONDDQ.

* Cincom-supplied Module
** Required Only for Restart/Recovery Functions

Figure 7-1. TOTAL Interface Modules--Intercomm Region

NOTE: Beginning with TOTAL Release 8, the entry point DATBAS does not exist in the interface module used with Intercomm: CSTAMVIC (attach mode), or CSTAMVMT (multi-batch mode). The CSECT TOTINT in the interface module (not in TOTINT) must be renamed to DATBAS via a linkedit CHANGE statement.

Function	Member	CSECT	Comments
Request Coordinated Checkpoint	DATBASXT*	DATBASXT	Required in Batch Region only if updates performed to on-line data base.

* Required only for Restart/Recovery Functions

Figure 7-2. TOTAL Interface Module--Batch Region

Function	Member	CSECT	Comments
Backout Utility	PMITOTRS	PMITOTRS	Reverses updates to TOTAL to a given checkpoint.
Driver Program	ATTOTRS	ATTOTRS	Attaches TOTAL before backing out the data base via PMITOTRS.
	CSITA014	CSITA013	TOTEXIT macro-generated.
		CSITA014	Manage TOTAL logging for Automated Restart.
	TOTAREOF	TOTAREOF	Calls ICOMFEOF for valid EOF on TOTAL log.

Figure 7-3. Interface Modules for TOTAL Data Base Restoration, Logging

7.2 INTERFACE SUPPORT MODULES

Before describing detailed installation procedures, this section presents a brief description of support modules and their logic for the Intercomm and TOTAL interface; modules are summarized in Figures 7-1, 7-2 and 7-3. Certain modules are required only for data base restart/recovery functions and need not be considered if only inquiries are performed by Intercomm subsystems, and if batch programs which execute concurrently with Intercomm perform no updates against on-line TOTAL files.

7.2.1 TOTAL File Table (TOTFILE, TOTFLGEN)

The TOTFILE module is a table created by the user via the Intercomm-supplied TOTFLGEN macro. It supplies the names of all the TOTAL data sets (both variable entry and master) which the user intends to access via Intercomm subsystems, and how they are to be accessed (read only, update, etc.). The data base descriptor name and TOTAL logging options may also be defined.

7.2.2 TOTAL Startup Routine (TOTSTART)

This module performs all the Intercomm region startup processing for TOTAL. If present, it is called by STARTUP3 with a parameter list passing the address of the EXEC card PARM field. The parm field is checked for a DBMOD-name override for TOTFILE (see Section 7.7).

If TOTAL is to run as a subtask of Intercomm (specified by TOTATT=YES for the SPALIST macro assembly), TOTSTART attaches TOTAL using the following parameters:

```
ATTACH EP=TOTALMT,PARAM=parm-list,ETXR=ABTOTEND,ECB=TOTEND
```

This provides TOTAL with the information which would normally be provided to it through the PARM field on the TOTAL EXEC card. The parm-list is in the TOTFILE table and is defined as:

```
DC Y(parm-len),C'xxxxxx,log-opt,NUL,Y,access'
```

where:

parm-len is the length of the parameter list.

xxxxxx is the data base descriptor name provided by the user via the TOTFILE table. The name may be overridden by an EXEC statement PARM option (see Intercomm Region JCL, Section 7.7).

log-opt specifies TOTAL logging options to be used as specified in TOTFILE via the TOTFLGEN macro (default is NBNN).

NUL,Y indicates the TOTAL cross-region SVC is not used and Y indicates the environment is MVS.

access is the type of data set access (default is RDONLY).

The ATTACH is followed by a Dispatcher wait of 30 seconds to allow TOTAL to complete its initialization processing.

The remainder of TOTSTART processing is the same for both modes of operation (separate region or Attach mode). A TOTAL sign-on is executed. If unsuccessful, a message is issued and TOTAL requests will not be executed. If the TOTAL data base access is not specified in TOTFILE as RDONLY (via ALL=READ or by specifying only a READ file list), and TOTAL executes in another region, a QUIET is issued because Intercomm must regulate checkpoint times. If logging of any form is used, a MARKL is then issued.

All TOTAL data sets to be used (as specified in the TOTFILE Table) will then be opened with an OPENX, OPENV or OPENM (depending on how the files were defined) via calls to DATBAS (in the Cincom-supplied interface module).

If a bad return code is received from any of these opens, a message (DB103I) is issued for each failing file name. Then when open processing completes, a WTOR will be issued to the CPU operator in the following form:

OPEN FAILED FOR nnn TOTAL DATA BASE FILES - REPLY CONT OR CANC

If the operator replies CANC, Intercomm will be aborted (User abend 2004); otherwise, processing will continue but the failing data set(s) will not be accessible during this run of Intercomm. However, if failure was on an OPENX when the ALL parameter is coded on the TOTFLGEN macro, then no specific file names are provided and the number in the message is 1 (the actual number of failing files is not available to Intercomm). Check the file names specified in TOTFILE, DBMOD gen, and Intercomm JCL against those referenced in the DB103I message(s). If the reply is CONT, see also Section 7.2.10 - CPLUNCSS.

Finally, TOTCHKPT is conditionally called (at entry TOTCKPT1) to coordinate batch region checkpointing (see Section 7.2.9).

7.2.3 Normal On-Line Processing (PDATBASE, TOTINT, DATBAS)

All calls to TOTAL from on-line subsystems are to PDATBASE (entry point DBINT), an Intercomm-supplied program included in Intercomm's linkedit. The parameter list must be the standard one for TOTAL, preceded by the address of the input message the application subsystem is processing. (Only Intercomm calls DATBAS for system functions.)

The resource management thread number (IJKTHRED) is used to uniquely identify each user within the Intercomm region; it is passed to TOTAL in register 12. The address of the standard TOTAL parameter list is placed in register 1 and a call is made to TOTINT (a Cincom-supplied program) which will in turn communicate the request to TOTAL. Upon return, register 1 is checked for zero. If not zero, the value in register 1 is the address of an ECB to be waited on by PDATBASE (via the Intercomm Dispatcher) for completion of the I/O request.

When register 1 is zero, PDATBASE checks the STATUS field (the second parameter passed to TOTAL) for the characters HELD or TFUL. A status of HELD means that the record is being held exclusively by another user. A status of TFUL means TOTAL's tables are full. The request will be resubmitted after a short time delay. However, if the calling subsystem has already timed out, the request is not resubmitted, but PDATBASE exits so the thread can be purged. Any other status means the request was satisfied instantaneously, or an error condition indicated by the STATUS field was discovered, and PDATBASE returns to the caller.

When the ECB that PDATBASE has been waiting on is posted, another call is made to TOTINT. The status returned by TOTINT on this call is checked for HELD or TFUL. The processing described above is followed for these status codes. To prevent loops or subsystem time-outs, PDATBASE keeps a count of the number of times it has to resubmit the call, and after 15 TFUL codes, or after 1001 HELDs, PDATBASE will change the STATUS field to REPC and return to the caller.

For the TALY$DA system command display of subsystem thread status, PDATBASE contains logic to set a flag when a subsystem interfaces to TOTAL. The flag is turned off before PDATBASE returns to the subsystem. However, if the subsystem should time-out during the TOTINT call and subsequently become hung (cannot be purged) after a second wait for the subsystem TCTV time to expire, then the flag is not reset. A subsystem status of HUNG-DB in the TALY display indicates a potential problem in accessing the TOTAL data base. If several subsystem threads are hung accessing TOTAL, then disk access contention or batch program processing against the same DBMOD-name is often the cause. Due to the resulting reduction in the number of subsystem threads able to access TOTAL concurrently, a response time degradation may occur, requiring Intercomm (and TOTAL) closedown and restart.

In TOTAL Release 8, the DATBAS module has been replaced by an interface module which the user must construct via linkedit. The name of this module will be CSTAMVIC or CSTAMVMT depending on the TOTAL environment. To ensure proper operation of Intercomm/TOTAL, the Csect TOTINT in this interface module (not in the member TOTINT) must be renamed to DATBAS via a linkage editor CHANGE statement.

7.2.3.1 PDATBASE User Exit--USERPDBE

After PDATBASE acquires and chains a save area, a user exit whose Csect name must be USERPDBE is conditionally called. At entry, register 1 points to the parameter list passed to PDATBASE (DBINT). Standard linkage conventions must be used. The exit may be used to gather statistics on data base calls (see System Accounting and Measurement facility in the Operating Reference Manual) or to record user log entries on data base activity. Note that if the exit may give up control to the Intercomm Dispatcher, it must be coded as reentrant (acquire and chain a dynamic save/work area) and subsystem TCTV times may need to be increased.

7.2.4 Transaction Termination Processing (DBPURGE)

In the load module PDATBASE, there is a DBPURGE entry point which is called by RMPURGE after the normal or abnormal termination of a transaction. This Csect issues a DEQUE to TOTAL for that transaction's thread number via an internal call to PDATBASE. DBPURGE is not called if the thread has timed out and become 'hung' (see Section 7.2.3).

7.2.5 Closedown Processing (TOTCLOSE)

During Intercomm's normal closedown processing, a call is made to the Intercomm-supplied program TOTCLOSE at entry DBCLOSE. This routine issues a sign-off to TOTAL. If checkpointing is used, a coordinated (with batch region(s)) Intercomm checkpoint is taken, a request for a QUIET record to be written to the TOTAL log is issued, and then a MARKL is issued (unless Intercomm has abended).

If TOTAL is executing in a separate region, a FINAL call followed by a DEQUE call is made to TOTAL. If TOTAL is a subtask of Intercomm, an ENDTO call is made followed by a DETACH of the subtask.

7.2.6 Abend Processing (ABTOTEND, STAEEXIT, DBSTAE)

Special processing is necessary to insure data integrity if either Intercomm or TOTAL should abnormally terminate.

If TOTAL should end abnormally when running as a subtask of Intercomm, control will be given by MVS to a resident routine ABTOTEND (this routine name was supplied in the ETXR parameter of the ATTACH). This program will mark all subsystems using TOTAL as being nonschedulable.

The systemwide bit indicating that TOTAL is currently operational is reset. When this bit is off, PDATBASE will not allow any further TOTAL request to be initiated and will return a status of DOWN to any program calling it for a TOTAL function.

A detach of the TOTAL task is then made, followed by a WTOR:

TOTAL DOWN-REPLY CONTINUE OR ABEND OR RESTART

If the operator replies RESTART, TOTAL will be reattached, the TOTAL startup functions will be reinitiated (see the description of TOTSTART), and TOTAL programs will be marked as schedulable again. If this reattach of TOTAL is unsuccessful, then the RESTART option is not again provided the operator in a subsequent WTOR. The RESTART option is available only if the data base is used for inquiry alone, because if the data base is being updated, its status will be unpredictable when TOTAL is reattached without initiating restart procedures. In an updating environment, only the CONTINUE (without TOTAL subsystems) or ABEND options are provided.

When Intercomm and TOTAL are running in two separate regions, if either task goes down, the one remaining must also be cancelled, and restart procedures must be initiated to maintain an up-to-date status of the data base.

To provide for file integrity in the event that Intercomm abnormally terminates, it is necessary to close all TOTAL data sets. To accomplish this, Intercomm's STAEEXIT routine calls the entry DBSTAE in PDATBASE. DBSTAE in turn calls TOTCLOSE to perform TOTAL closedown functions.

7.2.7 Checkpoint Processing (DBCHKDSP, CHCKPTSS, CHECKPT3)

To provide data integrity utilizing Intercomm/TOTAL, the checkpointing facilities of both must be synchronized. This is accomplished by allowing Intercomm to initiate all checkpointing for both tasks when TOTAL and Intercomm are operating simultaneously.

The Intercomm data base checkpoint routine (DBCHKDSP) is triggered periodically on the time interval specified in the SPA. When the timer "goes off," a message is formatted and directed to the checkpoint subsystem CHCKPTSS, as discussed in Chapter 3.

With Intercomm TOTAL support, a terminal operator may also enter a checkpoint request by entering a CKPT verb (transaction-ID) which is directed to the checkpoint subsystem. An entry is required in the Intercomm Front End Verb Table (BTVRBTB) which may optionally restrict this transaction to the control terminal (see Chapter 5).

After CHCKPTSS has ensured that all subsystems that update TOTAL have quiesced, an Intercomm checkpoint is taken via a call to the CHECKPTO Csect in CHECKPT3 and is recorded on INTERLOG. The checkpoint subsystem then forces a TOTAL checkpoint by requesting a QUIET record to be written to the TOTAL log. The call will be in the following form:

```
CALL DATBAS,(QUIET,STAT,COUNT,ENDUP)
```

where the parameters are defined as follows:

```
QUIET   DC   C'QUIET'
STAT    DC   C'****'
        DS   0F
COUNT   DC   X'7FFFFFFF'
ENDUP   DC   C'END.'
```

This will immediately be followed by a MARKL request to record the time of the checkpoint on the TOTAL log.

The checkpoint subsystem then completes processing by activating quiesced subsystems as discussed in Chapter 3.

7.2.8 On-Line Restart Processing (PMITOTRS, LOGPROC, DBRSTRT, ATTOTRS)

The Intercomm restart facility is based on both systems backing up to the last coordinated checkpoint and restarting all processing affecting the data base which was initiated since that time.

TOTAL will be backed up to the last QUIET through the use of PMITOTRS, a utility program supplied by Intercomm. This off-line program reads the TOTAL log backwards and restores before-images of all updates done since the last checkpoint through the use of the TOTAL command WRITD. (The user may cause back-up to other than the last checkpoint by supplying the desired checkpoint time in the PARM field of the EXEC card.) The PMITOTRS program will write the time of the checkpoint to the operator via a WTO. It will then be the responsibility of the operator to supply this time to Intercomm in reply to the ENTER CHECKPOINT TIME REQUEST message produced by DBRSTRT at restart time unless automated restart processing is used (see Section 7.2.11). If TOTAL is running as a subtask of Intercomm, then the driver program, ATTOTRS, must be used to execute PMITOTRS and attach TOTAL using the entry point TOTALMT.

During startup, Intercomm calls LOGPROC to process the previous Intercomm log. The LOGPROC program (Intercomm-supplied) has the responsibility of selectively restarting messages. All messages processed by programs using TOTAL in update mode will be restarted unless indicated otherwise by the application subsystem as discussed in Chapter 3. In order to coordinate checkpoint times used at restart time, DBRSTRT is dispatched by LOGPROC. It will request from the operator the time of the QUIET record to which TOTAL was backed up. When LOGPROC has located the last checkpoint written to the log, it will verify that the checkpoint time is the same as the time of the QUIET record to which TOTAL was backed up. If it is not, LOGPROC will continue selectively restarting messages back to the next previous checkpoint until both Intercomm and the data base have reversed activity to the same checkpoint or QUIET record. LOGPROC then calls the RESTORE routine to recreate the system tables as they existed at that checkpoint. The PMITOTRS utility may be running at the same time as the Intercomm restart procedures if TOTAL operates in a separate region; no TOTAL activity will be started by Intercomm until restart is completed. If TOTAL is running as a subtask of Intercomm, PMITOTRS must be executed before Intercomm is brought up.

For implementation of the Intercomm Serial (single-threaded) Restart facility via REQONDDQ, see the Operating Reference Manual.

7.2.9 Batch Processing (DATBASXT, TOTCHKPT)

Batch programs are able to run concurrently with Intercomm to access on-line data base files. Application programs which are inquiry-only require no change. Batch programs which perform updates to the on-line data base must include one extra module, DATBASXT, in their linkedit, to provide for checkpoint synchronization with the Intercomm and TOTAL regions at batch region startup. Also, TOTCHKPT must be included in the Intercomm region updating the same TOTAL data base as the batch update region.

DATBASXT is an Intercomm-supplied exit routine which is called by the TOTAL interface program, DATBAS, each time a call is made to TOTAL. DATBASXT checks if the TOTAL call being issued is a sign-on request. If not, no action is taken; if it is, interface with the Intercomm TOTAL region is established by a series of system-wide enqueues issued by DATBASXT and TOTCHKPT. If the Intercomm TOTAL interface region is not executing, a checkpoint is requested from the batch region (QUIET and MARKL commands) and a message is issued providing the checkpoint time for eventual TOTAL data base backout in case the batch region fails. If the Intercomm TOTAL region is up (ENQ test not successful), then the enqueues are used to tell TOTCHKPT to force a checkpoint from the Intercomm region. TOTCHKPT tests every 10 seconds whether a batch region is coming up (TOTAL sign-on in progress). Once the checkpoint completes, a dequeue from the Intercomm region allows processing in the batch region to proceed. This checkpoint time now provides a common point at which all regions updating the same TOTAL data base descriptor name may be restarted in case of system or region failure.

The rname used for the enqueues issued by TOTCHKPT and DATBASXT is ICOMTOT1. If several Intercomm regions access different TOTAL data bases, special versions of these programs with different rnames must be linked with each appropriate Intercomm and batch region. The rname may be found at the label ITOT in each program.

TOTCHKPT (entry TOTCKPT1) is called at Intercomm startup by TOTSTART to initialize coordinated batch checkpointing, and is called again at Intercomm closedown by TOTCLOSE (entry TOTCKPT2) to suspend the checkpointing.

Special considerations apply if an Intercomm region using automated TOTAL restart updates the same DBMOD as a batch region. See Section 7.2.11.

7.2.10 Couple and Uncouple System Control Commands (CPLUNCSS)

If it is not possible for TOTAL to be running at the time that Intercomm is started up, communication between Intercomm and TOTAL may be established at a later time via the COPT (couple) system control command. If it is desired to take TOTAL down without bringing down Intercomm, once communication between Intercomm and TOTAL has been established, then communication may be broken via the UNPT (uncouple) system control command.

The COPT command causes the Intercomm region processing for TOTAL (TOTSTART module) to be executed at some time other than Intercomm startup when it is usually called. The COPT command allows certain startup parameters to be overridden dynamically via command operands. Its effect is to allow TOTAL to be started at any time after Intercomm is started without having to bring Intercomm down. The UNPT (uncouple) command causes Intercomm to perform closedown processing related only to TOTAL and then have Intercomm sign off from TOTAL before the time this is usually done (at Intercomm closedown). In effect, UNPT allows TOTAL to be brought down without also having to bring down Intercomm. In addition, a physical close of TOTAL files may be requested.

The format of the COPT command is as follows:

```
COPT[$ALL][$DB=yyyyyy]@
```

where:

ALL will cause all data base update subsystems to be marked scheduleable. If omitted, no DB update subsystems are marked scheduleable.

DB=yyyyyy overrides the default data base descriptor name coded in the TOTFILE table or provided on the Intercomm EXEC statement. Enter six characters (padded on the right with blanks, if necessary).

The format of the UNPT command is as follows:

```
UNPT[$CLOSE]@
```

where:

CLOSE will cause TOTAL files to be physically closed before uncoupling. If omitted, no files are closed.

In the above command descriptions, the $ is the system separator character as defined in the SPA (SPALIST macro, SEP parameter).

To implement the couple and uncouple system control commands, the following steps must be performed:

- Code Front End Verb Table entries in the copy member USRBTVRB and reassemble BTVRBTB:

  ```
  BTVERB VERB=COPT,SSCH=x,SSC=y
  BTVERB VERB=UNPT,SSCH=x,SSC=y
  ```

- Code a subsystem control table entry in the copy member USRSCTS for the CPLUNCSS subsystem and reassemble INTSCT:

  ```
  (name)    SYCTTBL    LANG=NBAL,SBSP=CPLUNCSS,SUBH=x,SUBC=y,
                       NUMCL=1,MNCL=1,RESTART=NO,...
  ```

  where x and y are the same user assigned subsystem codes as specified on the BTVERB macros.

- Include the module CPLUNCSS in the Intercomm linkedit as resident or in an overlay group; or define it as dynamically loadable.

- The modules TOTSTART and TOTCLOSE must be resident.

7.2.11 Automated TOTAL Restart Processing (AUTORCVR, CSITA014, TOTAREOF)

Under Intercomm Release 10, a user-supplied automated restart facility is provided to eliminate operator intervention after abnormal termination of an Intercomm region executing updates to TOTAL, and using message recovery and the checkpoint and restart processing described above. This facility uses the same JCL stream for startup and restart processing and requires the following to be implemented in the Intercomm region:

- Checkpoint processing: documented in the Operating Reference Manual, Chapter 9 and above in Section 7.2.7
- Message restart/recovery: ORM, Chapter 9 and Section 7.2.8
- Serial restart: ORM, Chapter 9
- Automated restart by the new module AUTORCVR: ORM, Chapter 9
- Data Base restart via DBRSTRT: Section 7.2.8
- Intercomm log flip/flop data sets (INTERLOG and INTERLOC) and USERB37E user exit: ORM, Chapters 6 and 9. Note that this processing is recommended to reduce the restart scanning and EOF recovery of the Intercomm log. However, one large log disk data set or even tape data sets may be used, depending on on-line volume and desired rapidity of the restart process
- DD statements for INTERLOG, STRTUPSW, RESTRTLG, LOGDISK, CHEKPTFL, serial restart DDQ data sets, and for INTERLOC if disk log flip/flop is used.

and the following for TOTAL processing:

- Include with the TOTAL load module the provided user exit CSITA014 to manage all TOTAL region activity (update) logging as defined for the TOTAL region and via TOTFILE in the Intercomm region. Note that CSITA014 contains the Csect CSITA013 (generated by a TOTEXIT macro), therefore CSITA014 must be assembled using the TOTAL macro library, and then included with the TOTAL module in a manner that will override the default CSITA013 Csect within TOTAL.
- DD statements for 5 preallocated single extent TOTAL log data sets corresponding to the 5 Intercomm checkpoint records and managed (flip/flopped in wraparound sequence) by CSITA014. Whenever an on-line Intercomm checkpoint causes a QUIET record to be written to the log, CSITA014 flips to the next TOTAL log data set to record the Intercomm requested QUIET and MARKL. After a flip, the previous log data set is reset for overwriting in CSITA014 when needed. The ddnames must be TOTLOG01-5, replacing the standard TOTAL log data set.

- A catch-all preallocated TOTAL log data set with ddname TOTLOGRS which is written to by CSITA014 only during off-line data base backout processing. This prevents overlaying the checkpoint coordinated TOTAL log data sets (TOTLOG0n) during backout.

- DD statement for the TOTLOGSW data set (BDAM - one 20-byte record to record the ddnames of the previous and current TOTLOG0n data set being used by CSITA014). This file is created by the AUTORSET utility described in the ORM, Chapter 12, and using the ddname TOTLOGSW (instead of STRTUPSW and with a different data set name).

In addition, there are two other Intercomm programs to execute before executing TOTRSEX (PMITOTRS) to back out the TOTAL data base when a restart is necessary. The first is the LOGMERGE utility (see ORM, Chapter 12) to merge the most recent (and previous, if not empty) Intercomm INTERLOG or INTERLOC (if flip/flop used) disk log data set to the master log data set (DAILYLOG with ddname LOGOUT) to be used as the RESTRTLG data set during on-line message restart. The second is the utility TOTAREOF which calls ICOMFEOF (see ORM, Chapter 12) to recover the EOF for the most recently used TOTAL log data set (TOTLOG0n) as determined from the TOTLOGSW data record by TOTAREOF.

Following is a sample JCL stream (add STEPLIB DD statements) to execute the programs required for automated TOTAL restart. Note that each step (except TOTUNLK) contains a DD statement for the on-line Automated Restart STRTUPSW data set (see ORM, Chapter 9) which contains the completion status of the last Intercomm execution: STARTUP if Intercomm closed down, or RESTART if automated restart processing is needed. This record also will contain the checkpoint time to which the TOTAL data base is backed out and which is subsequently retrieved automatically by DBRSTRT during on-line restart via a call to the AUTORDBR entry in AUTORCVR.

```
//jobname  JOB  (accounting,data),'programmer.name',etc. ...
//*
//* SAVE THE PRIOR EXECUTION'S INTERLOG(S)
//*
//LOGMERGE EXEC PGM=LOGMERGE
//INTERLOG DD   DSN=this.regions.INTERLOG,DISP=SHR
//INTERLOC DD   DSN=this.regions.INTERLOC,DISP=SHR
//LOGOUT   DD   DSN=this.regions.DAILYLOG,DISP=MOD
//STRTUPSW DD   DSN=this.regions.STRTUPSW,DISP=SHR
//*
//* FORCE AN EOF ON THE APPROPRIATE TOTLOG DATASET
//*
//TOTLFEOF EXEC PGM=TOTAREOF
//STRTUPSW DD   DSN=this.regions.STRTUPSW,DISP=SHR
//TOTLOGSW DD   DSN=this.regions.TOTLOGSW,DISP=SHR
//TOTLOG01 DD   DSN=this.regions.TOTLOG01,DISP=SHR
//TOTLOG02 DD   DSN=this.regions.TOTLOG02,DISP=SHR
//TOTLOG03 DD   DSN=this.regions.TOTLOG03,DISP=SHR
//TOTLOG04 DD   DSN=this.regions.TOTLOG04,DISP=SHR
//TOTLOG05 DD   DSN=this.regions.TOTLOG05,DISP=SHR
//*
```

```
//* THIS STEP RECOVERS THE DATABASE TO THE LAST TOTAL CHECKPOINT
//*
//TOTRSEX  EXEC PGM=TOTRSEX,PARM='TOTAL,L'
//STRTUPSW DD   DSN=this.regions.STRTUPSW,DISP=SHR
//TOTLOGRS DD   DSN=this.regions.TOTLOGRS,DISP=SHR      FOR CSITA014
//TOTLOGSW DD   DSN=this.regions.TOTLOGSW,DISP=SHR
//TOTLOG01 DD   DSN=this.regions.TOTLOG01,DISP=SHR
//TOTLOG02 DD   DSN=this.regions.TOTLOG02,DISP=SHR
//TOTLOG03 DD   DSN=this.regions.TOTLOG03,DISP=SHR
//TOTLOG04 DD   DSN=this.regions.TOTLOG04,DISP=SHR
//TOTLOG05 DD   DSN=this.regions.TOTLOG05,DISP=SHR
//CHEKPTFL DD   DSN=this.regions.CHEKPTFL,
                DCB=(DSORG=DA,OPTCD=RF),DISP=OLD
//CTLX     DD   DSN=total.CTLX,DISP=SHR
//xxxx     DD   DSN=total.database,DISP=SHR
//*
//* THIS STEP DOES A BRUTE FORCE UNLOCK OF THE TOTAL DATA FILES
//* THE ASSUMPTION IS THAT THIS IS SAFE SINCE THE PRIOR STEP SHOULD
//* HAVE REPAIRED ANY DAMAGE TO THEM FROM A PREVIOUS ABEND
//*
//TOTLUNLK EXEC PGM=UNLOCK                              CINCOM PROGRAM
//CTLX     DD   DSN=total.CTLX,DISP=SHR
//xxxx     DD   DSN=total.database,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DSN=intercom.CONTROL(UNLKcards),DISP=SHR
//*
//* NOW WE GO ONLINE
//*
//INTERCOM EXEC PGM=intercom[,PARM=optional]
//INTERLOG DD   DSN=*.LOGMERGE.INTERLOG,
                DCB=(RECFM=VB,DSORG=PS,OPTCD=C,NCP=16),DISP=SHR
//STRTUPSW DD   DSN=this.regions.STRTUPSW,
                DCB=(DSORG=DA,OPTCD=R),DISP=SHR
//RESTRTLG DD   DSN=this.regions.DAILYLOG,
                DCB=(RECFM=U,BLKSIZE=6233,DSORG=PS),DISP=SHR
//LOGDISK  DD   UNIT=(DISK,,DEFER),SPACE=(3300,(200,100),RLSE),
                DCB=(RECFM=F,BLKSIZE=3300,DSORG=DA)
//CHEKPTFL DD   DSN=this.regions.CHEKPTFL,
                DCB=(DSORG=DA,OPTCD=RF),DISP=OLD
//DEFLTDDQ DD   DSN=this.regions.DEFLTDDQ,              FOR REQONDDQ
                DCB=(DSORG=DA,OPTCD=RF),DISP=OLD
//PMIQUE   DD   DSN=this.regions.PMIQUE,
                DCB=(DSORG=DA,OPTCD=RF),DISP=OLD
            .
            .   Intercomm and user files
            .
//PMISTOP  DD   DUMMY ***********************************************
//INTERLOC DD   DSN=*.LOGMERGE.INTERLOC,
                DCB=(RECFM=VB,DSORG=PS,OPTCD=C,NCP=16),DISP=SHR
//CTLX     DD   DSN=total.CTLX,DISP=SHR
//TOTLOGSW DD   DSN=this.regions.TOTLOGSW,DISP=SHR
//TOTLOG01 DD   DSN=this.regions.TOTLOG01,DISP=OLD
//TOTLOG02 DD   DSN=this.regions.TOTLOG02,DISP=OLD
//TOTLOG03 DD   DSN=this.regions.TOTLOG03,DISP=OLD
//TOTLOG04 DD   DSN=this.regions.TOTLOG04,DISP=OLD
//TOTLOG05 DD   DSN=this.regions.TOTLOG05,DISP=OLD
//xxxx     DD   DSN=total.database,DISP=SHR
```

Normal (non-recovery) processing flow:

- LOGMERGE reads STRTUPSW and sees that the restart-mode is "STARTUP" so it does not use ICOMFEOF. It reads INTERLOG and INTERLOC, determines their relative sequence based upon the time-stamps in the Intercomm message header, copies them to the end of DAILYLOG in the correct order, and then marks these two logs empty by reseting the EOF pointer in the DSCB. If either input log is empty, only the other is copied; if both are empty, no copying is done.

- TOTAREOF and TOTRSEX both note that the restart-mode is "STARTUP" and return immediately to the operating system.

- TOTUNLK unlocks the TOTAL databases. Since the system is in "STARTUP" mode this is actually unnecessary as the databases should already be unlocked. The user may add an earlier step to make sure that any batch processing that may have occured while Intercomm was down also completed normally. If this check is not made, there is a possiblity that the on-line system would be unaware of a batch ABEND that could have corrupted the databases.

- Intercomm step brings up the on-line (satellite) region. STARTUP3 calls AUTORSTU to determine the restart-mode. AUTORSTU reads STRTUPSW, plugs "STARTUP" into the parameter field as if it had been coded on the EXEC statement, and then rewrites the status record with a restart-mode of "RESTART" in case this job does not terminate normally. The TOTAL databases are opened by TOTSTART, which generates TOTAL log entries. CSITA014 begins its task of writing the log data to the appropriate TOTLOG0n file and updates TOTLOGSW to indicate the "current" TOTAL log dataset. During on-line processing, each Intercomm checkpoint is synchronized with a MARKL record written to the TOTAL log. The preceding QUIET record triggers CSITA014 to advance to the next TOTLOG0n dataset and to update TOTLOGSW accordingly. When Intercomm terminates normally, CLOSDWN3 will call AUTORCLD to update STRTUPSW to "STARTUP" since no database recovery will be required in the next execution of this job.

Restart/recovery processing flow with database recovery:

- LOGMERGE reads STRTUPSW and sees that the restart-mode is now "RESTART". It will then read the first record from INTERLOG and INTERLOC. The log with the higher time-stamp is determined to be the log that was in use when the earlier system failure occured. It then loads ICOMFEOF and calls it with a parameter of the appropriate ddname, and waits for that program to correctly terminate the most recently used Intercomm log. Then LOGMERGE copies the two on-line logs in correct sequence to the end of DAILYLOG and marks INTERLOG and INTERLOC empty (as necessary).

- TOTAREOF reads STRTUPSW and sees that the system is in "RESTART" mode. It then reads TOTLOGSW to obtain the ddname of the TOTAL log that was in use when the earlier Intercomm job terminated, and loads and calls ICOMFEOF with the

appropriate parameter to insure a good EOF on the most recently used TOTLOG0n file.

- TOTRSEX now does the database recovery. ATTOTRS reads STRTUPSW and, based upon the "RESTART" indication, then attaches TOTALMT and calls PMITOTRS. (If attached TOTAL is not used, PMITOTRS checks the restart-mode status from STRTUPSW.) PMITOTRS reads TOTLOGSW to determine the ddname of the TOTAL log to use for recovery and searches it to find the last Intercomm MARKL. If no MARKL was found on that log, it tries to find one on the previous TOTAL log (as determined by TOTLOGSW). If no MARKL is found on the previous log either, PMITOTRS will abend. Sequence checking on the TOTAL log records is done before before-images are applied, and the MARKL and QUIET records are validated for the Intercomm task-id. Once the MARKL is located, PMITOTRS database recovery logic proceeds with before-images being written to the database. When the database has been backed out, PMITOTRS termination logic copies the time from the MARKL record to STRTUPSW (for DBRSTRT) and rewrites that record. During this step, CSITA014 logs on the TOTLOGRS data set (if needed), due to its presence in the step JCL.

- TOTUNLK then unlocks the TOTAL databases since they have been properly recovered.

- Intercomm now begins to execute. STARTUP3 calls AUTORSTU, and that program returns a parameter of "RESTART". DBRSTRT calls AUTORDBR to obtain the checkpoint time that coincides with the MARKL that PMITOTRS has backed the TOTAL databases out to. Single-thread restart saves all Intercomm transactions that must be reprocessed on to a DDQ, and then executes them one at a time to insure that they execute in the same order as they were originally input. When all of the messages are reprocessed, REQONDDQ calls AUTORREQ to advise that the recovery is complete. At this point in time, the database is fully restored to its status when the system failure occured, and Intercomm is now ready to acept new processing.

With this system implemented, the Intercomm Job need only be resubmitted after an Intercomm abend or a CPU crash; no other operator control is needed. Due to the implementation of flip/flopped Intercomm and TOTAL log data sets, their small size greatly reduces the time for EOF recovery processing, thus speeding up system restart. Restart time can be further reduced by linkediting ICOMFEOF with both LOGMERGE and TOTAREOF.

Within this design, the following alternatives are available:

1) TOTAL in a separate region - add DD statements for TOTLOGSW and TOTLOG01-5 to the TOTAL region JCL (DISP=SHR) when CSITA014 is linked with the TOTAL load module, and omit them from the Intercomm EXEC step JCL. Do not link ATTOTRS with PMITOTRS when creating the TOTRSEX load module. Omit TOTAL data set DD statements (except CTLX). Define TOTLOGRS only for TOTAL region (DISP=SHR).

2) Standard Intercomm logging without flip/flop - omit the IXFB37 and USERB37E modules from the Intercomm load module; omit INTERLOC DD statements (LOGMERGE and INTERCOM steps). Also omit LOGOUT DD statement from LOGMERGE (ICOMFEOF executed against INTERLOG data set when needed), if no merge required.

3) If INTERLOG and INTERLOC combined are large enough to hold all the log records for one Intercomm normal execution, they still must be merged so that determining the concatenation sequence for RESTRTLG is unnecessary.

4) If multiple TOTAL logging is not used, ICOMFEOF must be executed against the TOTAL log if a CPU crash occurs, before submitting the Intercomm JCL. Omit the TOTAREOF step, use JCL discussed in Section 7.9 for TOTRSEX, omit TOTLOGSW and TOTLOG01-5 from INTERCOM step, but keep STRTUPSW in all steps.

5) If coordinated checkpointing with batch update jobs is used, use standard TOTAL logging (depending on time across 2 checkpoints) and use a parm override for checkpoint time for TOTRSEX step (see Section 7.9.1). Intercomm JCL as a PROC simplifies operator job submission for this case.

6) If multiple TOTAL logging is not used because a complete log is thought needed to restore the data base from a backup copy, consider the following:

   a) the AUTORSET utility can be used to both reinitialize the STRTUPSW record to STARTUP and recreate the TOTLOGSW data set, allowing the restart Intercomm JCL stream to be brought up in startup-mode despite a previous problem. If used with the (daily) backup copy of the TOTAL data base, only the update transactions would need to be reentered on-line.

   b) use the Intercomm LOGINPUT facility (see ORM, Chapter 8) against the merged Intercomm logs (DAILYLOG) to reprocess all data base update transactions against the backup before allowing new on-line transaction processing within the region (by the update subsystems).

   c) modify CSITA014 to unload the third most recently used TOTAL log to a master log file after a flip is done (adapt from the Intercomm USERB37E exit for Intercomm flip/flop logging). Also develop a TOTAL version of LOGMERGE for the current (after TOTAREOF executes) and previous TOTAL logs (based on the TOTLOGSW data record), to merge them to the master log. Be sure that a previous day's log data is not merged (check log record sequence numbers).

7.3 SYSGEN OF TOTAL

In addition to standard procedures required to sysgen TOTAL as described in CINCOM-supplied publications, the following should be noted:

- If access to the data base from multiple regions (Intercomm or batch) is desired, use of a Cincom-supplied SVC is required. The SVC number is supplied via the SETGLOBE global &TOTSVC for the TOTFILE table assembly and is used only when not operating TOTAL as an Intercomm subtask. Two Intercomm regions may not update the same TOTAL DBMOD if checkpointing and/or automated TOTAL restart is used.

- If TOTAL is to be attached, the TOTAL load module (with entry point TOTALMT) must be on a library accessible via STEPLIB or JOBLIB at Intercomm execution time.

- The module TOTINT contains the TOTTABLE which controls the maximum number of concurrent tasks processed by TOTAL, with a supplied value of fifteen. Cincom documentation must be consulted to alter this value. The Intercomm RESOURCE macro may be used in conjunction with the SYCTTBL macros describing subsystems which access TOTAL in order to limit the maximum number of processing threads which may be started through those application programs (see the Operating Reference Manual).

- The module TOTINT must reside on a library accessible at Intercomm linkedit time.

- Beginning with TOTAL 8, the module DATBAS has been replaced by one of two interface modules, depending upon the operating environment: CSTAMVIC or CSTAMVMT. These modules must be altered by the user via linkedit; for use with Intercomm, the Csect TOTINT in the interface module must be renamed DATBAS via a linkage editor CHANGE statement. This module must then be linkedited with Intercomm along with TOTINT.

- Ensure all outstanding PTFs for TOTAL modules are applied, particularly for the interface modules when executing under MVS/XA.

- Include the supplied CSITA014 user exit with the TOTAL load module for regions where automated TOTAL restart is implemented, and allocate the five TOTAL log data sets as sequential, single extent files (same BLKSIZE, etc. as usually used for the TOTAL log data set).

7.4 CONDITIONAL ASSEMBLIES

As discussed in Chapter 4, the members INTGLOBE and SETGLOBE control conditional assembly of the System Parameter Area (SPALIST macro). Globals specified in these members also control conditional assembly for the TOTAL File Table (see Figure 7-4).

Module	Global(s)	Condition Necessitating Reassembly
TOTFILE	&TOTDESC	TOTAL data base descriptor name. The setting of &TOTDESC gives the default data base descriptor. This value may be overridden at execution time by an EXEC statement parm option (see Intercomm Region JCL) or at assembly time by coding the name for the DBMODNM parameter on the TOTFLGEN macro.
TOTFILE	&TOTSVC	Type II SVC number used when TOTAL operates in a separate region from Intercomm.

Figure 7-4. TOTAL Conditional Assembly Requirements

A change to any global described above necessitates reassembly and linkedit of TOTFILE.

The member SETGLOBE on SYMREL contains the following global default settings:

```
&TOTDESC    SETC    'XXXXXX'
&TOTSVC     SETC    'NUL'
```

If any of these default settings are not applicable to your installation, the correct setting should be assigned in the member SETGLOBE on SYMLIB before the TOTFILE table is assembled. Note that as of Intercomm Release 9.0, the global &TOTMOD is obsolete; determination of execution mode (attached or separate region) is based on coding of the TOTATT parameter of the specific SPALIST macro module linked into each Intercomm region.

The functions of these globals are:

- &TOTSVC

  defines the SVC number assigned to the TOTAL interregion SVC supplied by Cincom. This global is used only if TOTAL is not operating as a subtask of Intercomm. For example, if the TOTAL interregion SVC number is 251, the following statement must be defined in SETGLOBE:

```
&TOTSVC     SETC    '251'
```

- &TOTDESC

  defines the default TOTAL data base descriptor name for all Intercomm regions. It must be set to a six-character string. This value may be overridden for a specific region via the TOTFILE table or at Intercomm startup; see Intercomm Region JCL. For example, if the default name of the TOTAL data base descriptor is TOTXXX, the following statement is defined in the member SETGLOBE:

```
&TOTDESC    SETC    'TOTXXX'
```

  Note that if this global is not changed, the default data base descriptor will be XXXXXX.

7.5 Intercomm Region Tables

In addition to the Intercomm table entries and parameter changes described in Chapter 4, the following table entries affect Intercomm when TOTAL support is implemented:

- TOTFILE Table--required in every Intercomm on-line region interfacing to TOTAL in order to define all TOTAL files accessed by on-line subsystems.

- Front End Verb Table--(Intercomm control region only if a Multiregion system used) requires entries if a terminal operator is to be allowed to enter checkpoint or couple/uncouple commands. See Chapter 5 and section 7.2.10.

- SPALIST macro assembly member (default module name is INTSPA)-- the TOTATT parameter indicates to TOTSTART whether TOTAL is executing as an attached subtask (YES) or in a separate region (NO). The default is YES. Must be specifically coded for each region accessing TOTAL.

7.5.1 TOTFILE Table

The user must supply the names of all the TOTAL data sets to be accessed by the on-line system. This is done by providing a TOTFILE table module. The table is created using one Intercomm-supplied TOTFLGEN macro which also generates the module TOTFILE CSECT and END statements at assembly time. The format of the TOTFLGEN macro as provided for Intercomm Release 8 and for downward compatibility under Intercomm Releases 9 and 10 is:

[blank]	TOTFLGEN	[VARIENT=(vvvv[,...,vvvv])] [,MASTER=(mmmm[,...,mmmm])]

where:

vvvv--specifies the name(s) of any TOTAL variable entry data sets accessed during Intercomm execution.

mmmm--specifies the name(s) of any TOTAL master data sets accessed during Intercomm execution.

Figure 7-5 illustrates creation of the TOTFILE module on the library INT.MODUSR for one master data set, MAST, and two variable entry data sets, VAR1 and VAR2 (Intercomm Release 8 and/or TOTAL 5 or 6 parameters).

```
//        EXEC  ASMPCL,Q=USR,LMOD=TOTFILE
//ASM.SYSIN DD *
*   THE CSECT TOTFILE WILL BE GENERATED BY THE MACRO TOTFLGEN
*   MASTER OR VARIENT MAY BE OMITTED IF NOT APPLICABLE.
          TOTFLGEN VARIENT=(VAR1,VAR2),MASTER=MAST
```

Figure 7-5. Sample TOTFILE Table Assembly

The format of the TOTFLGEN macro under Intercomm Releases 9 and 10 is:

[blank]	TOTFLGEN	[DSECT={YES}] [{NO }] [,DBMODNM=name] [,CONTIN={YES}] [{NO }] logging options: [,LGSONOF={A}] [,LGMODFN={A}] [{U}] [{U}] [{N}] [{N}] [,LGBEFOR={N}] [,LGAFTER={A}] [{B}] [{N}] file names (old style): [,VARIENT=(dsname[,dsname...])] [,MASTER =(dsname[,dsname...])] file names (new style): [,ALL={READ}] [{IUPD}] [{SUPD}] [{EUPD}] [,READ=(dsname[,dsname...])] [,IUPD=(dsname[,dsname...])] [,SUPD=(dsname[,dsname...])] [,EUPD=(dsname[,dsname...])]

Figure 7-6. TOTFLGEN Macro Parameters

Note: if none of the parameters to define file names are coded, then TOTSTART bypasses issuing open requests. That is, it is assumed that the files to be accessed via Intercomm have already been opened in a previous step (batch job). This option can only be used for a Central Version of TOTAL. File names must be defined is using attached TOTAL.

DSECT
specifies whether the TOTFILE table or a Dsect of the fields in the table is being generated. For a Dsect, code YES. The default is NO.

DBMODNM
may be used to provide a specific six-character Data Base Descriptor Name for assembly of this table. If not coded, the default name coded for the &TOTDESC global in Intercomm's SETGLOBE will be used. This value may be overridden at execution time as described in section 7.7.

LGAFTER
specifies whether TOTAL logging of after-images of data base access is to be performed. A specifies yes, while N (default) specifies no. A must be coded for restart/recovery if on-line updates are executed.

LGBEFOR
specifies whether TOTAL logging of before-images of data base access is to be performed. N specifies no, while B (default) specifies yes. B must be coded if LGAFTER=A is also coded.

LGMODFN
specifies whether and which function executions are to be logged by TOTAL. A indicates log all functions, U indicates log functions only if they result in data base updates, while N (default) specifies do not log any user access functions.

LGSONOF
specifies whether or not TOTAL logging of SINON and SINOF commands is to be performed. A specifies that all such commands are to be logged, U specifies that such commands are to be logged only for tasks signing-on in update mode, while N (default) specifies no logging of these commands.

Note: logging options must match the LOGGING parameter coded for the gen of the TOTAL DBMOD or as a TOTAL region startup execution parameter.

MASTER
specifies a list of one or more four-character TOTAL file names to be opened as master data sets via an OPENM command. Code as a list enclosed in parentheses.

VARIENT
specifies a list of one or more four-character TOTAL file names to be opened as variable entry data sets via an OPENV command. Code as a list enclosed in parentheses.

Note: if VARIENT and/or MASTER is coded, the new file definition parameters below may not be used. A maximum of 50 files may be defined for VARIENT and/or MASTER.

ALL

allows the user to instruct TOTAL to open (via OPENX) all the files present in the DBGEN without explicitly naming them and indicates what type of file activity is to be allowed: READ (inquiry only), or IUPD (inquiry and update), or SUPD (shared update), or EUPD (exclusive control update). If ALL is coded, none of the other file naming parameters may be coded.

READ

specifies a list of one or more four-character TOTAL file names to be opened via OPENX for inquiry-only (RDONLY) processing. Updates against these files will be rejected by TOTAL.

IUPD

specifies a list of one or more four-character TOTAL file names to be opened via OPENX for inquiry and update processing.

SUPD

specifies a list of one or more four-character TOTAL file names to be opened via OPENX for shared update processing.

EUPD

specifies a list of one or more four-character TOTAL file names to be opened via OPENX for exclusive update processing.

Notes: the above parameters may only be used for access to TOTAL Release 8 (and upward) file processing. A maximum of 50 files may be defined on each parameter (READ/IUPD/SUPD/EUPD), unless CONTIN=YES is coded.

CONTIN

specifies whether any of the file list parameters (READ, IUPD, SUPD, and/or EUPD) is continued on a subsequent TOTFLGEN macro. If so, specify CONTIN=YES. The last TOTFLGEN macro must have CONTIN=NO (default) to delimit the list after the file list parameters are processed.

7.6 INTERCOMM REGION LINKEDIT

In addition to linkedit requirements for the non-DBMS functions of the Intercomm region (Intercomm programs, tables, service routines, Intercomm and user subsystems, etc.), the linkage editor control cards for TOTAL interface modules listed in Figure 7-7 are required. (The ICOMLINK parameters DBASE, DBLIBR and CHKRES may be used.)

```
The Following Members for both Inquiry and Update
   INCLUDE SYSLIB(TOTSTART)
   INCLUDE SYSLIB(user-TOTFILE-name)
   INCLUDE SYSLIB(PDATBASE)
   INCLUDE SYSLIB(USERPDBE)              PDATBASE user exit-if coded
   INCLUDE SYSLIB(TOTCLOSE)
   INCLUDE SYSLIB(ABTOTEND)              Only if TOTAL ATTACHed
   CHANGE  TOTINT(DATBAS)
   INCLUDE TOTLIB(interface-module-name) Cincom-supplied (DATBAS entry)
   INCLUDE TOTLIB(TOTINT)                Cincom-supplied

The Following Members only if Restart/Recovery Performed
   INCLUDE SYSLIB(DBCHKDSP)              Must precede CHECKPT3
   INCLUDE SYSLIB(CHECKPT3)
   INCLUDE SYSLIB(DBRSTRT)
   INCLUDE SYSLIB(RESTORE3)
   INCLUDE SYSLIB(CHCKPTSS)              (omit if dynamically loaded)
   INCLUDE SYSLIB(LOGPROC,READBACK,INTDBLOK)
   INCLUDE SYSLIB(TOTCHKPT)        only if batch update/checkpointing
   INCLUDE SYSLIB(REQONDDQ)              if Serial Restart desired
   INCLUDE SYSLIB(USRSEREX)              Serial Restart user exit
   INCLUDE SYSLIB(AUTORCVR)              Automated Restart
```

Figure 7-7. Intercomm Region Linkedit: TOTAL Requirements

7.7 INTERCOMM REGION JCL

Both the Intercomm startup and restart JCL for Intercomm TOTAL support must include all DD statements necessary to operate without utilization of TOTAL. Please refer to the Operating Reference Manual for complete details.

In addition to the standard JCL for Intercomm, it is necessary to include a DD statement to identify the TOTAL CTLX. (If TOTAL is a subtask of Intercomm, of course all the TOTAL JCL must then be included with that for the Intercomm region.) As illustrated in Figure 7-8, all TOTAL DD statements must follow the //PMISTOP DD DUMMY statement so that the TOTAL files are not processed by the Intercomm File Handler at startup.

A parameter, DB=xxxxxx (where xxxxxx is a TOTAL data base descriptor name) may be specified in the PARM field of the EXEC statement. This will cause the default TOTAL data base descriptor, coded for the region's TOTFILE table or that named by the SETGLOBE setting &TOTDESC, to be overridden. For example:

```
// EXEC PGM=ICOMEXEC,PARM='STARTUP,DB=TOTLLT'
```

causes the data base descriptor TOTLLT to be passed to TOTAL.

```
//INTERCOMM     JOB     ----
//INTR          EXEC    PGM=ICOMEXEC,PARM='STARTUP,[DB=xxxxxx,]...'
//INTERLOG      DD      DSN=INTERLOG, etc.
                .       REMAINDER OF DD STATEMENTS DEFINING INTERCOMM
                .       AND USER FILES, etc.
                .
//PMISTOP       DD      DUMMY     delimits Intercomm files
//CTLX          DD      (see TOTAL documentation for format)

    (For attached TOTAL, all DD statements defining TOTAL files and
    log must follow and a JOBLIB or STEPLIB statement defining the
    program library containing the TOTAL load module is required.)
```

Figure 7-8. Intercomm JCL Requirements for TOTAL

For additional restart mode JCL, see the Operating Reference Manual. For coordinated checkpoint processing, a DD statement for the Intercomm CHEKPTFL file is required, as described in the Operating Reference Manual. For automated TOTAL restart JCL, see Section 7.2.11.

7.8 BATCH REGION LINKEDIT AND JCL

No special consideration need be made for batch programs using TOTAL which execute concurrently with on-line Intercomm unless those programs update the same TOTAL files as Intercomm subsystems. In this instance, coordination of TOTAL checkpoints for the batch programs and the Intercomm system must be considered. This is accomplished by including the Intercomm-supplied user exit DATBASXT with each affected batch region. No additional Intercomm modules (other than the TOTAL/Intercomm interface module) or JCL are required in the batch region(s). Review Figure 7-1 for Intercomm region requirements if updates are performed by batch programs and the Intercomm region.

7.9 TOTAL BACKOUT UTILITY (PMITOTRS)

The Intercomm-supplied utility PMITOTRS backs up the TOTAL data base to a specified checkpoint and runs as a batch job. The checkpoint to which TOTAL is to be restored is selected by the user and may be any one of the following:

1) The last checkpoint initiated by Intercomm
2) Any checkpoint whose exact time is known
3) The last checkpoint on the TOTAL log, regardless of source.

The selection of which checkpoint to use is indicated to PMITOTRS via EXEC statement PARM options. The following JCL is necessary to run the job:

```
//       EXEC   PGM=TOTRSEX,PARM='TOTAL[,{tttttttt}]'
//STEPLIB  DD   DSN=                      {L       }
//CTLX     DD   (TOTAL control file)
//LOGI     DD   (TOTAL log data set - tape/disk)
//CHEKPTFL DD   (Intercomm checkpoint file)
```

where:

TOTRSEX	is a load module consisting of PMITOTRS and CSTAMVMT with a CHANGE statement as for the on-line linkedit. (DATBAS if not TOTAL 8.)
STEPLIB	contains the load modules TOTFILE (member's name or alias must be TOTFILE) and TOTRSEX.
CTLX	is the TOTAL Control Data Set.
LOGI	is the last TOTAL log data set; if the TOTAL log on tape is not standard label, the DCB BLKSIZE parm must be coded.
CHEKPTFL	is the Intercomm checkpoint file, containing checkpoint records, as described in the Operating Reference Manual. The file is required only if it is desired to restore to the last Intercomm-initiated checkpoint. (See PARM options, below.)
tttttttt	is the checkpoint time (if desired); see below.

7.9.1 Selecting A Checkpoint

To back out the TOTAL data base to the last Intercomm checkpoint, code the PARM field of the EXEC statement as:

```
PARM='TOTAL'
```

To use this option, the DD statement for CHEKPTFL must be in the JCL.

To select a specific checkpoint time, code the PARM field as:

```
PARM='TOTAL,yyyyyyyy'
```

Where yyyyyyyy is the checkpoint time requested. If this method is selected, the CHEKPTFL DD statement is not required.

To back out the data base to the last checkpoint (MARKL record) on the TOTAL log, code the parm field:

PARM='TOTAL,L'

where L is coded exactly as shown. CHEKPTFL is not required.

PMITOTRS may also run in the same region as TOTAL. In this case TOTAL is attached by the module ATTOTRS (on INT.MODREL). To create the executable load module, the following linkedit must be done:

```
//  EXEC  LKEDP,Q=USR,LMOD=TOTRSEX
    INCLUDE SYSLIB(ATTOTRS)
    INCLUDE SYSLIB(PMITOTRS)
    CHANGE  TOTINT(DATBAS)
    INCLUDE DBLIB(CSTAMVIC)     (DATBAS if not TOTAL 8)
    ENTRY ATTOTRS
//DBLIB DD DSN=TOTAL-load-library,DISP=SHR
```

NOTE: The JCL for executing TOTRSEX with TOTAL attached requires, additionally, a JOBLIB or STEPLIB statement referencing the library on which TOTAL resides, and all DD statements defining TOTAL files. The TOTFILE table is loaded by PMITOTRS at execution time and must be in a library defined for STEPLIB.

The following two examples demonstrate how to select the proper checkpoint time when Intercomm is running simultaneously with a batch data base update program. The time which is chosen as input to PMITOTRS is the same time which must be given in reply to the restart WTOR 'ENTER CHECKPOINT TIME REQUEST'. The reply is given in the form 'nnnnnnnn,nnnnnnnn' where nnnnnnnn is the time chosen in these examples.

Example 1:

```
Console Sheet
     INTERCOMM VERSION nn.00 STARTING...
                    .
     CHECKPOINT TAKEN AT 01422222
     JOB A BEGINNING EXEC
     CHECKPOINT TAKEN AT 01433333
                    .
     CHECKPOINT TAKEN AT 01444444
                    .
     CHECKPOINT TAKEN AT 01455555
     *** SYSTEM FAILURE ***
```

Time 01433333 is the correct time to restore to, as this was the checkpoint initiated by the start of Job A via the DATBASXT exit routine. The batch job must be executed again after TOTAL backout completes.

Example 2:

```
Console Sheet
     INTERCOMM VERSION nn.00 STARTING...
                    .
                    .
     CHECKPOINT TAKEN AT 01422222
                    .
                    .
                    .
     CHECKPOINT TAKEN AT 01433333
     JOB A BEGINNING EXEC
                    .
                    .
                    .
          (no checkpoint taken)
          *** SYSTEM FAILURE ***
```

Correct time is 0143333, the last Intercomm checkpoint.

7.10 ON-LINE SUBSYSTEMS

On-line and batch programs can concurrently access and update the same data base. For efficient recovery, certain criteria should be met by on-line subsystems using TOTAL:

- On-line subsystems which are to do data base updates should be short-running programs. This aids in bringing the checkpoint quiesce time down to a minimum. Long-running programs should only interrogate TOTAL files.

- To request TOTAL data base access, call PDATBASE (or entry DBINT), the Intercomm-supplied interface routine which in turn passes the request to TOTAL. Only Intercomm system routines may call DATBAS (entry in CSTAMV..).

Intercomm itself performs some of the functions which would normally be done by a batch program. They are:

- Intercomm sign-on to TOTAL at startup time for the entire task. The TOTAL sign-on call may not be coded in a subsystem. Each time a new thread ID is presented to TOTINT, a new task is automatically added to TOTAL's active task entry list.

- All TOTAL data sets to be used by any on-line program are opened at startup time; OPEN calls are not to be made by an on-line program.

Other subsystem coding and design conventions are:

- The standard TOTAL parameters on the call must always be preceded by the address of the message being processed.
- If reentrant COBOL is being used, the REENTSBS routine code to pass to COBREENT is 84 (for DBINT).
- Programs loaded above the MVS/XA 16meg line under Intercomm Release 10 must call DBINT, or use the DBINT code 84 in REENTSBS, as applicable to the programming language.
- A DEQUE call should be included in each subsystem before returning to the Subsystem Controller.

An example of an Assembler Language subsystem for TOTAL use under Intercomm is provided in Figure 7-9. An example of a reentrant COBOL subsystem is given in Figure 7-10.

7.11 BATCH APPLICATION PROGRAMS

Coding for batch region application programs executing concurrently with Intercomm's TOTAL Interface remains the same as for standard TOTAL operation.

Batch programs which perform data base updates as well as updates to standard OS/VS files should not do any file processing until the on-line Intercomm sign-on call to TOTAL has been done. If either job is restarted, this ensures that both the TOTAL data base and other files are all in the same condition as when originally started.

Intercomm does not provide restart capabilities for batch programs for other than the TOTAL data base; however, a backup of OS/VS files may be taken before running update programs. In the event of a restart, these files should be restored before the batch job is rerun.

```
SA       CSECT
         LINKAGE LEN=WKLEN,PARM=(4),SPA=(12),                    @
                 BASE=(3),MSG=(5),GPREQ=REGA
         USING   MSGHDR,R5
         USING   WORKSECT,R13
           .
           .
           .
*   SET UP CALL PARAMETERS
         MVC     STAT,ASTK
         CALL    DBINT,(MSGHDR,READM,STAT,MAST,CONT,SEGLIST,     @
                 USERAR,ENDP),VL,MF=(E,LIST)
         CLC     STAT,ASTK    WAS I/O OK
         BNE     IOERR        NO-DO ERR. PROCESSING
*   OTHER PROCESSING
           .
           .
           .
*   CLOSE PROCESSING
         CALL    DBINT,(MSGHDR,DEQUE,STAT,ENDP),VL,MF=(E,LIST)
           .
*   FREE INPUT MESSAGE
           .
           .
         RTNLINK ADDR=(13),LEN=WKLEN,RC=0
MAST     DC      C'MAST'
ASTK     DC      C'****'
READM    DC      C'READM'
ENDP     DC      C'END.'
DEQUE    DC      C'DEQUE'
WORKSECT DSECT
         DS      18F          SAVE AREA FOR REGS
LIST     DS      8F           LIST AREA FOR TOTAL CALL
STAT     DS      F            TOTAL STATUS FIELD
CONT     DS      CLn          CONTROL (KEY) HOLD AREA
SEGLIST  DS      CLn          AREA FOR SEGMENT LIST
USERAR   DS      CLn          READ IN AREA
WKLEN    EQU    *-WORKSECT
MSGHDR   DSECT
         COPY   MSGHDRC
         END
```

Figure 7-9. Assembler Language Intercomm/TOTAL Subsystem

NOTE: if the Assembler Language program is linked for dynamic loading above the MVS/XA 16meg line under Intercomm Release 10 (see the Assembler Language Programmers Guide), the constant parameters (MAST, READM, ENDP, etc.) passed to DBINT must first be moved to fields in the dynamic save/work area and use the latter field names for the calls. 31-Amode parameters may not be passed to the 24-Amode DBINT entry in the Intercomm load module.

```
ID DIVISION.

PROGRAM-ID. XXXXX.

ENVIRONMENT DIVISION.

WORKING-STORAGE SECTION.

77  MAST            PIC     X(4)          VALUE      'MAST'.
77  ASTK            PIC     X(4)          VALUE      '****'.
77  READM           PIC     X(5)          VALUE      'READM'.
77  ENDP            PIC     X(4)          VALUE      'END.'.
77  DEQUE           PIC     X(5)          VALUE      'DEQUE'.
77  PDATBASE        PIC     S9(4)         VALUE      +84 COMP.

LINKAGE SECTION.

01  INMSG           COPY    ICOMINMG.
    02 IN-TEXT      PIC     X(...).
01  SPA             PIC     X(4).
01  SCT             PIC     X(4).
01  ICOM-RETURN     PIC     S9(7)         COMP.
01  DWS             COPY    ICOMDWS.
    .
    .
    02 CONT         PIC     X(...).
    02 SEGLIST      PIC     X(...).
    02 USERAR       PIC     X(...).
    02 STAT         PIC     X(4).

PROCEDURE DIVISION USING INMSG, SPA, SCT, ICOM-RETURN, DWS.
        .
        .
Process input message and set up areas for TOTAL
        .
        .
    MOVE ASTK TO STAT.
    CALL 'COBREENT' USING PDATBASE, INMSG, READM, STAT, MAST, CONT,
                    SEGLIST, USERAR, ENDP.
        .
        .
Close Processing
    CALL 'COBREENT' USING PDATBASE, INMSG, DEQUE, STAT, ENDP.
    GOBACK
```

Figure 7-10. Reentrant COBOL Intercomm/TOTAL Subsystem

NOTE: If the Intercomm copy member ICOMSBS is copied into the WORKING-STORAGE SECTION, the entry DBINT (code 84) may be used instead of PDATBASE. In this case, special 77 level coding of the PDATBASE code is unnecessary. If the program is linked for dynamic loading above the MVS/XA 16meg line under Intercomm Release 10, see the COBOL Programmers Guide for coding restrictions on passing parameters to routines in the Intercomm load module.

Chapter 8

INSTALLING ADABAS SUPPORT

8.1 INTRODUCTION

In addition to the general requirements for DBMS installation common to all systems, as discussed in Section 5, the following steps are required to utilize Intercomm support for ADABAS:

- SYSGEN of ADABAS Multi-Programming Module (MPM)
- The Intercomm Region Linkedit Considerations
- The ADABAS Region
  - -- Linkedit Considerations
  - -- JCL Considerations
- Batch Region(s)
  - -- Linkedit Considerations
  - -- JCL Considerations
- Restart/Recovery Procedures
- Data Base Backout Utilities
- Coding On-Line Subsystems
- Coding Batch Application Programs
- Coding Interface Programs

Before describing detailed installation procedures, this section presents a brief description of support modules for the ADABAS interface, summarized in Figures 8-1 and 8-2.

8.2 INTERFACE SUPPORT MODULES

The following is a brief description of the modules necessary within the Intercomm region to execute with ADABAS:

- ADASVC50

  This is the Type III SVC utilized by ADABAS for interregion communication.

- DBSTART

  This module is called by the INTERCOMM startup module, STARTUP3. It enqueues upon the INTERCOMM TCB and dispatches GDBSTUP, if present, for checkpoint processing.

- ADALNI50

  This module processes all ADABAS requests issued within the INTERCOMM region.

- DBRELEX

  This routine will be called each time a subsystem terminates normally or abnormally. It will issue a CLOSE to ADABAS for the thread (based on terminal ID). This will release ISNs held by ADABAS. It should not be included if subsystems are passing ISNs to other subsystems or if subsequent messages received by a subsystem are to act upon ISNs retrieved via a previous message. DBRELEX is called only after a "conversational" subsystem (using CONVERSE) has either program-checked or processed the last phrase of conversation.

- DBCLOSE

  This module is called during INTERCOMM closedown and in STAE processing (via entry point DBSTAE). It dequeues the INTERCOMM TCB.

- GDBSTUP

  This module is dispatched to DBSTART. Its function is to wait for indication from ADABAS that a checkpoint prepare (C2) or checkpoint command (C3) has been processed. It formats and queues a message to the checkpoint subsystem and is responsible for activating update subsystems following checkpoint and triggering the next checkpoint.

- DBCHKDSP

  This module is dispatched by INTERCOMM startup if checkpointing is to be done to receive control after the period of time specified in the TCHP parameter of the SPALIST. It initiates checkpoint coordination processing.

- CHCKPTSS

  This is the checkpoint subsystem which quieces update subsystems and issues the INTERCOMM checkpoint via CHECKPT3, writes the checkpoint record and dispatches a checkpoint routine (DBCHKCOM) to issue the ADABAS checkpoint.

- CHECKPT3

  This module checkpoints the INTERCOMM tables.

- DBADACHK

  This module has two entry points, DBCKPREP and DBCHKCOM, which issue C2 and C3 commands to ADABAS.

- RESTORE3

  This module is called during startup to format the checkpoint directories and during restart to restore the INTERCOMM tables to the last checkpoint.

- LOGPROC

  This module reads the log backwards and restarts all messages which updated data bases/files since the last checkpoint.

- DBRSTRT

  This module is dispatched by LOGPROC to validate that INTERCOMM and the DBMS have backed out messages and updates to the same checkpoint. It accomplishes this through the response the console operator gives to a PMIWTOR issued.

Function	Member	CSECT & Entry Points	Residency
Interregion Type III SVC Routine	ADASVC50 **	IGC00nnn nnn is SVC #	SYS1.SVCLIB
Startup Processing	DBSTART**		Resident
Data Base Request Handling	ADALNI50** DBRELEX**	ADABAS DBINT DBRELEX	Resident (Optional)
Closedown Processing	DBCLOSE**	DBCLOSE	Closedown Overlay
Abend Processing	DBCLOSE**	DBSTAE	Resident
Checkpoint Processing	GDBSTUP* DBCHKDSP* CHCKPTSS* CHECKPT3* DBADACHK*	GDBSTUP CHECKPT CHCKPTSS CHECKPTO DBCKPREP DBCHKCOM	Resident Resident Resident/Overlay Transient " Resident
Restart Processing	RESTORE3* LOGPROC* DBRSTRT*	 LOGPROC DBRSTRT	Startup Overlay Startup Overlay Startup Overlay

* INTERCOMM supplied on system release tape.
** DBMS Vendor supplied. The source for these modules resides on ADABAS.SOURCE.

Figure 8-1. Interface Modules--INTERCOMM Region

NOTE: Those modules listed under checkpoint and restart should not be included in the INTERCOMM linkedit if recovery procedures are not being utilized (i.e., only inquiries to the data base are being made on-line).

Function	Member	CSECT & Entry Points	Residency
INTERREGION SVC Type III Routine	ADASVC50**	IGC00nnn nnn is SVC #	SYS1.SVCLIB
Data Base Request Handling	ADALNK50**	ADABAS	Resident

* INTERCOMM supplied
** Vendor supplied. The source for these modules resides on ADABAS.SOURCE.

Figure 8-2. Interface Modules--Batch Region

8.3 SYSGEN OF THE ADABAS MPM

The SYSGEN considerations for the use of ADABAS with INTERCOMM are only concerned with the MPM. The tape distributed by Software A.G. contains a partitioned data set ADABAS.SOURCE. The member DOCMPM50 on this data set contains all the directions necessary for generating the MPM for operation with INTERCOMM. This member may be printed out using the following JCL:

```
//            EXEC    PGM=IEBGENER
//SYSUT1      DD      DSN=ADABAS.SOURCE(DOCMPM50),DISP=SHR
//SYSUT2      DD      SYSOUT=A,DCB=(RECFM=FA,BLKSIZE=80)
//SYSPRINT    DD      SYSOUT=A
//SYSIN       DD      DUMMY
```

The instructions in this member demonstrate how to select the appropriate options for your installation. The instructions for the ADABAS MPM which are provided here merely emphasize those features which are necessary to interface with INTERCOMM.

The following steps must be taken:

- Select the appropriate MPM processor of the three available. If only inquiry processing is necessary and no checkpoint/restart procedures are to be utilized, the module ADAMP150 may be used. If synchronized checkpoint/restart procedures are to be used currently or in the future, ADAMP250 should be selected.

- The ADABAS Type 3 SVC must be assembled and linkedited as specified in DOCMPM50. This SVC is used only for ADABAS code. The Intercomm interregion SVC IGC250 is not used for interregion communication with ADABAS. (The Intercomm interregion SVC IGCICOM may still be required for certain VS features and for the Multiregion version of Intercomm.)

- The MPM must be linkedited after all appropriate options have been selected.

- The appropriate parameters to supply in the MPM/ADABAS region must be selected. The critical parameters which may be selected for the MPM execution, with respect to Intercomm, are the following: Checkpoint (CP), Users (NU), CKPTMIN (CL), CKPTMAX (CU), TIMEMIN (TL), TIMEMAX (TU) and TIMEOUT (TO). All these parameters are concerned with the synchronization of checkpoint and some must be coordinated with similar parameters within the Intercomm region:

  - CHECKPOINT

    This parameter must be specified as YES (Y).

  - USERS

    Must be set to the maximum number of user programs which may be accessing the data base through ADABAS at a time. This parameter must be specified for checkpoint.

    NOTE: No matter how many active Intercomm/ADABAS subsystems there are, Intercomm counts as only one user.

  - CKPTMIN

    Must be specified as a decimal integer. This number is the minimum number of updates which must have been done before checkpoint synchronization can take place.

  - CKPTMAX

    This must be specified as a decimal integer. When the number of successful ADABAS updates has reached this number, checkpoint synchronization will begin if TL seconds have elapsed.

— TIMEMIN

Must be specified in seconds. Checkpoint synchronization will not be started until this much time has elapsed.

— TIMEMAX

Must be specified in seconds. Checkpoint synchronization will begin if this much time has elapsed and at lease CL number of updates to ADABAS have been done. (CL=update count lower limit.)

— TIMEOUT

Must be specified in seconds. This is the maximum amount of time that ADABAS will allow checkpoint synchronization to continue. Non-TP programs which have received a 05 response code and have not answered with a C3 (checkpoint command) will be timed out, that is, they will not be included in the checkpoint.

To set the parameters to efficient values, it is necessary to understand the communication between Intercomm and ADABAS during checkpoint.

1. Intercomm, during startup, dispatches a routine CHECKPT (DBCHKDSP load module) to receive control in the number of seconds specified in the TCHP parameter in the SPALIST.

2. When DBCHKDSP receives control, it calls DBCKPREP (load module DBADACHK).

3. DBCKPREP calls ADABAS with a C2 command. (This requests that checkpoint synchronization be entered.)

4. ADABAS will respond to the C2 command from Intercomm only after the Batch region has quiesced.

5. When ADABAS does respond with a 0 to the C2, GDBSTUP receives control via an ECB being posted and formats a message to the checkpoint subsystem (CHCKPTSS).

6. CKCKPTSS quiesces all update subsystems, takes an Intercomm checkpoint and dispatches DBCHKCOM.

7. DBCHKCOM then sends the C3 command to ADABAS which causes that region to checkpoint. ADABAS will respond with a zero response code when checkpointing is complete.

8. When the checkpoint is complete GDBSTUP becomes active via an ECB post.

9. GDBSTUP then marks update subsystems schedulable again and redispatches DBCHKDSP on a timer interval.

Since the C2 command sent to ADABAS by Intercomm will not be posted until the batch region has been synchronized, it is necessary to force batch synchronization prior to Intercomm's issuing of the C2 command. It is suggested that CL (update count lower limit) and TL (time lower limit) be set to zero, so that any C2 received from a batch region will initiate batch synchronization.

In a TP environment it is desirable for the on-line monitor to control the initiation of checkpoints. Therefore, TU (time upper limit) should be set to a number of seconds slightly greater than TCHP so that when Intercomm issues a C2, ADABAS will have already initiated batch synchronization and will be ready to respond to the Intercomm C2 command. To preclude the possibility of ADABAS checkpointing when Intercomm is not ready, CU (update count upper limit) should be set to some high value. Also, to allow the TP monitor complete control over the issuing of checkpoints, it is suggested that ADABAS batch update programs not be alllowed to issue C2 commands.

The remaining parameter which must be coordinated with Intercomm is the TO parameter or the checkpoint timeout value for ADABAS. This value should be greater than the timeout value specified in the SYCTTBL macro defining CHCKPTSS (TCTV).

8.4 INTERCOMM REGION LINKEDIT

In addition to linkedit requirements for the non-DBM functions of the Intercomm regions (Intercomm programs, tables, service routines, Intercomm and user subsystems, etc.) the following linkage editor control cards are required in the Intercomm region:

```
INCLUDE    ICOMLIB(DBADACHK)
INCLUDE    ICOMLIB(GDBSTUP)
INCLUDE    ICOMLIB(DBCHKDSP)
INCLUDE    ICOMLIB(CHECKPT3)
INCLUDE    ICOMLIB(RESTORE3)
INCLUDE    ICOMLIB(CHCKPTSS)
INCLUDE    ADALIB(DBSTART)
INCLUDE    ADALIB(ADALNI50)
INCLUDE    ADALIB(DBRELEX)
INCLUDE    ADALIB(DBCLOSE)
```

ICOMLIB and ADALIB are described by the following DD cards:

```
//ICOMLIB    DD    DSN=PMI.MODLIB,DISP=SHR
//           DD    DSN=PMI.MODREL,DISP=SHR
//ADALIB     DD    DSN=ADABAS.MPMLOAD,DISP=SHR
```

where ADABAS.MPMLOAD is a PDS containing the ADABAS load modules.

8.5 THE ADABAS REGION LINKEDIT

Assuming all linkedited modules are on the SYSLMOD (ADABAS MPMLOAD) library:

```
INCLUDE   SYSLMOD(ADAMP050)
                  (ADALOG50)      (OPTIONAL)
                  (ADAXCP50)      (OPTIONAL)
                  (ADANUC)
OVERLAY   A
INCLUDE   SYSLMOD(ADAMIN50)
OVERLAY   A
INCLUDE   SYSLMOD(ADAMP250)
ENTRY     ADAMPM
NAME      ADAMPMEX (R)
```

will create the executable MPM (ADABAS) module.

8.6 SAMPLE ADABAS REGION JCL

```
//          EXEC PGM=ADAMPMEX,REGION=200K,TIME=60,
//          PARM=(' ... see parm list description...')
//STEPLIB   DD  DSN=ADABAS.MPMLOAD,DISP=SHR,UNIT=3330,
//          VOL=SER=XXXXXX
//SYSODUMP  DD  SYSOUT=A
//DDASSOR2  DD  DSN=ASSO,DISP=OLD,UNIT=3330,VOL=SER=aaaaaa
//DDDATAR2  DD  DSN=DATA,DISP=OLD,UNIT=3330,VOL=SER=dddddd
//DDWORKR2  DD  DSN=WORK,DISP=OLD,UNIT=3330,VOL=SER=wwwwww
//DDSIBA    DD  LABEL=(,SUL),UNIT=2400-3,VOL=SER=tttttt
//          DSN=RESTART,DISP=(,KEEP),
//          DCB=(RECFM=V,BUFNO=1,BLKSIZE=3124)
//DDPRINT   DD  SYSOUT=A,DCB=BLKSIZE=132
//ADALOG    DD  SYSOUT=A,DCB=BLKSIZE=133(ADALOG REPORT-OPTIONAL)
//ADASNAP   DD  SYSOUT=A (SNAP OF USER BUFFERS ON ERROR -
                          OPTIONAL)
```

8.7 BATCH REGION LINKEDIT AND JCL

Assuming the USERLIB (USERLOAD) data set contains USERPROG, and SYSLMOD(ADALOAD) contains the compiled and linkedited ADALNI50,

```
INCLUDE          USERLIB(USERPROG)
INCLUDE          SYSLMOD(ADALNK50)
ENTRY            USERENT
NAME ADAUSER(R)
```

will create an ADABAS batch user module which may be executed using the following JCL:

```
//EXEC      PGM=ADAUSER,REGION=rrrK
//STEPLIB   DD  DSN=ADALOAD,DISP=SHR,UNIT=3330,VOL=SER=xxxxxx
//SYSUDUMP  DD  SYSOUT=A
        .
        .
        .
// user DD statements
        .
        .
        .
```

8.8 RESTART/RECOVERY PROCEDURES

As stated in Section 3, there are four types of data base failure conditions requiring recovery procedure. The INTERCOMM/ADABAS support provides the ability to recover from all of these failures by, in brief: requiring periodic dumps of the data bases for backout purposes; logging all ADABAS updates; and, while in execution, utilizing the following restart procedures.

In the case of partial or full, logical or physical loss of a data base, the data base must be restored from the latest backup tape.

- The ADABAS utility ADAFIX performs the dump/restore functions (see execution instructions under ADAFIX in the ADABAS Utilities Manual).

- Any ADABAS log tapes which are produced subsequent to the last backup tape should then be used to apply after-images to the data base.

In all other cases, failure of batch programs, failure of INTERCOMM, or both, these steps must be executed.

A. Before executing the ADABAS WARMSTART, the ADABAS region, if still operational, must first be terminated by responding to the outstanding ADABAS WTOR with ADAEND.

B. If the ADABAS log tape (DDSIBA) has not been properly closed (as in the case of a hardware or operating system failure), it will be necessary to run the COPY function of the ADARES Utility (see execution instructions under ADARES Copy Function in the ADABAS Utilities Manual) before WARMSTARTing the data base.

C. In any case the WARMSTART function of the ADARES Utility should be run restoring the data base to the last previous sync checkpoint.

D. The ADABAS region must then be brought up again.

E. An INTERCOMM RESTART must then be initiated.

NOTE: ADABAS must be executing before starting INTERCOMM. During INTERCOMM restart processing, a WTOR to the console requesting the checkpoint time will be issued (PMIWTOR ID DB601R). The operator should reply with the time specified in the PMIWTO ID PMIDB5701 immediately preceding the ADABAS checkpoint notification WTO.

Example:

Console Sheet

```
PMIDB507I CHECKPOINT TAKEN AT 12345678.  ALL USERS
PARTICIPATED BLOCK ID (0567).  ADABAS SYNCHRONIZED
CHECKPOINT TAKEN -
PMIDB570I CHECKPOINT TAKEN AT 23456789
     .
     .
     .
SYSTEM CRASH
```

In this case, the system went down in the middle of a coordinated INTERCOMM/ADABAS checkpoint. ADABAS must be backed out to the sync point at block ID 0567. The correct response to the WTOR DB601R is 12345678,12345678. The checkpoint taken at 23456789 does not have a corresponding ADABAS checkpoint and therefore cannot be used. All ADABAS batch update programs exeucting at the time of the failure would have to be restarted from sync point 0567 as well.

F. Any batch update programs which were active at system termination must be restarted from the last synchronized checkpoint.

8.9 ADABAS BACKOUT UTILITIES

The ADABAS utility ADARES performs a number of functions. The following JCL may be used for a WARMSTART of data base, i.e., restoring to a sync point.

```
//          EXEC   PGM=ADARES,REGION=110K,TIME=60
//STEPLIB   DD     DSN=ADABAS.LOAD,DISP=SHR,UNIT=3330,VOL=SER=xxxxxx
//SYSUDUMP  DD     SYSOUT=A
//DDDRUCK   DD     SYSOUT=A,DCB=BLKSIZE=132
//DDBACK    DD     DSN=RESTART,DISP=OLD,UNIT=2400-3,
//                 VOL=SER=tttttt,LABEL=(,SUL),
//                 DCB=(RECFM=U,LRECL=3620,BLKSIZE=3624)
//DDASSOR2  DD     DSN=ASSO,DISP=OLD,UNIT=3330,VOL=SER=aaaaaa
//DDDATAR2  DD     DSN=DATA,DISP=OLD,UNIT=3330,VOL=SER=dddddd
//DDWORKR2  DD     DSN=WORK,DISP=OLD,UNIT=3330,VOL=SER=wwwwww
//DDKARTE   DD     *
WARMSTART
/*
```

For the additional RESTART functions (COPY, BACKOUT, REGENERATE, REPAIR), see the ADABAS Utilities Guide for execution details.

8.10 CODING ON-LINE SUBSYSTEMS

Coding on-line subsystems requires no special techniques when using ADABAS; however, the nature of an on-line system requires unique design considerations.

In addition to following the general system design criteria in Section 4, the following ADABAS commands/procedures must be used with care:

- The use of a HI command with ADABAS allows an application program to hold an ISN indefinitely. This command should not be used in an on-line environment since a subsystem may program-check or time-out after issuing a HI command, locking out all subsequent subsystems to access of that ISN. (If the subroutine DBRELEX is included in the Intercomm region, a close will be issued for the terminating thread freeing that ISN).

- The DBRELEX function of issuing a close to ADABAS at subsystem termination time safeguards against application programs inadvertently holding ISNs indefinitely if they should program-check or time-out after issuing the close to ADABAS. If this feature is being utilized at an installation, terminals being utilized for ADABAS functions must be used only conversationally; that is, no new input may be sent from a terminal until a response to the previous message has been received. This is accomplished through the use of the Front End Conversational Terminals feature for verbs going to ADABAS subsystems (not to be confused with the CONVERSE facility). The reason this is necessary is that ADALNI50 associates a thread with a terminal-ID. If two messages from the same terminal should process at the same time, ADALNI50 would not be able to distinguish one thread from the other; therefore, a close might inadvertently be done for a message in process.

8.11 CODING BATCH APPLICATION PROGRAMS

Batch application programs for ADABAS Checkpoint/Restart requires the following:

- Issuing an open (OP) command with a record buffer indicating file numbers to be accessed and updated (e.g., RB='ACCESS=nnn...,UPDATE=nnn...').

- The ability to respond to a response code=5 (meaning: command ignored; prepare to checkpoint by issuing a C3 command as soon as possible).

- Optionally, the ability to issue a C2 (request for synchronized checkpoint) and the ability to handle a response code=4 (too soon to request a checkpoint) - in case the checkpoint "window" has not been entered.

- The ability to issue a C3 command (ready to checkpoint) in response to a response code=5 or a response code=0 on a C2 command.

- Issue a close (CL) command prior to termination of the batch application.

Chapter 9

INSTALLING IDMS SUPPORT

9.1 INTRODUCTION

The installation considerations described in this chapter apply to IDMS Release 5.0 and up.

IDMS is a product of the Cullinane Corporation. IDMS product information and manuals may be obtained from:

Cullinane Corporation
Wellesley Office Park
20 William Street
Wellesley, MA 02181

In addition to the general requirements for DBMS installation common to all systems as discussed in Chapter 4, the following items are required to utilize Intercomm support for IDMS.

- SYSGEN of IDMS
- The Intercomm Region
  - Tables
  - Linkedit Considerations
  - JCL Considerations
- The IDMS Region
  - Linkedit Considerations
  - JCL Considerations
- Batch Region(s)
  - Linkedit Considerations
  - JCL Considerations
- Restart/Recovery Considerations
- Data Base Back-out Utilities
- Coding On-Line Subsystems

Before describing detailed installation procedures, this section presents a brief description of support logic and associated programs for the IDMS interface, summarized in Figures 9-1 and 9-2.

Function	Member	CSECT/ENTRY	Residency
Interregion Type 1 SVC Routine	IGCxxx**	IGCxxx	SYS1.SVCLIB
Start-up Processing	IDMSINTI***	DBSTART	Resident
Data Base Request Handling	IDMSINTI**	DBINT DBRELEX	Resident
Closedown Processing	IDMSINTI***	DBCLOSE	Resident
Message Restart Processing	LOGPROC*	LOGPROC RESTORE	Startup Overlay
ATTACH Processing	IDMSATCM**	IDMSATCM	Resident

*Intercomm suppiled on the System Release tape
**DBMS Vendor-Supplied. Load and Source library member names are those suggested by the vendor.
***User generated via DBMS Vendor-Supplied macro of the same name.

Figure 9-1. Interface Modules--Intercomm Region

Intercomm checkpoint processing is not used for IDMS restart as checkpointing and data base recovery is a function of the data base.

Load Module Name	CSECT	Entry
user-defined	IDMSINTI	DBINT DBRELEX DBSTART DBCLOSE
IDMSATCM	RHDCATCO	IDMSATCM

Figure 9 Ver --Supplied Load Module--CSECT--Entry Structure

The following modules are provided on the IDMS installation Tape for users who are operating IDMS under Intercomm:

- IDMSINTI

  Macro-generated database interface program that accepts calls from Intercomm subsystems for data base access. This module requests service from the Central Version by posting, if it is in the same region, or by issuing the IDMS SVC if the Central Version is in another region. The Central Version is an IDMS load module identified by the IDMSINTI macro. This load module is described in the IDMS Installation Guide.

- IDMSATCM

  Interface to the Central Version when IDMS is executing as an attached task in the Intercomm region.

9.2 SYSGEN OF IDMS

IDMS may run as an attached subtask of Intercomm or as an independent job in another region. It is also possible to have an Intercomm with attached IDMS executing concurrently with an independent IDMS region.

Batch jobs may access the IDMS data base through standard IDMS supplied interregion communication for either attached IDMS or independent IDMS, as shown in Figure 9-3.

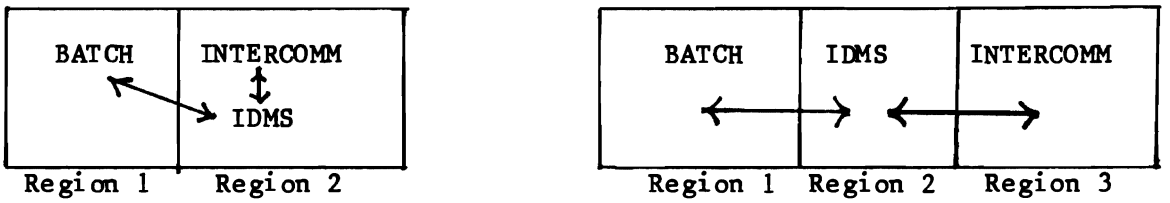

Figure 9-3. Attached IDMS and non-attached IDMS

Using the Intercomm Multiregion Support Facility (MRS), it is possible to have an attached IDMS in any or all of the MRS regions and an optional independent IDMS region.

IDMS BATCH REGION	INTERCOMM CONTROL REGION IDMS a	INTERCOMM SATELLITE REGION1 IDMS b	INTERCOMM SATELLITE REGION2

In, this example, the BATCH, Intercomm Control, and Intercomm Satellite 1 Regions can each access only the IDMS data base within their respective regions.

Intercomm Satellite 2 Region can access only one other IDMS. When the load module of Satellite Region 2 is prepared, the IDMSINTI macro will specify the IDMS (Batch, IDMS a, or IDMS b) with which Satellite Region 2 will communicate.

The IDMSINTI macro is used to specify:

- Whether IDMS is to be attached or execute in a separate region, and which Central Version to use.
- The DDname of the file containing the Central Version control information.
- The SVC number used for interregion communication (0 if nonattached).
- The load module name of the Central Version to be used when running in attached mode.

See IDMS Installation Guide for detailed instructions on coding this macro.

9.3 THE INTERCOMM REGION

9.3.1 Intercomm Region Tables

The SPA, SPA Extension, and SCT described in Chapter 4 are required in the Intercomm region to implement IDMS support:

- SPALIST coding: if the IDMS-access subsystems are dynamically loadable, the value coded for the MAXLOAD parameter must be very generous and should be coordinated with the maximum concurrent threads defined for IDMS.
- Subystem Control Table coding:

  RESOURCE macro coding can be used to control concurrency and force IDMS subsystem enqueues on a common resource (IDMS).

  SYCTTBL macro TCTV value could be at least 15 minutes to prevent premature time-outs and allow for concurrent access to the data base from other regions. TCTV value can be coordinated with DBMS INTWAIT value coded for the Central Version used for the region. Thread concurrency for a subsystem is controlled by the MNCL parameter. DBASE=DB must be defined.

9.3.2 Intercomm Region Linkedit

Figure 9-4 shows the linkage editor statements required for installation of IDMS modules with Intercomm. These are in addition to linkedit requirements for the non-DBMS functions of the Intercomm regions (Intercomm programs, tables, service routines, Intercomm and user subsystems, etc.)

```
     INCLUDE IDMSLIB(IDMSINTI)            IDMSTSKT table
*
     INCLUDE IDMSLIB(IDMSATCM)            ONLY IF ICOM USING
*                                         ATTACHED IDMS
*
*
*
*                                         OPTIONAL:
*                                         IF MESSAGE RESTART
     INCLUDE SYSLIB(LOGPROC)              PROCESSING IS TO
     INCLUDE SYSLIB(RESTORE3)             BE MADE AVAILABLE

     OVERLAY A            STARTUP OVERLAY
         INSERT LOGPROC
         INSERT RESTORE3
```

Figure 9-4. Intercomm Region Linkedit

9.3.3 Intercomm Region JCL

The IDMS library must be defined by a STEPLIB or JOBLIB statement. If Intercomm is executing with an ATTACHed IDMS, then all Central Version JCL for the region must also be defined. With an independent executing IDMS, normal Intercomm JCL is used for the Intercomm region; only the Central Version control file must be defined.

9.4 THE IDMS REGION

See the IDMS Installation Guide for linkedit and JCL considerations.

9.5 BATCH REGION(S)

See the IDMS Installation Guide for linkedit and JCL requirements for an IDMS BATCH Region.

9.6 RESTART/RECOVERY CONSIDERATIONS

9.6.1 Checkpoints

Intercomm checkpoint processing is not used with IDMS. All IDMS checkpointing is a function of the data base.

9.6.2 Restart/Recovery

The IDMS warm start facility is used in conjunction with the IDMS journals to automatically reset the IDMS data base.

If Intercomm is brought up in RESTART mode messages will be restarted (if logged and requested via SYCTTBL RESTART parameter) as follows:

- Completed messages are not processed.
- Messages in process are reprocessed.
- Messages that were queued for processing are requeued for processing.

For Intercomm Restart JCL Requirements see the Intercomm Operating Reference Manual (GDBWKFL data set is not used).

9.6.3 IDMS 'Backout-on-the-Fly'

After an IDMS thread of Intercomm completes, DBRELEX is called. If the thread failed to issue a 'FINISH' to IDMS, then DBRELEX will signal the IDMS to backout any updates made by the thread.

When a batch job that is signed on to IDMS terminates abnormally, then all its updates are automatically backed out by IDMS. If the Protected Update Usage Mode of IDMS is used, then Intercomm will be locked out of the data base at the 'area' level until the batch job terminates. This will affect on-line response times.

If Shared Update Usage mode is used, then while the batch job is executing, Intercomm will be locked out of the data base at the record level. However, other records within the 'area' may be updated, decreasing the impact on on-line response time. (See Section 9.3.1).

9.6.4 ABEND Processing--For Attached IDMS

If the IDMS subtask abnormally terminates, then Intercomm execution must be stopped, using the NRCD or IMCD commands. The Intercomm/IDMS job must be restarted (in RESTART mode, if message restart is used).

If the Intercomm region abends, restart the Intercomm job (RESTART mode if message restart is used).

9.6.5 ABEND Processing--For Nonattached IDMS

If IDMS abends, it is recommended to perform the following, especially if Intercomm subsystems do not requeue failing messages:

1. Bring down Intercomm (NRCD or IMCD);
2. bring up IDMS;
3. bring up Intercomm (RESTART mode if applicable).

However, traffic to subsystems accessing IDMS may be temporarily halted via Intercomm control commands (see System Control Commands).

If Intercomm abends, bring up Intercomm again (RESTART mode if applicable). IDMS restart is not necessary.

9.7 DATA BASE BACK-OUT UTILITIES

See IDMS Utilities Manual.

9.8 CODING ON-LINE SUBSYSTEMS

For COBOL and PL/I, the IDMS vendor has supplied a CODASYL language preprocessor for coding IDMS requests. Reentrant programming considerations affect placement (in working storage or the linkage section) of data items and control areas for IDMS interface.

For Assembler Language subsystems, IDMS requests are handled by IDMS supplied macros. The library containing the macros should be concatenated after the Intercomm libraries in the SYSLIB DD statement for the subsystem assembly.

See the IDMS Programmers Reference Guide for the applicable language.

10. INSTALLING MODEL 204 SUPPORT

Model 204 is supported with a combination of INTERCOMM-supplied programs (the GDB facility) and CCA-supplied programs. Section 5 of this document describes installation procedures for GDB; it should be reviewed carefully, bearing in mind that the phrase "user-supplied" or "user program" implies a Model 204 interface program supplied by CCA.

Detailed installation procedures for Model 204 and the associated INTERCOMM interface may be obtained from:

Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts 02139

11. INSTALLING SYSTEM 2000 SUPPORT

System 2000 is supported by an MRI-supplied subsystem which processes Natural Language requests entered from a terminal and MRI-supplied interface routines which process Procedural Language requests from a user-coded application subsystem.

Detailed installation procedures for System 2000 and the associated INTERCOMM interface may be obtained from:

MRI Systems Corporation
P.O. Box 9968
Austin, Texas 78766

SPR NO.

232

SYSTEM PUBLICATION REVISION

Title: DBMS Users Guide

Product: Intercomm

Date: 12/88

New or Revised Pages:

Title page, ii-x, Chapter 7

Deleted Pages:

None

Unchanged Backup Pages:

None

Subject of Attached Revisions:

Release 9 updates and Release 10.0 additions.

ISOGON CORPORATION