SPERRY UNIVAC
COMPUTER SYSTEMS

Technical Bulletin

Name: OS/3

Bulletin #6

Order No.: UP-8605.6-R1

OS/3 TECHNICAL BULLETIN

This document provides information on:

IMS 90 MULTI-THREAD

It defines the multi-thread concept, and contains information relating to system control, record locks, file usage and ACTION program design. The information is intended primarily for those individuals within an organization who are responsible for the implementation of a multi-thread IMS 90 system and the design of ACTION programs.

Request additional copies by submitting Sales Help Requisition form (UD1-578) through your local Sperry Univac representative to:

CUSTOMER INFORMATION DISTRIBUTION CENTER (CIDC)
Sperry Univac
555 Henderson Road
King of Prussia, PA 19406

Lists:	Bulletin No.:	Date:
18,19,20,21,75,76,CZ	6 Revision 1	September, 1979

OS/3 TECHNICAL BULLETIN SUMMARY

The following Technical Bulletins are published for the OS/3 system. Current items are identified with an "*" in column one; scheduled items are identified with an "**" in the date column:

SYSTEM	REL.#	DATE	ORDER#	ITEM and DESCRIPTION
*OS/3	4.3	1/78	UP-8605.1	OS/3 Technical Bulletin #1 (This document presents an overview of the UTS 400 support and gives some user guidelines.)
*OS/3	4.3,5.0	3/78	UP-8605.1-A	OS/3 Technical Bulletin #1-A (This update contains page replacements to UP-8605.1.)
*OS/3	ALL	4/78	UP-8605.2	OS/3 Technical Bulletin #2 (This document provides a list of the options that can affect the performance of an OS/3 IMS 90 system.)
OS/3	4.3	1/78	UP-8605.3	OS/3 Technical Bulletin #3 (This document is a User Guide for the UTS 400 CHARACTER PROTECTION MODE available with release 4.3.)
OS/3	5.0	7/78	UP-8605.3-R1	OS/3 Technical Bulletin #3-R1 (This document contains updated guide-lines for the UTS 400 CHARACTER PROTECTION MODE available with release 5.0.)
OS/3	5.0, 5.2, 5.2.1 6.0	11/78	UP-8605.4	OS.3 Technical Bulletin #4 (This document contains information on the use of the 8413 DISKETTE FILE CREATION UTILITY.)
OS/3	5.2	5/79	UP-8605.5	OS/3 Technical Bulletin #5 (This document contains information on the use of DATA UTILITIES for OS/3 Release 5.2.)

OS/3	ALL	12/78	UP-8605.6	OS/3 Technical Bulletin #6 (This document contains information on the use of IMS 90 Multi-Thread.)
*OS/3	ALL	9/79	UP-8605.6-R1	OS/3 Technical Bulletin (This document contains information on the use of IMS 90 Multi-Thread and supercedes the UP-8605.6 Bulletin dated December, 1978.)
OS/3	ALL	3/78	UP-8605.7	OS/3 Technical Bulletin #7 (This document contains information concerning techniques for processing unordered IRAM files.)
OS/3	5.2/ 5.2.1 6.0	5/79	UP-8605.8	OS/3 Technical Bulletin #8 (This document contains information on the use of CHARACTER PROTECTION MODE UTILITY for the UTS 400; this utility is available with Releases 5.2/5.2.1 and 6.0.)
*OS/3	5.2/ 5.2.1 6.0	5/79	UP-8605.9	OS/3 Technical Bulletin #9 (This document contains information on the use of the IBM 3741 MEDIA COMPATABILITY UTILITY for the UTS 400; this utility is available with Releases 5.2/5.2.1 and 6.0.)
*OS/3	ALL	7/79	UP-8605.10	OS/3 Technical Bulletin #10 (This document contains information concerning OS/3 FILE CATALOGING.)

NOTE: Technical Bulletins are issued as they become available, and may or may not be issued in sequential order. Please destroy all copies of OS/3 Technical Bulletins 3, 4, 5, 7, and 8 (UP-8605.3, UP-8605.3 Rev. 1, UP-8605.4, UP-8605.5, UP-8605.7, and UP-8605.8); all the information contained in these bulletins has been incorporated in the appropriate OS/3 user documents.

TABLE OF CONTENTS

TABLE OF CONTENTS

PREFACE

UP-8605.6

REVISION 1

This Technical Bulletin supersedes the UP-8605.6 bulletin dated December, 1978. Please destroy copies of the previously issued document. The 'R1' flag in the outside margin of a page indicates that technical content has been revised.

1. INTRODUCTION

This document is intended to give an insight into multi-thread IMS 90 as implemented in OS/3. It defines the multi-thread concept, and contains information relating to system control, record locks, file usage and ACTION program design. The information is intended primarily for those individuals within an organization who are responsible for the implementation of a multi-thread IMS 90 system and the design of ACTION programs.

It is assumed that the reader has a knowledge of the following Sperry Univac publications:

- IMS 90 Programmers Reference Manual UP-8083
- OS/3 IMS 90 System Support Functions, UP-8364
- IMS 90 Applications User Guide, UP-8614

2. OVERVIEW

The OS/3 IMS 90 system is available in both a single thread and multi-thread version. Single-thread IMS 90 provides low volume applications with low-memory serial message processing while multi-thread IMS 90 provides the large volume application the same consistent services as single thread with concurrent transaction processing requiring a modest memory increase.

The obvious reason predicating any consideration toward use of multi-thread IMS 90 is response time. For the new user, it may be the apprehension that single-thread IMS 90 will not be sufficient to meet the response time required. For the single-thread IMS 90 user this migration is usually apparent in an increase in volume. As volume increases, the need for consistent response times becomes a major concern.

With the decision to use multi-thread IMS 90, the user must be conscious of the affect the application may have on IMS 90 performance.

This paper is intended to assist the potential multi-thread IMS 90 user in attaining favorable results through extended explanations of IMS 90 facilities.

3. MULTI-THREAD IMS 90

3.1 MULTI-THREAD IMS 90 CONCEPT

Multi thread IMS 90 is, briefly, the processing of several user requests concurrently, as opposed to serially. Multi thread IMS 90 provides the capability to process input messages while concurrently processing input/output requests for existing threads and communication output requests for terminating threads.

Multi thread IMS 90 achieves its performance capabilities via four functions:

- Multi tasking
- Overlap of I/O requests
- Concurrent execution of user action programs
- Sticking power

Additional performance improvements are obtained by employing methods that enable IMS 90 to optimize the available system resources and share its own internal resources.

Multi thread IMS 90 is designed to maintain a smooth system balance while processing large volumes of input, and still maintain acceptable response times and provide increased system throughput.

3.1.1 Multi-tasking

Multi-tasking provides multi-thread IMS 90 with the ability to concurrenlty schedule transactions for incoming messages, process output messages for terminating actions/transactions and service outstanding I/O requests for existing threads (actions).

There are four subtasks which are always present in the multi-thread IMS 90 system. The primary (job step) subtask is used mainly for startup and shutdown. The secondary subtask, known as the IMC task, is utilized exclusively by the Internal Message Control (IMC) routines, the ICAM interface for IMS 90. The tertiary subtask, known as the Main Task, is employed for the execution of mainline code of IMS 90: the scheduling and terminating of transactions as well as the execution of user action programs. One or more additional subtasks may be allocated, via the job card, to process I/O requests on behalf of a user action program.

```
                    JOB STEP TASK
                    ------------
                   | INITIAL    |
                   | PHASE OF   |
                   | STARTUP    |
                   |____________|
                          |
                          |
        ------------------|-------------------------
        |                 |             |              |
        |                 |             |              |
    ----------            |         ----------     ----------
   | INTERNAL |           |        |          |   |          |
   | MESSAGE  |           |        |   I/O    |   |   I/O    |
   | CONTROL  |           |        | SUBTASK  |....| SUBTASK  |
   | SUBTASK  |           |        |          |   |          |
   |__________|           |        |__________|   |__________|
                          |
                     MAIN | SUBTASK
                  ----------------
                 | APPLICATIONS   |
                 |  MANAGEMENT    |
                 |________________|
```

FIGURE 3-1: Multi-Thread IMS 90 Tasking Structure

3.1.2 Overlapping of I/O Requests

Because single-thread IMS 90 executes through a single TCB, IMS90 is waited until the completion of each I/O request. In multi-thread IMS 90, a subtask is simply allocated from the available pool, whenever an action (thread) requests an I/O function be performed. After issuing the I/O order to data management, IMS 90 may continue to service other requests from the Main Task, the IMC Subtask or the other I/O Subtasks. Upon completion of the I/O function, the thread will be queued for subsequent execution by IMS 90. The number of subtasks specified on the job card predicates the number of I/O subtasks available during the online execution of IMS 90 (i.e, a value of 5 allows for 2 I/O subtasks, 6 allows for 3 I/O subtasks etc.) A value of 6 is usually sufficient.

3.1.3 Concurrency of Action Program Execution

Multi-thread IMS 90 allows for 2 or more threads to access the same action program concurrently. This feature not only contributes to the efficiency of IMS 90 but also conserves resources which may be employed toward scheduling and execution of additional threads. Details of this feature will be described in Section 7.

3.1.4 Sticking Power

Undoubtedly the most transparent feature of multi-thread IMS 90 is sticking power which allows an action program (non-resident) to remain in main memory until that space is needed for subsequent processing. When an action terminates, the space the action program occupies is essentially de-allocated, but not re-used as long as there is available space in the storage pool to continue the processing of incoming transactions. If this action program is needed again, it can be scheduled without being reloaded.

However, if an action program which is in memory has no potential users, and the space it occupies in the storage pool becomes critical toward the scheduling of a new thread, the action program space will be returned to the storage pool and be available for the scheduling of the new thread.

4. THREAD DEFINITION, COMPONENTS AND CONTROL

4.1 THREAD DEFINITION

In multi-thread IMS 90, a thread, which for simplicity's sake will be defined as a unit of work within the IMS 90 environment, consists of the following attributes:

1. Thread Control Block (THCB)
2. Terminal Control Table (TCT)
3. Activation Record (A/R)
4. User Action Program
5. File I/O Areas

4.2 THREAD COMPONENTS

A thread can be associated with every input message and is active only for the length of the action for which it is assigned. Therefore, it is conceivable that several threads may be allocated and deallocated during the span of a transaction.

4.2.1 Thread Control Block (THCB)

The THCB contains all necessary information pertaining to an action. It contains pointers to all areas of the activation record as well as the action, program and terminal control entries which are active on behalf of a thread.

When a thread is created, it is identified as either routine or urgent in priority.

Threads are ordered in a linked list which are serviced on a round-robin basis. Each thread that is designated as ready receives control when its turn comes and retains control until it must wait on some facility. Subsequently, this action is marked as busy and control is passed to the next thread that is ready. If no action threads are marked as ready, control will be transferred to the initial thread routine to determine if a new thread can be created.

4.2.2 Terminal Control Table (TCT)

Each terminal in the IMS 90 environment has a TCT. The TCT serves as a link between the terminal and the thread which is active on its behalf as well as providing constant terminal status and accumulated message counts.

4.2.3 Activation Record (A/R)

Each thread is allocated a corresponding activation record. This is, in essence, the users' area for any data manipulation. Areas within the activation record are allocated based on values supplied in the ACTION section of the configurator. The A/R is comprised of the Program Information Block (PIB), Input Message Area (IMA), Work Area (WA), Output Message Area (OMA) and the Continuity Data Area (CDA). The areas provide the user with facilities for program status, reception of input messages, temporary work areas, output reply messages to the terminals and essentially a common data area for succeeding actions respectively. An A/R is static for the length of the action for which it exists.

Figure 4-1 graphically depicts the multi-thread IMS 90 A/R in main storage and parameter list at the time an action program is given control.

ACTIVATION RECORD (A/R)	Rel. ADDR.
PROGRAM INFORMATION BLOCK (PIB)	A
OUTPUT MESSAGE AREA (OMA)	B
CONTINUITY DATA AREA (CDA)	C
WORK AREA (WA)	D
INPUT MESSAGE AREA (IMA)	E
DEFINED RECORD AREA (DRA)	F

Reg. 1 AT ACTION PROGRAM ENTRY
(PARAM LIST)

AREA	Rel. ADDR.
PIB	A
IMA	E
WA	D
OMA	B
CDA	C
DRA	F

FIGURE 4-1: Multi-thread IMS 90 Activation Record Layout

4.2.4 User Action Programs

Action programs are loaded upon request, if not already in
memory, or are specified as permanently resident at configuration
time. Non-resident action programs are loaded randomly within
the IMS 90 main storage pool. Resident action programs are
R1 loaded once at startup time and then permanently reside in the
R1 main storage pool immediately preceding the IMS 90 Input Staging
R1 Area.

4.2.5 File I/O Areas

File I/O areas are allocated for each ISAM/IRAM file an action wishes to access before the thread is scheduled. I/O areas are allocated based upon information supplied in the FILES parameter of the ACTION section of the IMS 90 configurator. I/O areas are shareable between threads, implying simply that if an I/O area already exists when a thread is initialized, it will be shared with the existing thread. Dam Relative files are not buffered; therefore, an I/O area is not allocated when the thread is scheduled.

4.3 THREAD CONTROL

Most of the resources employed by a thread are unique and apart
from other active threads in the mix. Therefore, each thread
must, in some way, be uniquely identifiable from other active
threads in the system. To achieve this, IMS 90 utilizes the
unique date and time stamp which ICAM provides with each input
message. If the incoming message indicates the initiation of a
new transaction, the date and time stamp is placed in the
Terminal Control Table (TCT) and corresponding Thread Control
Block (THCB). This unique stamp remains throughout the life of
the transaction regardless of the number of actions involved or
types of succession employed. Since IMS 90 manipulates the
R1 date-time stamp to guarantee uniqueness, the time stamp cannot be
R1 used as a valid time of day. However, as of Release 6.0, a valid
R1 date and time of Action Scheduled is made available in the PIB.

Subsequent to allocation of the THCB and insertion of the date-time stamp, the user A/R is allocated and corresponding addresses are placed in the THCB, any required I/O areas are allocated and their corresponding addresses placed in the File Control Table (FCT), the user action program is loaded (if not resident), the Program Control Table (PCT) address is placed in the THCB, the user input message is edited into the IMA from the Input Staging Area and control is transferred to the user action program at its designated entry point address.

Figure 4-2 shows the multi-thread IMS 90 memory layout.

```
 ------------------------------------------------------------------
|PREAMBLE, TCB's                                                   |
|OPEN FILE TABLE & EXTENT AREA                                     |
|------------------------------------------------------------------|
|IMS 90 LOAD MODULE                                                |
|                                                                  |
|CONTAINS 1) DATA MANAGEMENT MODULES                               |
|            (EXCLUDED IF SHARED DATA MGT)                         |
|         2) DTF's, FCTI, FCT's SIB                                |
|         3) TTT's, TCT's                                          |
|         4) NAMEREC & AUDIT I/O AREA                              |
|         5) IMS 90 CODE                                           |
|MAIN STORAGE SUBPOOL                                              |
|                                                                  |
|CONTAINS FOR EACH ACTIVE THREAD                                   |
|         1) THREAD CONTROL BLOCK (THCB)                           |
|         2) ACTIVATION RECORD (A/R)                               |
|            INCLUDES PIB, IMA, WORK, OMA, CDA & DRA               |
|         3) ACTION PROGRAM (NON-RESIDENT)                         |
|         4) FILE I/O AREAS (EXCEPT DAMR)                          |
|         5) FILE LOCKS, RECORD LOCKS                              |
|------------------------------------------------------------------|
|TASK RELATED CONTROL ENTRIES                                      |
|------------------------------------------------------------------|
|RESIDENT ACTION PROGRAMS and SUBPROGRAMS                          |
|------------------------------------------------------------------|
|                                                                  |
|INPUT MESSAGE STAGING AREA                                        |
|                                                                  |
|------------------------------------------------------------------|
|                                                                  |
|CONFIGURATION TABLES (Transaction id table , Actor Control        |
|                      table, Program Control Table)               |
|                                                                  |
 ------------------------------------------------------------------
```

FIGURE 4-2: Multi-thread IMS 90 Memory Layout

5. LOCK FEATURE OF MULTI-THREAD IMS 90

5.1 LOGICAL LOCKS

The lock feature of multi-thread IMS 90 is provided to maintain the integrity of the users' data files during online execution. It is not the intent of this document to describe the lock mechanism as this is done quite adequately in the IMS 90 Programmer Reference Manual UP-8083 and UP-8614 (as of release 5.2). Instead, a description of when locks are imposed as well as when they are released, is provided to enable users to detect, eliminate and avoid both bottlenecks and deadlocks which may occur in a multi-thread IMS 90 system.

Either of two logical lock options, lock-for-update or lock-for-transaction, may be selected for a Dam Relative, ISAM or IRAM files in the File section at configuration time.

5.1.1 Lock-for-update

For Dam Relative files, the lock-for-update option causes a lock to be imposed for a logical <u>record</u> when the record is retrieved via the GETUP function or added to the file via the INSERT function. For ISAM files, the GETUP and INSERT functions will impose a logical lock for the <u>file</u> being accessed. These locks prohibit access to the record (DAMR and IRAM) or file (ISAM) by other transactions until this lock is released. It does not prohibit further access to the same record or file by the same transaction. These locks are released when one of the following occurs within the transaction which imposed the lock:

1. For the GETUP, the record is updated by means of a PUT or DELETE function. For an INSERT function, the lock is released upon successful return from Data Management.

2. The action in which the lock was imposed or a subsequent action terminates with the TERMINATION-INDICATOR of the PIB set to 'N' (normal transaction termination) or 'A' (voluntary, abnormal transaction termination) or 'S' (same as 'A' with a request for snap) or the transaction involuntarily, abnormally terminates.

3. The Program in which the lock was imposed or a subsequent Program terminates with the TERMINATION-INDICATOR set to 'E' (external successor) or 'D' (delayed internal successor).

The LOCK-ROLLBACK-INDICATOR of the PIB is not applicable for files which have had the lock-for-update option specified since IMS 90, in this case, does not perform online recovery for those files.

5.1.2 Lock-for-transaction

The lock-for-transaction, with DAM Relative and IRAM files, causes a lock to be imposed for a logical <u>record</u> when the record is retrieved VIA the GETUP function or added to the file via the INSERT function. For ISAM files, the GETUP and INSERT will impose a logical lock for the <u>file</u> as well as the <u>record</u> being accessed. These locks prohibit access to the record or file by other transactions until this lock is released. It does not prohibit further access by the same transaction. These locks are released when one of the following occurs within the transaction which imposed the lock:

1. The action in which the lock was imposed or a subsequent action terminates with the TERMINATION-INDICATOR set to 'N', 'A' or 'S' or the transaction involuntarily, abnormally terminates.

2. The action which terminates with the TERMINATION-INDICATOR set to either 'E' or 'D' and the LOCK-ROLLBACK-INDICATOR set to 'H' with pending locks outstanding. The logical file lock imposed by the GETUP is released and a record lock is imposed providing file access to concurrent threads.

3. The action in which the lock was imposed or a subsequent action terminates with the TERMINATION-INDICATOR set to 'E' or 'D' and the LOCK-ROLLBACK-INDICATOR set to 'R'. In this case, only those locks that have been imposed via GETUP function requests, and for which no corresponding PUT or DELETE function requests have been issued, are released.

4. The action in which the lock was imposed or a subsequent action terminates with the TERMINATION-INDICATOR set to either 'N', 'E', or 'D' and the LOCK-ROLLBACK-INDICATOR set to 'O' to cause a) the rollback of all updates performed by this transaction to the previous rollback point, b) the release of all locks active for this transaction, and c) to cause a new rollback point to be established for this transaction.

5. The action in which the lock was imposed or a subsequent action terminates with the TERMINATION-INDICATOR set to either 'E' or 'D' and the LOCK-ROLLBACK-INDICATOR set to 'N' will cause IMS 90 to establish a new rollback point for the transaction. Subsequent requests for file rollback will be effective only to the new rollback point. This assumes that other rollback points are not established later in the transaction by subsequent actions.

Finally, logical file locks imposed for ISAM files cannot be carried from action to action. Any file locks active at action termination will be released:

1. File and record locks will be released for pending updates if the LOCK-ROLLBACK-INDICATOR is set to 'R'.

2. If a SETL was issued, an ESETL will be issued by IMS 90 if it is not done by the action program.

5.2 CONTROL AND RELEASE OF INTERNAL LOCKS

5.2.1 ACTION Program Control

The user action program may, at its own discretion, at action termination specify subsequent lock discipline via the LOCK-ROLLBACK-INDICATOR (LRI) of the PIB. Lock discipline is available only for those files which have specified lock-for-transaction (LOCK=TR) in the FILE Section of the IMS 90 Configurator. Using Lock-for-update (LOCK=UP) does not provide lock carryover across succeeding actions. The default value for the LRI is 'N', which releases all locks previously imposed by this action and establishes a new rollback point.

The holding of locks across actions requires the specification of either 'R' or 'H' for the LRI at action termination.

Specifying the value of 'H' will cause IMS 90 to hold all locks active for this action and any preceding actions within this transaction. The value 'R' in the LRI at action termination will release any pending locks active for this action as well as any pending locks active for previous actions within this transaction which may have been held over. A pending lock is defined as one for which a GETUP was issued but the corresponding PUT or DELETE function was not issued.

If during an action program execution the determination is made that any previous updates are void due to current circumstances; these updates, for the existing action and any previous actions for which locks were <u>held</u>, may be rolled back, to the last rollback point. Specification of 'O' in the LRI at action termination will force all updates performed by this transaction, or to the last established rollback point if one has been established since initialization of this transaction, to be rolled back to their initial status. Further, all locks active for this transaction will be released and a new rollback point established for this transaction, providing the Termination-Indicator is not set to 'N'. As previously stated, the rollback facility is available only for files which have lock-for-transaction specified.

Table 5-1 summarizes the function of each of the LOCK-ROLLBACK-INDICATORS.

TABLE 5-1: LOCK-ROLLBACK-INDICATOR (LRI) Values - (LOCK=TR)

LRI VALUE	FUNCTION
R	Release all pending locks. Pending locks are incompleted function requests (i.e. GETUP w/o corresponding PUT or DELETE function).
H	Hold all locks. This includes pending and complete function locks.
N	Release all locks active to this point imposed by this transaction. Establish a new rollback point for this transaction.
O	Rollback all updates active by this transaction to the last rollback point. Establish a new rollback point for this transaction.

5.2.2 IMS 90 Internal Control

Whenever a lock is imposed by an action, the lock will reflect the date and time stamp of the thread which imposed the lock. Further, a bit-map in the TCT is updated to reflect the file for which the lock was imposed. Subsequent requests to release locks or normal transaction termination, causes this bit map in the TCT to be scrutinized to determine which files (FCT's) should be scanned in locating locks held by this transaction. When scanning each files' lock list entries, the date-time stamp assigned to this transaction is used to identify and release any and all locks which may exist for the transaction in the list being analyzed.

Upon successful action/transaction termination and releasing of existing locks (if not held across actions) the user's output message is scheduled for delivery and all allocated resources are released. If the action has indicated a successor, the terminal remains in interactive-mode and the next incoming message will not initiate a new transaction.

6. FILE USAGE WITH MULTI-THREAD IMS 90

6.1 FILE SHARING

All data files are shareable within the IMS 90 environment thru File Management. ISAM, IRAM, DAM Relative and SAM files are shareable among actions on a function-by-function basis. ISAM and IRAM files may be allocated exclusively to an action for a series of sequential file operations. Files are subsequently deallocated either explicitly by the action or implicitly at action termination.

6.2 FILE LOCKS

Tables 6-1 and 6-2 summarize when locks are imposed and released via function calls.

TABLE 6-1: LOCK-FOR-UPDATE

	FILE LOCK	RECORD LOCK	RELEASE FILE LOCK	RETRIEVE LOCKED RECORD	WILL NOT RE-TRIEVE LOCK-ED RECORD	NO LOCKS IMPOSED
SETL	X(1)					
ESETL			X(1)			
GETUP	X(3)	X(2)			X	
INSERT	X(3)	X(2)	X(3)		X	
PUT/DELETE		X(2)	X(3)			
GET (SEQNTL)				X		X
GET (RANDOM)					X	X

TABLE 6-2: LOCKED-FOR-TRANSACTION

	FILE LOCK	RECORD LOCK	RELEASE FILE LOCK	RETRIEVE LOCKED RECORD	WILL NOT RE-TRIEVE LOCK-ED RECORD	NO LOCKS IMPOSED
SETL	X(1)					
ESETL			X(1)			
GETUP	X(3)	X(2)			X	
INSERT	X(3)	X(2)	X(3)		X	
PUT/DELETE		X(2)	X(3)			
GET (SEQNTL)				X		X
GET (RANDOM)					X	X

1. ISAM and IRAM files
2. All files.
3. For ISAM only.

7. ACTION PROGRAM CONSIDERATIONS

7.1 ACTION PROGRAM DESIGN

One of the prime factors predicating the overall performance and memory size of multi-thread IMS 90 is the initial design of the user action programs. It is imperative that the design of the user action programs receive the utmost attention in any system design. The action program should never be construed as an online batch-type job. It is in the better interests of response times and overall availability of IMS 90 resources that action programs abstain from extensive sequential searches on ISAM or IRAM files or extensive updating of any one or group of files in a single action.

There are relatively few rules which should be considered when designing an efficient multi-thread IMS 90 action program. Several of the following points reflect response time considerations; the remainder directly affect memory sizes which could determine whether or not an extra thread can be scheduled for execution.

Figures 7-1 and 7-2 depict the relationships between a Transaction, Action and Program in the IMS 90 environment. Figure 7-1 indicates a simple relationship where a single action program execution constitutes a transaction, action and program. Figure 7-2 shows a compound relationship where several action iterations make up the total transaction and in the final sequence, multiple action program executions comprise the resulting action.

```
INPUT MESSAGE     |       |      |
PROCESSING        |program|Action|transaction
OUTPUT MESSAGE    |       |      |
NORMAL TERMINATION|       |      |
```

FIGURE 7-1: SIMPLE TRANSACTION/ACTION/PROGRAM RELATIONSHIP

```
INPUT MESSAGE                      |        |       |
PROCESSING                         |        |       |
EXTERNAL SUCCESSION                |PROGRAM |ACTION |
  (Locks held - No Rollback Point) |        |       |
                                                    |
                                                    |
INPUT MESSAGE                               |       |
PROCESSING                         |        |       |
DELAYED INTERNAL SUCCESSION        |PROGRAM |ACTION |
  (No Locks held - Logical         |        |       |
   Rollback Point|                 |        |       |
                                                    |TRANSACTION
                                                    |
INPUT MESSAGE                      |        |       |
PROCESSING                         |        |       |
IMMEDIATE INTERNAL SUCCESSION      |PROGRAM |       |
  (Locks held - No Rollback Point) |        |       |
                                            |ACTION |
                                            |       |
PROCESSING                         |        |       |
NORMAL TERMINATION                 |PROGRAM |       |
  (IMPLIED ROLLBACK POINT)         |        |       |
                                                    |
```

FIGURE 7-2: COMPOUND TRANSACTION/ACTION/PROGRAM RELATIONSHIP

R1 ### 7.1.1 Type of Action Program

R1 The most efficient form of action programs in a multi-thread
R1 environment is the re-entrant (BAL) or shareable (COBOL).
R1 Re-entrant and shareable action programs can provide processing
R1 of several concurrent threads. Serially reusable actions cannot
R1 provide this type of processing because they are self-modifying.
R1 When possible and when there is a choice of action program types
R1 to be used for an IMS 90 application, re-entrant or shared code
R1 should be used. In any case, care should be exercised in the
R1 design of an IMS 90 application to remove the potential of

R1 deadlocks (see Section 8. Deadlock).

R1 Re-entrant and Shareable action programs allow multiple threads
R1 concurrent access. If, for example, thread 1 has program-A and
R1 thread 2 also requests Program-A. Thread 2 will be queued until
R1 program-A issues a function request to IMS 90. At this time
R1 thread 2 will be given use of program-A. When thread 2 performs
R1 a function request to IMS 90, thread 1 will be re-scheduled for
R1 program-A and so on.

7.1.2 ACTION Program Size

Keep the size of action programs as minimal as possible. Never design an action program to do it all. The larger the action program, the fewer number of concurrent threads that can be scheduled. The object of multi-thread IMS 90 is to service as many requests as possible, concurrently. Large action programs tend to not only occupy more memory but also hamper concurrent processing due to being either CPU bound or imposing extended lists of locks for records and/or files which are also needed by other threads. These situations may also lead to extended periods of internal waiting for resources to become available.

7.1.3 I/O Requests

Do as few I/O's as possible. A rule of thumb to follow to provide performance and eliminate extended locking of files and records, is to limit action programs to seven I/O's. Granted, this will not always be the case; but under no circumstances should it become the exception. Long sequential searches on Indexed files (SETL/ESETL) should be avoided at all costs. A limit of 100-200 I/O's should be the maximum for sequential search functions. Extended sequential functions lock out the file from all users and cause not only increased response times but also deadlock situations. If sequential searches are a necessity, random GET's should be employed with the action program incrementing the key each time. This will eliminate file locking and permit access to all users.

7.1.4 Number of Files Accessed

Access as few files as possible. Before a thread can be scheduled, all resources that will be needed for execution must be secured. This includes I/O areas for each file. A lack of available resources will cause a thread to be queued and a smaller thread to be scheduled. If an I/O area needed for a transaction being scheduled has been allocated previously for an existing thread, this I/O area will be shared between the two threads. But this may not always be the case. For example, if a thread to be scheduled needs four separate ISAM or IRAM files whose average blocksize is 2K bytes; an extra 8K bytes of overhead memory exists for this thread, if the files requested are not being used by currently active threads.

7.1.5 Locks

Impose locks only when necessary. The majority of deadlocks that occur in an established multi-thread IMS 90 environment originate from file locks, record locks and the use of serially-reuseable programs.

If the user is performing simple interrogation of a file or a selected group of records within a file, the random GET function should be used instead of the GETUP or SETL sequences. The SETL function will impose file locks for ISAM and IRAM files. The GETUP function will impose file locks for ISAM and record locks for IRAM and DAMR files.

Providing for as many of the above points as possible when designing action programs will assist greatly in attaining and maintaining the desired through-put necessary for any application.

8. DEADLOCKS

8.1 DEADLOCK DEFINITION

The deadlock situaton occurs when Multi-Thread IMS 90 detects an uncorrectable situation within the transaction mix which will indefinitely inhibit the processing of one or more threads due to conflict in availability of necessary resources. Deadlocks result from inconsistencies in user design of the overall application. Whenever a transaction is in design phases, the user must also be cognizant of other transactions which may be active concurrently within the IMS 90 system mix. The design of a transaction should be done in such a way so as not to cause direct conflict with other transactions over available resources.

8.2 DEADLOCK SITUATIONS

The following sections are examples which provide explanations of feasible deadlock situations.

8.2.1 Deadly Embrace - File Availability

Thread 1 issues a GETUP (and imposes a lock) for FILE-A and a subsequent GETUP for FILE B. Thread 2 which is executing concurrently with thread 1, issues a GETUP (and imposes a lock) on FILE-B and issues a subsequent GETUP to FILE-A.

This is more commonly referred to as a deadly-embrace. Both threads will be waited indefinitely because Thread 1 holds the lock for FILE-A which Thread 2 is waiting for, and Thread 1 will be waited for FILE-B which thread 2 holds the lock for. The only resolution is for IMS 90 to cancel one or both threads and allow terminal operators to reenter the transactions.

In order to avoid this situation all action programs should access all files in the same order and insure the PUT or DELETE is issued as soon as possible after the GETUP is performed.

8.2.2 Deadly Embrace - Program Availability

This situation can only occur with serially-reuseable (non-shareable) action programs.

Thread 1 employs program-A which issues a GETUP (and imposes a lock) for FILE-X. Subsequently, a PUT is issued (record is still locked) and succession (regardless of type) is done to program-B. Program-A also elects to carry the record lock over to program-B. Thread 2 has meanwhile been scheduled using program-B which issues a GETUP to FILE-X requesting the same record thread 1 is holding.

This also is a deadly embrace. Thread 1 cannot continue because thread 2 has Program-B. Thread 2 has been queued because the record requested is locked by thread 1.

If program-B were shareable or reentrant, thread 2 would be queued for the record, and thread 1 would proceed through succession and execution of Program-B. When the record lock is released, thread 2 would continue normal processing.

8.2.3 Pending Locks

When using Immediate Internal Succession the user should exercise great care not to leave pending locks when terminating the action program. Lock discipline is not interrogated during Immediate Internal Succession and pending locks normally imply _file_ locks (except for DAMR files). If the user intends to perform Immediate Internal Succession an ESETL, PUT or Delete should be issued for each outstanding SETL and GETUP respectively before action program termination. This will eliminate any unnecessary waiting by other threads for the files being held by this action.

8.2.4 Record Lock

When a record is locked by a thread, regardless of filetype, all subsequent requests for access to that record will be queued.

The most common occurence of this situation is when one or more transaction types update a control record for a file. This should be avoided whenever possible.

These are the common causes of deadlocks. Care should be taken to avoid these situations and thus eliminate any bottlenecks that will hamper processing.

9. MULTI-THREAD IMS 90 MODULES and General Flow

9.1 MULTI THREAD IMS 90 MODULES

The following sections give a brief description, by functional area, of each of the Multi thread IMS 90 modules.

9.1.1 Functional Area - STARTUP

The START-UP modules prepare IMS 90 for on-line execution.

MODULE NAME	REQUIRED VS. OPTIONAL	FUNCTIONS
ZB#START[1]	R	1. READS PARAM CARD 2. CALLS ZQ#START 3. ASSIGNS SECONDARY STORAGE KEY 4. ATTACHES SUBTASKS 5. ATTACHES MAIN SUBTASK STXIT CODE 6. OPENS FILES 7. ESTABLISHES STORAGE POOL 8. LOADS RESIDENT ACTION PROGRAMS 9. CALLS ZB#LOAD TO LOAD ON-LINE PHASE
ZQ#START[2]	R	1. READS CONFIGURATION TABLES FROM NAMEREC 2. PERFORMS NECESSARY LINKAGE BETWEEN TABLES 3. ALLOCATES INPUT MESSAGE STAGING BUFFER AREA
ZC#MOPMT[3]	R	PERFORMS INTERNAL MESSAGE CONTROL ORIENTED START-UP PROCEDURES 1. LINK TO ICAM VIA MOPEN SVC 2. SEND 'IMS READY' MESSAGE TO APPROPRIATE TERMINALS 3. ATTACHES IMC SUBTASK, 4. ACTIVATES STXIT AND OPCOM STXIT CODE IF OPCOM=YES
ZC#IOPEN[3]	R	EXECUTES ACTUAL ICAM MOPEN
ZC#BTCHA[2]	R	PROCESSES // PARAM BA CARD
ZC#BTCHX[3]	O	PERFORMS BATCH ORIENTED INITIALIZATION PROCEDURES

[1] FUNCTIONS 1 - 4 ARE EXECUTED UNDER JOB STEP TASK; 5 - 8 UNDER MAIN SUBTASK.

[2]

EXECUTES UNDER JOB STEP TASK.

3
EXECUTES UNDER IMC SUBTASK.

9.1.2 Functional Area - THREAD SCHEDULING

The THREAD-SCHEDULING modules allocate and deallocate action related resources and control thread execution.

MODULE NAME	REQUIRED vs. OPTIONAL	FUNCTIONS
ZT#TM[4]	R	1. SCHEDULES THREADS AND I/O FOR THREADS 2. CREATES THREADS 3. TERMINATES THREADS 4. WAITS THREADS FOR INTERNAL FACILITY 5. POSTS THREADS WHEN FACILITY AVAILABLE 6. REQUESTS I/O SUBTASK FOR THREAD 7. POSTS THREADS WHEN I/O COMPLETE 8. CONTAINS INTERRUPT TIMER STXIT CODE 9. CONTAINS PROGRAM CHECK START CODE 10. CONTAINS ABTERM STXIT CODE
ZT#IMC[4]	R	AWAKES IMC SUBTASK FOR OUTPUT PROC
ZB#LOAD[4]	R	LOADS IMS 90 PHASES
ZA#AS[4]	R	1. ALLOCATES MAIN STORAGE RESOURCES FOR ACTION VIA ZS#MSM 2. CALLS ZC#RDMT TO BUILD IMA 3. CALLS ZJ#SCHED TO READ DDR AND DETERMINE DRA SIZE 4. CREATES THREAD FOR ACTION VIA ZT#TM 5. CALLS ZA#LOADR TO LOAD PROGRAM 6. CALLS ZF#GEN2 TO READ CONDATA 7. SETS UP SECONDARY STORAGE PROTECTION FOR USER 8. DEALLOCATES MAIN STORAGE RESOURCES FOR TERMINATING ACTION 9. CALLS ZF#GEN2 TO WRITE CONDATA 10. CALLS ZF#GEN2 TO PERFORM FILE MANAGEMENT TERMINATION PROCEDURES 11. REQUESTS OUTPUT PROCESSING

ZS#MSM[4]	R	ALLOCATES AND DEALLOCATES (CONDITIONALLY OR UNCONDITIONALLY) BLOCKS OF STORAGE FROM THE MAIN STORAGE POOL
ZA#LOADR[5]	R	LOADS USER ACTION PROGRAMS
ZQ#IOnnn where: nnn=CONFID OF NETWORK SECTION IN CONFIGURATOR	R	1. CONTAINS THOSE TABLES REQUIRING ASSEMBLY GENERATION A. IMS 90 AND USERR DATA FILE DTFS B. ICAM-IMS 90 SHARED TABLES 2. CONTAINS PREALLOCATED I/O AREAS FOR AUDFILE AND NAMEREC
ZG#MTMSO[6]	R	1. SENDS ERROR MESSAGES TO CONSOLE 2. PRINTS SNAP FOR ABNORMALLY TERMINATED ACTIONS

[4] EXECUTES UNDER MAIN SUBTASK.

[5] EXECUTES UNDER CONTROL OF AN I/O SUBTASK.

[6] EXECUTES UNDER CONTROL OF MAIN SUBTASK AND I/O SUBTASK.

9.1.3 Functional Area - INTERNAL MESSAGE CONTROL (IMC)

The IMC modules control the communications environment (i.e., terminal input and output).

MODULE NAME	REQUIRED VS. OPTIONAL	FUNCTIONS
ZC#IMCMT[3]	R	1. DIRECTS PROCESSING OF INTERNAL MESSAGE CONTROL (IMC) MODULES 2. CONTAINS ALL ENTRY POINTS FROM ICAM AND DETERMINES SUCCESSIVE PROCESSING 3. CONTAINS INTERFACE WITH APPLICATION MANAGEMENT FOR OUTPUT PROCESSING
ZC#I1PMT[3]	R	1. QUEUES INPUT MESSAGE FOR APPLICATION MANAGEMENT 2. PROCESSES REGULAR TERMINAL COMMANDS 3. ALLOCATES AND DEALLOCATES BUFFERS FROM THE INPUT MESSAGE STAGING AREA FOR INPUT AND OUTPUT MESSAGES
ZC#FKYMT[3]	O	PROCESS FUNCTION KEYS
ZC#MTCMT[3]	R	PROCESSES MASTER TERMINAL COMMANDS
ZC#OPCOM[7]	O	ALLOWS SEVERAL MASTER TERMINAL COMMANDS TO BE ENTERED FROM CONSOLE
ZC#ICODE	R	SENDS AUTOMATIC STATUS MESSAGES TO APPROPRIATE TERMINALS UNDER CONTROL OF THE IMC INTERRUPT TIMER STXIT CODE
ZC#RDMT[4]	R	DETERMINES SIZE OF IMA AND MOVES USER-DESIRED INPUT MESSAGE INTO IMA. PERFORMS NO EDITING, GENERAL EDITING AND/OR LOWER CASE TRANSLATION AS SPECIFIED IN ACTION SECTION
ZC#EDMT[4]	O	PERFORMS EXPANDED INPUT PROCESSING ON INPUT BASED ON EDIT RECORD

		CREATED BY OFFLINE EDIT TABLE GENERATOR (ZH#EDT)
ZO#QUTMT[3]	R	1. CONTROLS OUTPUT MESSAGE PROCESSING AT ACTION TERMINATION 2. CONTROLS OUTPUT MESSAGE PROCESSING DURING ACTION VIA SEND COMMAND BY DIRECTING USE OF UNSOLICITED OUTPUT MODULE (ZO#UNSMT)
ZO#UNSMT[3]	O	1. PROCESSES CONTINUED RESPONSES TO ORIGINATING TERMINAL 2. PROCESSES SWITCHED OUTPUT (I.E., OUTPUT TO OTHER THAN ORIGINATING TERMINAL) 3. CONTROLS USE OF UNSOLICITED OUTPUT INDICATOR
ZO#CONMT[3]	O	1. VERIFIES AUXILIARY DEVICE SPECIFICATION IN USER OUTPUT MESSAGE 2. CONTROLS CONTINUOUS OUTPUT BY HANDLING DELIVERY NOTIFICATION 3. CONTROLS OUTPUT-FOR-INPUT
ZI#TAB	R	SERVES AS TRANSLATE TABLE FROM LOWER CASE TO UPPER CASE
ZC#U4MT	O	PROCESSES DOWN-LINE LOAD REQUESTS TO A UTS400 TERMINAL.

[7] EXECUTES UNDER CONTROL OF OPERATOR COMMUNICATIONS ISLAND CODE.

9.1.4 Functional Area - BATCH PROCESSING

The BATCH PROCESSING modules control the processing of batch transactions through IMS 90.

MODULE NAME	REQUIRED VS. OPTIONAL	FUNCTIONS
ZC#ICAM[3]	R	DECODES ICAM SVC REQUESTS FROM OTHER IMC MODULES
ZC#ZZBTH[3]	O	PROCESSES ZZBTH MASTER TERMINAL COMMAND
ZC#BTCHC[3]	O	DIRECTS THE PROCESSING OF BATCH FUNCTIONS
ZC#BTHMT[3]	O	PERFORMS MULTI-THREAD DEPENDENT OPERATIONS
ZC#BPRT2	O	CONTAINS A PRINTER DTF
ZC#BPRT[3]	O	CONTAINS A PRINTER DTF
ZC#BPRT3[3]	O	CONTAINS A PRINTER DTF
ZC#BPRT4[3]	O	CONTAINS A PRINTER DTF

9.1.5 Functional Area - FILE MANAGEMENT

FILE MANAGEMENT modules control the access to all IMS 90 and user data files.

MODULE NAME	REQUIRED VS. OPTIONAL	FUNCTIONS
ZF#GEN2 [4]	R	1. INTERCEPTS ALL REQUESTS FROM ACTION PROGRAMS AND FROM IMS 90 MODULES TO ACCESS/UPDATE FILES 2. VALIDATES PARAMETER 3. SELECTS AND GIVES CONTROL TO THE APPROPRIATE MODULE TO EFFECT THE REQUEST 4. MANAGES RECORD LOCKS 5. PERFORMS FILE MANAGEMENT RELATED TERMINATION PROCEDURES AT ACTION END INCLUDING RECOVERY PROCEDURES AT ABNORMAL TERMINATION
ZF#SIMR1 [6]	R	1. PROCESSES REQUESTS FOR ACCESS TO THE NAMED RECORD FILE (I.E., GET, GETC) 2. MAINTAINS A MAIN STORAGE SUBFILE OF THE MOST RECENTLY ACCESSED RECORD
ZF#SIAMS [6]	R	PROCESSES REQUESTS FOR ACCESS TO THE CONDATA FILE (I.E., GET, PUT)
ZF#AUDIT [6]	O	PROCESSES REQUESTS TO AUDIT FILE (I.E., GET, PUT)
ZF#ISAM [6]	R	PROCESSES THE FOLLOWING RANDOM AND SEQUENTIAL REQUESTS TO ISAM USER DATA FILES 1. GET 2. GETUP 3. PUT 4. DELETE 5. INSERT 6. SETL 7. ESETL
ZF#DAMR [6]	O	PROCESSES THE FOLLOWING REQUESTS TO DAM RELATIVE ORGANIZATION (RELATIVE RECORD) USER DATA FILES

		1. GET 2. GETUP 3. PUT 4. DELETE 5. INSERT
ZF#SEQ[6]	0	PROCESSES PUT REQUESTS TO SEQUENTIAL USER DATA FILES
ZF#TRACE[6]	0	WRITES BEFORE AND AFTER IMAGES TO A JOURNAL TAPE FOR OFFLINE RECOVERY
ZF#SBPRM	0	USER SUBPROGRAM INTERFACE
ZF#TOM2[6]	0	WRITES OUTPUT MESSAGE TO TOMFILE WHENEVER A ROLLBACK POINT IS ESTABLISHED.
ZG#SNAPM[4]	0	EDITS SNAP OUTPUT
ZF#O PC2[6]	0	PROCESS OPEN/CLOSE OF FILES IN RESPONSE TO ZZOPN/ZZCLS
ZF#IRAM[6]	0	PROCESS THE FOLLOWING RANDOM AND SEQUENTIAL REQUESTS TO IRAM USER DATA FILES.

9.1.6 Functional Area - DEFINED RECORD MANAGEMENT

DEFINED RECORD MANAGEMENT controls all requests to user defined files.

MODULE NAME	REQUIRED VS. OPTIONAL	FUNCTIONS
ZJ#SCHED 4	R	REQUESTS THE RETRIEVAL OF THE APPROPRIATE DATA DEFINITION RECORD AND DETERMINES THE SIZE OF THE DEFINED RECORD AREA REQUIRED FOR THIS DEFINED FILE
ZJ#DRM7 4	O	INTERPRETS AND DIRECTS FUNCTION CALLS FROM UNIQUE AND USER ACTION PROGRAMS WHICH INVOLVE A DEFINED FILE. DEFINED RECORD MANAGEMENT SUPPORTS RANDOM RETRIEVAL, SEQUENTIAL RETRIEVAL, AND RANDOM UPDATE OF DEFINED FILES FROM ONE OR MORE LOGICAL FILES

9.1.7 Functional Area - SHUTDOWN

SHUTDOWN performs those functions necessary to terminate IMS 90.

MODULE NAME	REQUIRED vs. OPTIONAL	FUNCTIONS
ZT#SHDWN	R	1. WRITES RESTART RECORDS TO NAMEREC 2. CLOSES ALL FILES

9.2 MULTITHREAD IMS 90 GENERAL FLOW

Figures 9-1, 9-2 and 9-3 are provided to summarize the general, internal procesing flow thru several of the functional areas of multi thread IMS 90.

9.2.1 Input Message

Figure 9-1 shows the general processing for an input message from receipt by IMS 90 to the scheduling and loading of an ACTION program.

```
 ------         -------  (2)        -------------
|      |(1)| IMC    |----------|             |
|ICAM  |-->|SUBTASK|        ->|THREAD MGMT  |
 ------      -------    (3)|  |-------------|
    ↑                    ->|ACTION SCHED.|
    |                     ->|             |<-
----------                 |  |-------------| |
\TERMINAL/              (4)|  | GEN. REQ.   | |
 \      /                  |  |   PROC      | |
  ------                   |  |-------------| |(5)
                          ->|MAIN STORAGE | |
                            | MANAGEMENT  | |
                            |-------------| |
                            |   -------   | |
                            |  | ACTION|  |<-
                            |  |PROGRAM|  |
                            |   -------   |
                             -------------
```

1. ICAM notifies Internal Message Control of an input message.

2. Internal Message Control performs editing of the input message, queues the action and awakes the main task.

3. Thread Management schedules Action Scheduling to allocate resources to process the input.

4. Action Scheduling gives Main Storage Management Control to allocate space for Control Blocks, Action Program and the Activation Record.

5. Once this has been done, Action Scheduling loads the Action Program and moves the input message into the users Input Message Area (IMA).

FIGURE 9-1: GENERAL FLOW - INPUT MESSAGE

9.2.2 GET Request

Figure 9-2 is provided to show the internal processing of an ACTION program GET request for a record in a user data file.

```
                 MAIN TASK
               ---------------
              | THREAD MGMT   |
              |---------------|              I/O
              | ACTION SCHED  |            SUBTASK
              |---------------|  (3)      ----------
           -> | GEN REQ PROC  |------->|  FILE     |
           |  |               |<-------|MANAGEMENT|
      (1)  |  |---------------|  (4)   |__________|
           |  |MAIN STORAGE   |             ^
           |  | MANAGEMENT    |             | (3)
           |  |---------------|             v
           |  |    ACTION     |         ----------
           -> |    PROGRAM    |        |  DATA    |
              |_______________|        |MANAGEMENT|
```

1. User Action Program requests a record to be retrieved from a user data file.

2. General Request Processor gives control to the appropriate File Management module.

3. File Management awakes an I/O subtask which issues the request for the record from Data Management.

4. The record is returned to the Action through File Management and General Request Processor.

FIGURE 9-2: GENERAL FLOW - GET REQUEST

9.2.3 Termination Processing

The general processing flow for termination from an ACTION program is shown in Figure 9-3.

```
 ---------          -----------     (3)  ------------
|         | (4) |               |<-----|              |
|  ICAM   |<----|IMC SUBTASK    |----->|THREAD MGMT   |<-
|_________|     |_______________|(5) --|              |  |(2)
     |                           (6)|  |--------------|  |
     |                             ->|              |--
     V                             --|ACTION SCHED  |<-
                                    |  |--------------|  |
 ----------                         |  |              |  |
\TERMINAL/                       (7)|  |GEN REQ PROC  |  |
 \      /                           |  |--------------|  |(1)
  ------                           ->|MAIN STORAGE  |  |
                                     | MANAGEMENT   |  |
                                     |--------------|  |
                                     |   ACTION     |  |
                                     |  PROGRAM     |--
                                     |______________|
```

1. User Action Program terminates itself and Action Scheduling deallocates the resources for this action.

2. Action scheduling then initiates output message processing.

3. Thread Management issues a CAWAKE to the IMC task.

4. The IMC subtask takes the output message and issues an MWRITE to ICAM.

5. Once the output message is passed to ICAM, the Internal Message Control Task awakes the main task.

6. Thread Management notifies Action Scheduling to complete termination processing.

7. Action Scheduling gives control to Main Storage Management to release the areas assigned to the Activation record, Control Blocks, and Action Program.

FIGURE 9-3: GENERAL FLOW - TERMINATION PROCESSING

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(System) *(Release)* *(Level)*

(Document Title)

(Issue Number) *(Revision Number)* *(UP- Number)* *(Revision Number)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY

SPERRY UNIVAC

ATTN.: SERIES 90
SOFTWARE CONTROL

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD

CUT