SPERRY RAND

UNIVAC

1108

MULTI-PROCESSOR SYSTEM

# EXEC 8 INTERNAL ORGANIZATION VIDEO SEMINAR

# HOW TO USE THE MATERIAL

To derive maximum benefit from the EXEC 8 video tape presentations, the seminar should be conducted in the following manner:

1. The supporting documentation for the particular presentation should be read by each participant.

2. The particular video tape presentation should then be shown (no more than two 1-hour presentations should be shown in one day).

3. Following each presentation, a seminar-type discussion group should then be formed to talk over all the points raised in the presentation.

4. After all video tapes have been shown, the discussion group should tie together all the major components of the EXEC 8 Operating System covered in the various presentations.

UNIVAC is a registered trademark of Sperry Rand Corporation.

© 1969 - SPERRY RAND CORPORATION

PRINTED IN U.S.A.

# CONTENTS

# LIST OF ILLUSTRATIONS

# 1. INTRODUCTION AND SUPPORTING ELEMENTS

Introduction

This Seminar is designed to give the viewer a working knowledge of EXEC 8. After completing this seminar, the viewer should be capable of resolving most of his problems from the materials supplied with EXEC 8. UNIVAC recommends this seminar as a prerequisite for further study and understanding of the 1108 Operating System.

Recommendations

It is recommended that the viewer have large scale system experience, familiarity with the 1108, and a working knowledge of the 1108 Programmers Reference Manual (UP-4144).

Contents

The contents of the seminars by subject are as follows:

1. Introduction and Supporting Elements (this section).

2. Symbionts

   Identify the input and output of EXEC 8 and its utilization by USER/EXEC.

3. Coarse Scheduler

   Definition of the scheduling and activation of runs in the executive, including facility assignments. Discussion of DEMAND and BATCH Runs.

4. Dynamic Allocator

   The control of central memory and swap file space. The placing of a program into initial execution.

5. Dispatcher

   Execution control and the method of granting time slices. A discussion of the FORK and ACT/DEACT mechanism. A discussion of User's TABLE/QUEUE relation.

6. Termination

   The methods utilized in terminating an activity (normal and abnormal), a program and a run. A discussion of the accounting functions.

7. Run Stream Analysis

   A summary of the entire seminar as related to a run stream.

The depth of the seminar is at a design definition level and is at or near the current level in operation. UNIVAC will maintain the accuracy of the seminar and notification will be given of updates.

Definition of Terms

1. RUN.
   A set of user requests for executive services with its beginning marked with a @RUN and its end by a @FIN.

2. TASK.
   The individual service request, within a run, preceded by a master space.

3. RUN STREAM.
   The collective set of tasks and the data images supporting each of the tasks.

4. PROGRAM.
   An absolute element, loaded and executed in response to an executive task.

5. ACTIVITY.
   All or part of a program given an independent time slice under the control of a unique switch list entry.

6. SWITCH LIST ENTRY.
   The key to activity control. The handle by which the executive can control each activity currently in the execution mix.

SUPPORTING ELEMENTS

System Initialization

In order to start up the system and condition it to operate, a group of routines must be executed. This process is called System Initialization.

This may take place in two ways, from tape or from drum. The usual method is from drum, however the tape method involves extra steps and will be covered first.

The systems tape used contains the initialization routines and three major files:

1. EXEC 8 resident and non-resident
2. Processors
3. Libraries

All three files are read from tape and placed on drum for use by the system when active, and the resident portion of the EXEC is placed into memory and initialized.

In an initialization from drum the three major files already exist on drum, so the only action is that of placing the resident EXEC into memory and conditioning it for operation.

During initialization process, the operator is able to specify configuration changes which exist at the time. The major portion of the conditioning of the EXEC is composed of setting up tables, configuring the symbiont complex and communications environment, and activating certain elements of the system so that it may accept input.

Function Loader

A non-resident executive routine that occupies user memory when loaded, is called a function. Within the system there is an element called the Function Loader that has three entrances as follows:

1. FUNCLR: Load the function, activate it and return to the requestor.
2. FUNCLE: Load the function, activate it and return to the Dispatcher.
3. FUNCE: Used by the function to terminate itself. Upon termination, control is returned to the Dispatcher.

EXPOOL

EXPOOL is the EXEC pool of buffers available for temporary use. These buffers are available for EXEC elements to build tables/queues such as the Switch List for execution of an activity.

The elements of the EXEC Pool Routine are:

1. EXPOOL - Buffer Acquisition
2. EXREL - Buffer Release

The only table required is BUFTAB/BUFA which contains pointers to the buffers which are indexed by size (figure1).

A request for an EXPOOL buffer included:

1. Buffer size in words
2. Immediate request
3. Buffer chain request
4. Chaining to head/tail
5. Address of chain control (figure2)

The buffer (figure3) given to the requestor contains as the first words a Buffer Control Word (BCW).

The functions of the buffer request element EXPOOL are:

1. Check for available buffer
2. Break down larger buffer if necessary
3. Build the Buffer Control Word (BCW)
4. Update EXPOOL size (EXPSIZ)
5. Check for EXPOOL tight (EXTITE)
6. Exit to requestor

When the buffer is to be released, the requestor will supply to the element EXREL the address of the buffer.

The functions of EXREL are:

1. REMOVE the assigned flag
2. Attempt buffer link up
3. Release full core blocks
4. Exit to requestor

The EXEC pool routine is capable of requesting more core dynamically as required from another portion of the EXEC. The request is made to the Dynamic Allocator which handles all core requests.

| | | |
|---|---|---|
| T/S | 04 | Next available 4 wd Buffer |
| T/S | 010 | Next available 8 wd Buffer |
| T/S | 020 | Next available 16 wd Buffer |
| T/S | 040 | Next available 32 wd Buffer |
| T/S | 0100 | Next available 64 wd Buffer |
| T/S | 0200 | Next available 128 wd Buffer |
| T/S | 0400 | Next available 256 wd Buffer |
| T/S | 01000 | Next available 512 wd Buffer |

*Indexed by buffer size (Power-of-2)

Figure 1. BUFA/BUFTAB

| | |
|---|---|
| 1ST buffer in chain | Last buffer in chain |

Figure 2. Chain Control Word (CCW)

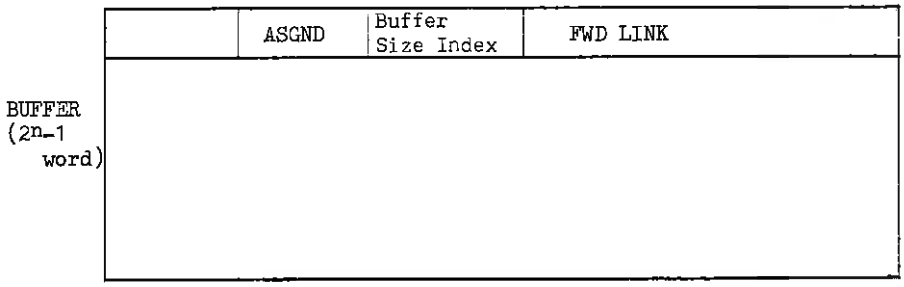



Figure 3. Buffer layout showing Buffer Control Word (BCW)

# 2. SYMBIONTS

Definition

The symbiont complex provides the interface between on-site and remote unit record equipment and the user.

Some of the services performed by the symbiont complex are:

1. Symbionts allow disassociation of the user (and the EXEC) from the relatively slow unit record device speeds, allowing tasks to proceed at higher internal and mass storage speeds.
2. In a multiprogramming and multiprocessing environment, the symbiont complex provides two important abilities:
   a. The ability to provide for multiple asynchronous input and output
   b. The ability to preread control stream input for pre-scheduling and facility allocation.

The symbiont complex is generally reentrant allowing multiple I/O on identical devices to be handled by a single set of routines.

Symbiont Relationships

In general, the symbionts interface with the user via READ$, PRINT$ and PUNCH$. System input, however, is controlled by a timed probe routine, and the input and output symbionts normally interface with other EXEC functions rather than the user.

The symbionts interface with EXEC functions such as:

1. The Control Stream Interpreter, for control card interpretation and syntax check.
2. The Coarse Scheduler, to allow prescheduling of runs at input time and for run time assignments and tasks loads.
3. Facilities Inventory Management, for acquisition of file space for input and output files.

General Symbiont Operational Philosophy

The operation of the symbiont complex is divided into three general phases:

1. Control Stream Input (ASYNCHRONOUS)
2. Run Time (SYNCHRONOUS)
3. End of Run (ASYNCHRONOUS)

Certain tables are essential to symbiont operations:

1. Probe Tables
2. SMLIST Table
3. SMACT Table
4. Control Tables
5. Data Tables
6. SMOUTQ

The Functions of the Three Phases of the Symbiont Complex

1. Control Stream Input:
   a. Scan, via SMPRB9, probe tables (Figure 1 & 2) for devices with input for the system.
   b. Activate SYMICR and the device symbiont.
   c. Enter activated symbionts in SMACT table (Figure 3) from information in SMLIST (Figure 9).
   d. Initialize the control table for the device (Figure 4).
   e. For a device which has input, input is initiated and controlled by SYMICR, which:
      (1) Interprets control statements
      (2) Checks control statements for proper syntax via the Control Statement Interpreter
      (3) Passes control to the Coarse Scheduler for valid RUN statements
      (4) Creates a mass storage file for each run stream
   f. Control of SYMICR and the device symbionts is maintained via the flag BITCTL (Figure 5) contained in the FCT of the control table for the device (Figure 4).

2. Run Time:
   a. READ$
      (1) Used by the Coarse Scheduler to obtain control statements from the run stream on mass storage.
      (2) Used by the user to obtain data from the run stream on mass storage.
      (3) READ$ activity is controlled via the RD$DT (Figure 6).

   b. PRINT$/PUNCH$
      (1) Used by the EXEC to place images such as accounting in the print file for the run.
      (2) Used by the user to place output data in the print file or punch file during execution.
      (3) Activity controlled via PR$DT (Figure 8) or PN$DT as appropriate
      (4) Data for the output files is placed on mass storage.

3. End of Run:

   a. When the Coarse Scheduler detects a FIN control statement, accounting information is placed in the PRINT$ file and control is passed to the end of run phase of the symbiont complex which will:

      (1) Close PRINT$/PUNCH$ files.
      (2) Queue output files in the proper SMOUTQ by device class.
      (3) Release the READ$ file associated with the run.

   b. As output devices are available, the physical output of files in the SMOUTQ is accomplished. When a file has been completed:

      (1) The file is removed from SMOUTQ.
      (2) The output file count for the run is decremented.
      (3) If the output file count is decremented to zero, REMID removes the RUNID.
      (4) The appropriate file and any remaining space is returned to the EXEC for other use.

Table Control Word

| | | | |
|---|---|---|---|
| | TNU9A | NAU9 | TNU9 |

Table Entries 1 Per Unit

| | | |
|---|---|---|
| Symbiont Name | | |
| | INUSE9 | DEVCLS |
| RSWL9 | | PSWL9 |
| Punch Symbiont Name On Same Channel | | |

TNU9A : NAU9 reset value
NAU9 : Index to current entry
TNU9 : Total number of entries

INUSE9 : Device in use flag
DEVCLS : Device class
RSWL9 : READ symbiont switch list address
PSWL9 : PUNCH symbiont switch list address for punch on same channel

Figure 1. 900 CPM Reader Control Table

| | | | | | |
|---|---|---|---|---|---|
| | | | | NAU | TNU |
| UNITID | | | | | |
| CHAN | SBCHAN | RMODE | ASGND | DEVCLS | |

NAU : Current entry value
TNU : Total number of entries
RMODE : Standard read mode flag
ASGND : Assignment flag
DEVCLS : Device class

Figure 2. On-site 1004 Probe Table (Two Word Entries)

<table>
<tr><td colspan="3"></td><td>MAXNO</td><td>NOENT</td></tr>
<tr><td colspan="3">Backward Link</td><td colspan="2">Forward Link</td></tr>
<tr><td colspan="5">SNAME or SITEID</td></tr>
<tr><td>EQPTYP</td><td>ACTBIT</td><td>FRKIND</td><td colspan="2">Control Table Address</td></tr>
</table>

Where: MAXNO - Maximum number of entries that can be held in this table buffer.

NOENT - Number of entries in this table buffer

S1: EQPTYP-Equipment Type
S2: ACTBIT

01 - KILFIL
010 - LOKBIT
020 - SUSBIT

S3: FRKIND-Indicates Symbiont To Fork

0 - None
1 - Print
2 - Punch

Figure 3. SYMACT Table (Two Word Entries)

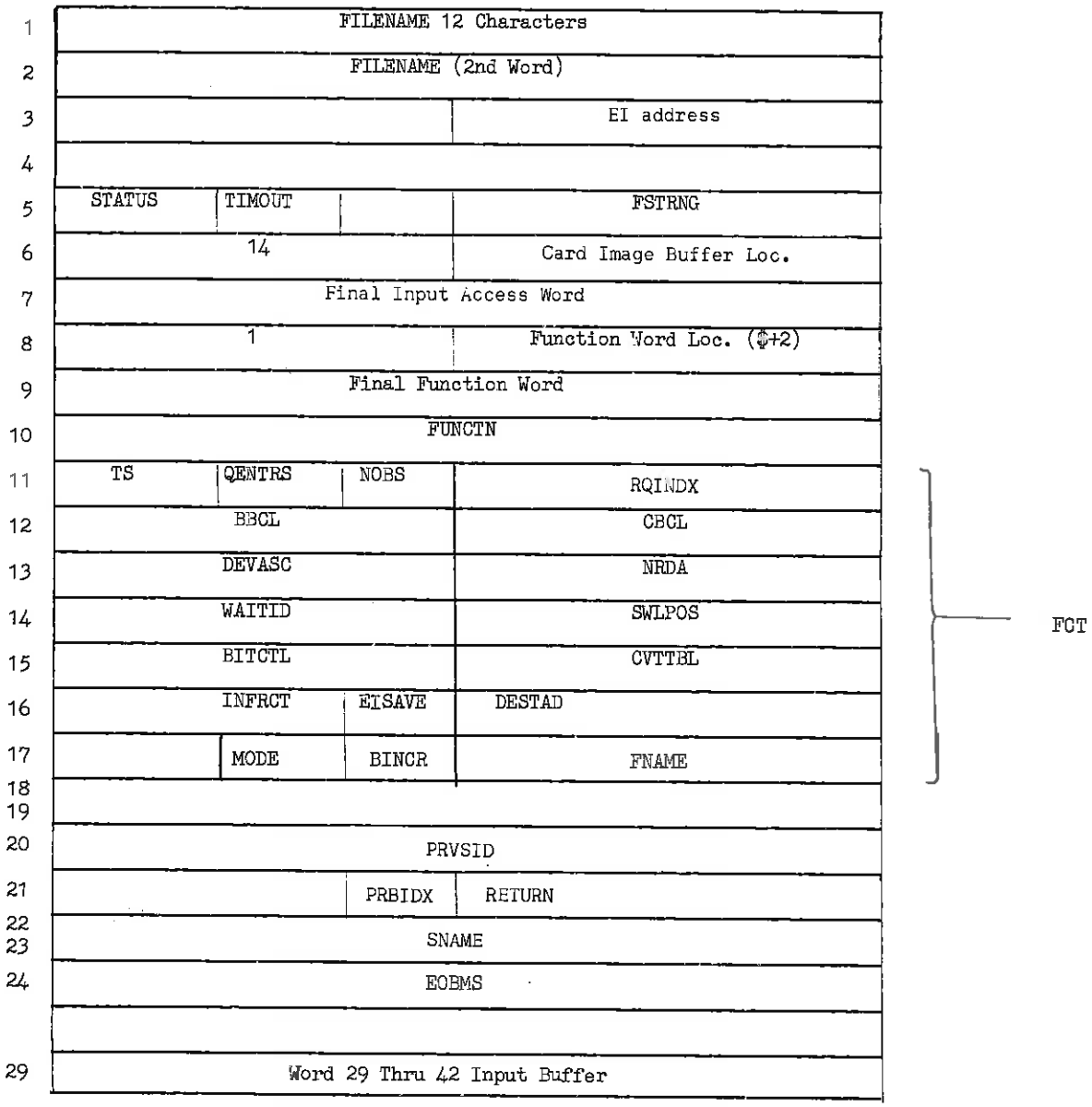


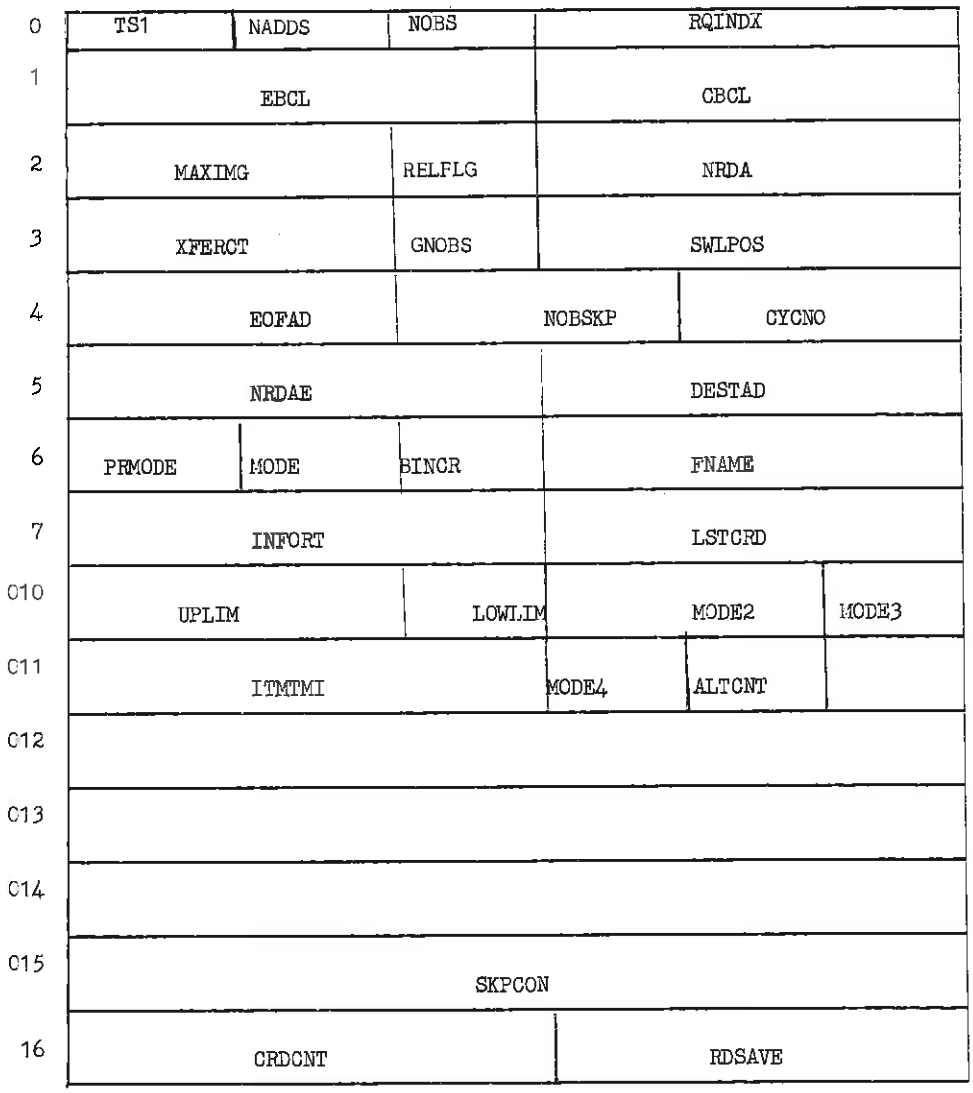

Figure 4. (1 of 2) 900 CMP Reader Control Table (56 Word EXPOOL Buffer)

| | | |
|---|---|---|
| TS | : | Test & Set inside of symbiont complex |
| QENTRS | : | Number of units in queue |
| NOBS | : | Number of buffer being filled |
| RQINDX | : | Address of RD$DT |
| BBCL | : | Location of beginning buffer in chain |
| CBCL | : | Location of current buffer in chain |
| NRDA | : | Relative drum address, current track |
| BITCTL | : | Condition flag (Figure5) |
| CVTTBL | : | Location of conversion table |
| INFRCT | : | Number of words in image |
| DESTAD | : | Destination address for image |
| MODE | : | Indicator Flag |
| BINCR | : | Increment to next unfilled word in buffer |
| FNAME | : | Name of mass storage file |
| PRVSID | : | Previous ID |
| PRBIDX | : | Probe index |
| RETURN | : | SYMICR return point to device symbiont |
| SNAME | : | Symbiont name |
| EOBMS | : | End of binary mode sentinel |
| EI Address | : | Address to which control is passed for processing an external interrupt for this device |
| STATUS | : | Status code returned on completion of function on this device |
| TIMOUT | : | Timeout location in ADH |
| FSTRNG | : | Function string for ADH |

Figure 4. (2 of 2) 900 CPM Reader Control Table

| OCTAL VALUE | EXPLANATION |
|---|---|
| 4000 | Continuation of run image |
| 2000 | Device interlock |
| 1000 | Run search mode |
| 400 | Initial run mode |
| 200 | File mode |
| 100 | Data mode or ELT,D mode |
| 40 | Multiple message flag |
| 20 | Alternate files |
| 10 | Binary mode |

Figure 5. BITCTL




Where TS1 : Test and set inside symbiont complex
NADDS : Number of open adds in this file
NOBS : Number of the buffer being filled
RQINDX : Contains address of symbionts data table

EBCL : End of current buffer chain location
CBCL : Location of current buffer in chain

MAXIMG : Max. image length output symbiont can process
RELFLG : The previous track has been release if set
NRDA : Next relative drum address for this file

XFERCT : Number of words to transfer
GNOBS : When NOBS less than GNOBS, get new 224 word block
SWLPOS : Switchlist position for PRINT$ Activity; set when PRINT$ waits for write routine

EOFAD : Address of user's EOF, or abnormal, return
NOBSKP : On read of new block, release buffers until NOBS = NOBSKP

Figure 6. (1 of 2) READ$ Data Table

```
CYCNO  : Specified cycle number for start of add files

NRDAE  : Beg. file addr. of start of add element
DESTAD : Destination address for image

PRMODE : More mode
MODE   : 040 - Set if file is on tape
         020 - Set if end return request on add
         010 - Set if table is closed
          04 - Set if in binary mode
          02 - Set if in data mode
          01 - Convert address to absolute (Data transfer from user area)
BINCR  : Increment to next word in buffer
FNAME  : Name of mass storage file

INFORT : Address of INFOR table
LSTCRD : Address of 28 word buffer with the last card in it

UPLIM  : Upper limit of allowable @CARDS
LOWLIM : Lower limit of allowable @CARDS
MODE2  : 040 - Set on first abnormal return (on second, give error exit)
         020 - Indicates user hasn't whole INFOR
         010 - Set when CSI requests continue cards
          04 - Set by CS before it goes to reads
          02 - Indicates no printing. don't release file as it is read.  Skip
               any cards of type which can't be returned.
          01 - Set during DEMAND runs
MODE3  : 040 - When set, allows next read to get any type (used by CS when
               processor is being called)
         020 - RDSAVE points to list of CLIST commands
          02 - Program can receive @CARDS
          01 - Program can receive data cards

ITMTMI : PCT item address relative to AA in PCT
         for current @ADD File
MODE4  : 040 - Reschedule run on recovery
         020 - Run was scheduled by @START
         010 - Set while processing @DATA and @ELT,D
          04 - Do not clear BCFSYM in ADD close
          02 - This is a dynamic ADD file
          01 - Free only use name in add close
ALTCNT : Count of active alternate files

SKPCON : Card count or label for @JUMP etc.

CRDCNT : Number of cards read by READ$

RDSAVE : READ$ data table save area during facility synopsis or pointer for CLIST
```

Figure 6. (2 of 2) READ$ Data Table

| RUNID | | | |
|---|---|---|---|
| QUALIFIER | | | |
| | | | |
| File Name | | | |
| File Name | | | |
| | UPBITS | PARTNO | UPDVAS |
| UPCHNB | | | UPCHNF |

UPBITS :

020 - Set for tape file
010 - Set for PUNCH file
04 - Set for user defined files
02 - Set when UPDVAS points to an on-site input entry in SMLIST
01 - Set if UPDVAS points to a particular output device (site)

PARTNO : Checkpoint number

UPDVAS : Device class

UPCHNB : Chain, backward link

UPCHNF : Chain, forward link

Figure 7. SMOUTQ

```
FILEQD : Pointer to queueing information after a @SYM

INFRCT : Word count of print image
PARTNO : Part (breakpoint) number for this file
SORCAD : User's buffer address

PRMODE : 040 - Lock on other calls to this table
         020 - Data transfer uses absolute addressing
         010 - File is defined in user's PCT
          04 - Reserved for input symbiont use
          02 - Same as 04
          01 - Same as 04

MODE   : 040 - Set if file is on tape
         020 - Set for kill on max pages
         010 - Set for kill on max cards
          04 - Set for DEMAND runs
          02 - Lock on data buffer chain and NOBS
          01 - Set for DEMAND runs
BINCR  : Increment to next available word in CBCL
FNAME  : Contraction of file name for tape or drum

ESTPCT : Estimated page count for run
ESTCC  : Estimated card count for run
CPGCT  : Current page count for run
CRDCT  : Current card count for run

ADDTBL : Address of saved information for @ADD or Ø
RD$RTN : Return address for non-ER call on PRINT$

FILQPU : FILEQD for PUNCH$ if no control table exists
FNMTBL : Address of information for Alt. files
```

Figure 8. PRINT$ Data Table
(1 of 2)

| | | | |
|---|---|---|---|
| TS1 | QENTRS | NOBS | RELSOR |
| BBCL | | | CBCL |
| DEVASC | | | NRDA |
| WAITID | | | SWLPOS |
| PUPRTN | | TABTYP | FILEQD |
| INFRCT | | PARTNO | SORCAD |
| PRMODE | MODE | BINCR | FNAME |
| | | | |
| ESTPCT | | | ESTCC |
| CPGCT | | | CRDCT |
| ADDTBL | | | RD$RTN |
| FILQPU | | | FNMTBL |

Where TS1 : Test and set inside symbiont complex
QENTRS : No. of blocks queued for writing
NOBS : No. of data buffers in use
RELSOR : Relative addr. of user's print buffer

BBCL : Beginning of buffer chain location
CBCL : Current buffer chain location

DEVASC : Device association index for runs output
NRDA : Next relative drum address for this file

WAITID : Address of symbiont switch list of LT table when waiting for buffers to take
SWLPOS : Switch list address when waiting for drum or tape write

PUPRTN : Part no. for PUNCH$ when no control table exists for it
TABTYP : Table type values are:

01 - PRINTS$
02 - PRNTA$
04 - PUNCH$
010 - PNCHA$
020 - READA$
040 - ADD

Figure 8. PRINT$ Data File
(2 of 2)

SMLIST : on-site Output

| UPSNAME | | | |
|---|---|---|---|
| | LSTBIT | | |
| CHAN | SUB-CH | UNIT | UPOSCH |

UPSNAME : Symbiont name
LSTBIT : Control flag

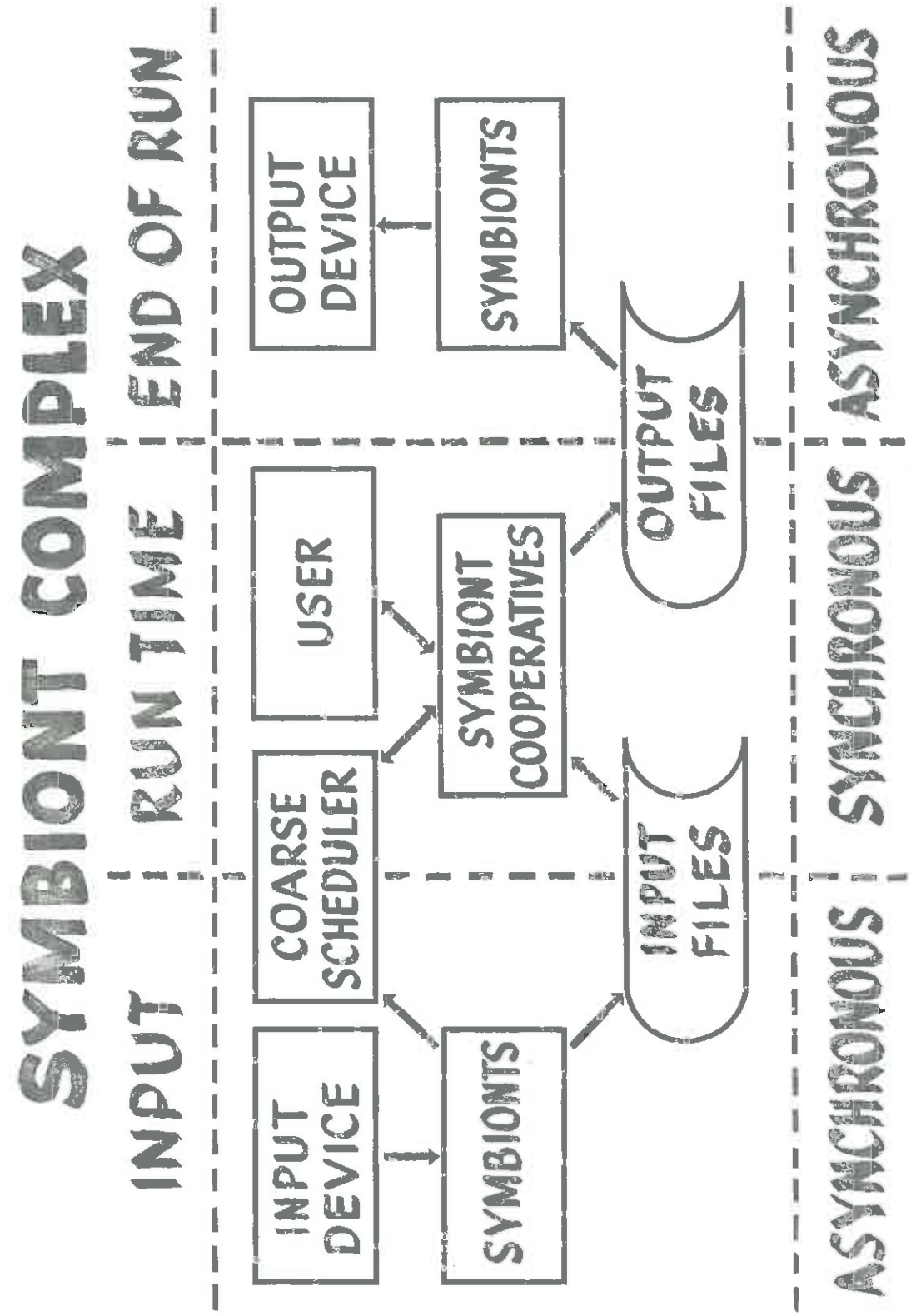
02 - activate SYM when device is available
01 - device not available

SMLIST : On-site Input

| SYMB NAME | |
|---|---|
| OSPRA1 | OSPUA1 |
| OSPRAS | OSPUAS |

OSPRA1 : Pointer from input group to associated printer group
OSPRAS : Pointer from input device to associated printer
OSPUA1 : Pointer from input group to associated punch group
OSPUAS : Pointer from input device to associated punch

Figure 9. SMLIST


SYMBIONT COMPLEX
INPUT
RUN TIME
END OF RUN
INPUT DEVICE
COARSE SCHEDULER
USER
OUTPUT DEVICE
SYMBIONTS
SYMBIONT COOPERATIVES
OUTPUT FILES
INPUT FILES
SYMBIONTS
ASYNCHRONOUS
SYNCHRONOUS
ASYNCHRONOUS

# 3. COARSE SCHEDULER

The Coarse Scheduler is responsible for the scheduling, selection, activation and analysis of all run streams entered into the operating environment of the 1108 system.

The types of runs capable of being submitted are:

1. Demand - Immediate activation

2. Batch - Scheduled and then selected on a priority basis with deadline time influencing priority revision, and start time allowing the priority to be considered.

The elements of the Coarse Scheduler are as follows:

1. Main

   a. CSP - Initializer of tables, and the prescheduler of BATCH runs.

   b. CSU - Updater of the scheduling queue.

   c. CSN - Selector of the highest priority run from the scheduling queue.

   d. CSA - Answerer, activator and analyzer of all runs.

2. Supporting

   a. CSI - Control statement interpreter.

   b. CSTART - Starter of a catalogued run stream.

   c. CSF - Analyzer of dynamic control stream requests made by the user during the execution of the absolute element.

   d. CSK - Interface between operator's unsolicited keyins and the Coarse Scheduler.

Because these elements operate independently of each other, and since each element has the ability of activating another element, there is a necessity of having tables, entries and queues in order to maintain information.

These are:

1. RUNIDQ Entry/Queue
2. RUNQ Entry/Queue
3. Scheduling Entry/Queue
4. Facility Request Pointers
   a. FSTAB - Initial Batch
   b. PFTAB - Between Batch
   c. DFTAB - Demand
5. CQE/CBQ - Core Queue Entry/Core Buffer Queue
6. RCT - Run Control Table
7. PCT - Program Control Table
8. LOCTAB - Location Table of System Information (Figure 12)
   (Includes linkages to Facility Request Pointers, and portions of the Scheduling Queue; also, EXEC 8 system standards for maximum time, maximum pages, and maximum cards.)
9. INFOR - Internal Format of Control Statements

Coarse Scheduler - Batch

The Coarse Scheduler's responsibilities for Batch runs are to:

1. Initialize Tables
2. Schedule Runs
3. Update Scheduling Queue
4. Select Runs
5. Activate Runs
6. Analyze Control Statements

Control is passed to the Coarse Scheduler (CSP) when SYMICR (Symbiont Input Control Routine) acknowledges the start of a runstream, i.e., encounters a run image.

CSP (Batch)

CSP's functions are as follows:

1. Build/Queue RUNIDQ to maintain uniqueness of run ID's.
2. Build/Queue RUNQ to maintain run image information.
3. Build/Queue "BASIC" scheduling entry to schedule run on a priority, deadline or start time basis.
4. Activate SYMICR to read in the remainder of the run stream.
5. Activate CSN to select the highest priority run.
6. Exit

The entries which CSP builds are:

1. RUNIDQ - Figure 1
2. RUNQ - Figure 2
3. 'BASIC' Scheduling Entry - Figure 3

CSU

The updating of the Scheduling Queue is performed by the element CSU.

The Scheduling Queue consists of four portions:

*1. PRIORQ (Fig.4) - Linked on a PRIORITY basis. This is the portion from which all entries are selected.

*2. STARTQ (Fig.5) - Linked on START TIME basis.

*3. DEADQ (Fig. 6) - Linked on DEADLINE TIME basis.

4. SCHQ (Fig.3) - Overflow from the other 3 portions. Linked on PRIORITY, START TIME, OR DEADLINE TIME.

PRIORITIES/DEADLINE TIMES are not considered for START entries until the START TIME has elapsed.

The philosophy behind the update function of CSU is to update all entries with either DEADLINE or START TIMES and then rechain those entries whose priorities may have been revised.

* - Fixed in length.

Time Update

** PRIORQ - Any DEADLINE entries

** DEADQ - All entries

STARTQ - All START TIMES of the entries. If this time has completely elapsed, then the entry assumes its PRIORITY/DEADLINE TIME status.

** SCHQ - Any entries with a DEADLINE TIME or START TIME.

** - As DEADLINE TIME passes into the critical zone, the priority of the entry is revised to indicate the critical nature of the entry.

Priority Rechain

PRIORQ - Rechain entries on a priority basis.

DEADQ

1. Attempt entry into PRIORQ ("BUMP" if necessary)
2. Fill PRIORQ

STARTQ - Entries which have assumed PRIORITY/DEADLINE TIME will be removed (see DEADQ #1 & #2)

SCHQ - Rechain as in PRIORQ, DEADQ and STARTQ. A "Bumped" entry. is moved down to its appropriate position in the Scheduling Queue.

CSN

CSN (the selector) performs the following functions:

1. Select the highest priority scheduling entry.
2. Build the CQE/CBQ (see the Dynamic Allocator).
3. Build the PCT (Program Control Table - Figure 7).
4. Build RD$DT, PR$DT tables (see Symbionts).
5. Open READ$/PRINT$ files.
6. Make initial run entry into the log.
7. Link facility request images;
   a. Canned - LIB$, TPF$, DIAG$.
   b. Facility Synopsis - up to first non-facility image, ignoring "transparent" images - @HDG, @MSG, @LOG.
8. Build Facility Request Pointer (FSTAB).
9. Activate FIMAIN (Facilities Inventory Routine) - See Figure 8A.
10. Activate CSU if no entires exist in the PRIORQ portion, but in other portions of the Scheduling Queue.
11. Exit.

The primary tables that CSN builds are:

1. PCT (Program Control Table) - see Dispatcher.
2. FSTAB - Figure 8B.

CSA (Batch)

The initial analysis phase of CSA is concerned with facility requirements and tasks. For facilities, it performs the following functions:

1. Rereads facility images causing these images to be placed in the PRINT$ file.
2. Places error messages, such as FAC WARNING/REJECTION from FIMAIN following the appropriate facility image into the PRINT$ file.
3. If a fatal error exists, informs user that remaining control statements are ignored and calls for RUN TERMINATION.

For task statements, CSA performs the following:

1. @HDG, @LOG, @MSG - passes control to appropriate routine within CSA to control the processing of these tasks.
2. @XQT, @FOR, @COB, etc. - requires an absolute element to be loaded and executed.

*a. Program File Search.
   1. LIBT - table within CSA of more commonly used LIB$ elements - Figure 9
   2. LIB$ - processors
   3. TPF$ - user's temporary file
   4. User specified file

b. Acquire I & D Bank memory requirements and Header Table Address specifying where in the file the absolute element resides.

c. Place FILE name into RCT - Figure 10.

d. Request PCT Swapout by Dynamic Allocator since I Bank and PCT must be contiguous in memory.

e. Request Initial Load by Dynamic Allocator of the absolute element into memory.

*If the absolute element can not be found, inform the user and call for RUN TERMINATION.

The between task analysis calls for RUN TERMINATION if the previous task terminated abnormally. For facilities, it:

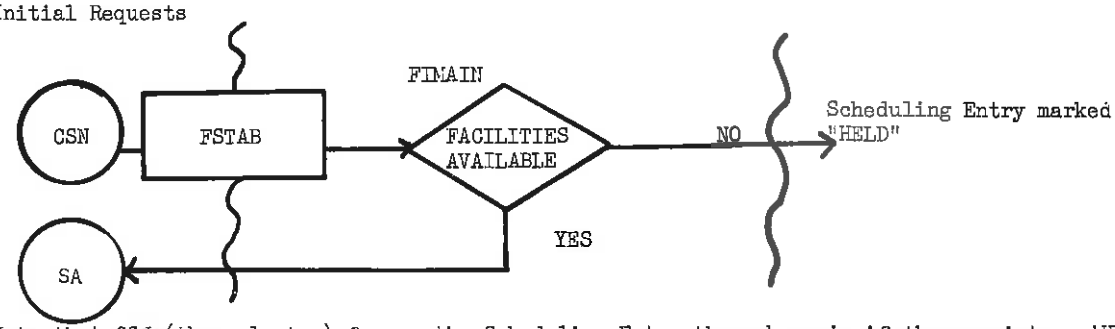
1) Performs Facility Synopsis
2) Builds Facility Request Pointer (PFTAB) - Figure 8B
3) Activates FIMAIN - Figure 8A
4) Exits

For tasks, analysis is the same as for the initial task analysis.

The element CSA will continue to process control statements until encountering a @FIN which will cause CSA to dequeue the RCT and call for RUN TERMINATION.

FIMAIN - Facility Request Routine (Batch)

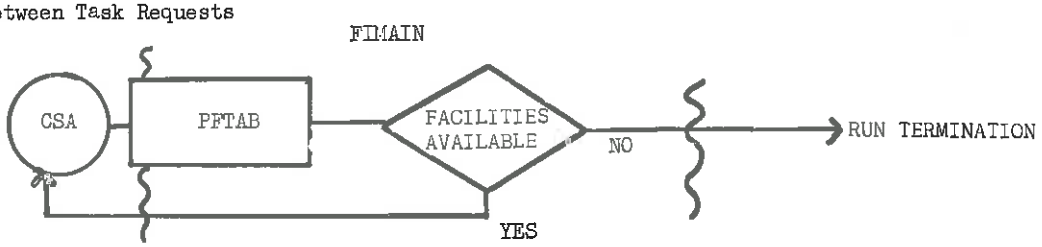
Initial Requests




Note that CSN (the selector) forces the Scheduling Entry through again if there exists a 'HELD' and there are no other entries in the PRIORQ portion of the Scheduling Queue with a priority the same as the 'HELD' Entry.

When all facilities requested are assigned to the run, the element CSA (Coarse Scheduler Answerer, Activator, Analyzer) is activated by FIMAIN to process control statements.

Between Task Requests




Coarse Scheduler (Demand)

The Coarse Scheduler:

1. Initializes tables.
2. Immediately activates the run.
3. Analyzes the control stream.

Upon detecting a run image from a demand remote device, SYMICR (Symbiont Input Control Routine) passes the image and control to the element CSP.

The functions of CSP are:

1. Build/Queue RUNIDQ Entry.
2. Build RUNQ Entry - do not queue.
3. Build CQE/CBQ - (see Dynamic Allocator).
4. Build PCT (Program Control Table) - Figure 7.
5. Make initial run entry in log.
6. Activate CSA for analysis of control stream.
7. Activate SYMICR for retrieval of the remaining run stream.
8. Exit.

As can be noted, CSP in the DEMAND mode performs many of the functions of CSN in the BATCH mode and eliminates the scheduling and updating of scheduling entries - thus emphasizing the immediate nature of DEMAND runs.

CSA (Demand)

The immediate activation of a DEMAND run requires the element CSA for processing of Control Statements.

Facility Analysis functions are to:

1. Build a Facility Request Pointer (DFTAB - Figure 8B) for each facility requested.
2. Initially include "canned" files - LIB$, TPF$, DIAG$.
3. Notify user by FAC WARNING/REJECTION message if facility error exists - no termination of the run in case of fatal error.
4. Perform a timed wait and resubmit the request if the facility is not presently available.

Task Analysis functions are performed as is defined for batch type runs. However, if the absolute element can not be found, CSA informs the user, but does not call RUN TERMINATION.

Therefore, the DEMAND user's run is not terminated for facility errors or improper requests for absolute elements; nor is it terminated if a previous task terminated abnormally.

Supporting Elements

CSI

(Control Statement Interpreter) is activated by the READ COOPERATIVE upon the sensing of a master space as the first character of the image; is responsible for formatting control statements into an internal format, INFOR (Figure 11), for easier manipulation of control image data by EXEC elements.

The main elements within the Coarse Scheduler which use INFOR are:

1. CSP - Run Image
2. CSN - Facility Images
3. CSA - All Control Statements

CSTART

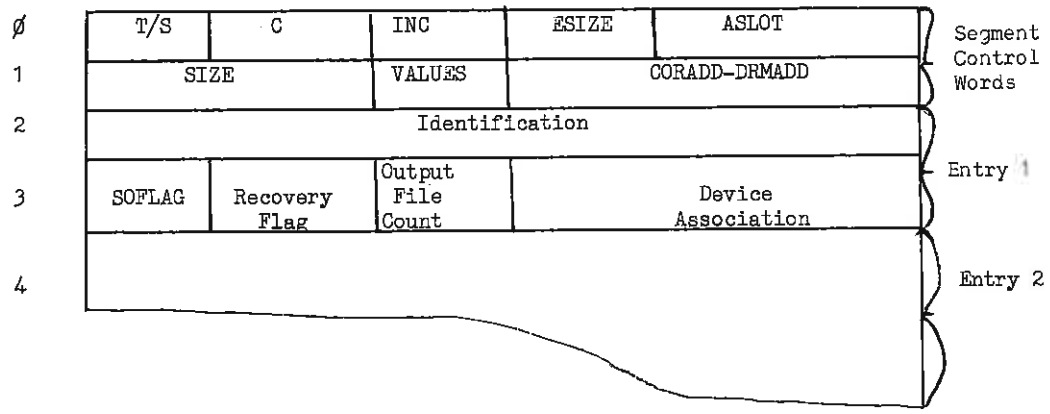
CSTART is activated by CSA when the control statement @START is encountered within the run stream implying the user's wish to start a catalogued run stream. CSTART is responsible for passing the run control image to CSP for scheduling as a BATCH Run. It should be noted that the READ COOPERATIVE will read the START specified file and that there is no necessity to activate SYMICR as the entire run stream is already on mass storage. CSTART may also be activated by the element CSF to honor a dynamic START request by the user during the execution of an absolute element.

CSF

CSF acknowledges dynamic control statement requests made by the user during the execution of the user's absolute element.

CSK

CSK allows unsolicited keyins by the operator to inform the Coarse Scheduler to mark scheduling entries as being held from selection, and if an operator can specify a hold, he also must be capable of notifying the Coarse Scheduler to release and allow these runs so that they may be processed. CSK also enables active run or backlog information to be passed to the operator.




Word Ø: S1 - T/S
S2 - C = Number of entries in next segment
S3 - INC = Number of routines using next segment
S4 - ESIZE = Entry size (number of words)
T3 - ASLOT = Index to first available slot in next segment
ØØ77 = no slots

Word 1: T1 - SIZE = Length of next segment (number of words)
S3 - VALUES = Ø = Non-critical request to EXPOOL; ØØ4Ø = Critical
H2 - CORADD = Abs. EXPOOL core address where next segment resides
H2 - DRMADD = Relative address (in SCHφ file) where next segment resides

Word 2: Identification = Unique run identification

Word 3: S1 - SOFLAG = "S" option flag
1 = Run in process
2 = Run terminated but output files exist
4 = Next run is being held for termination of this run

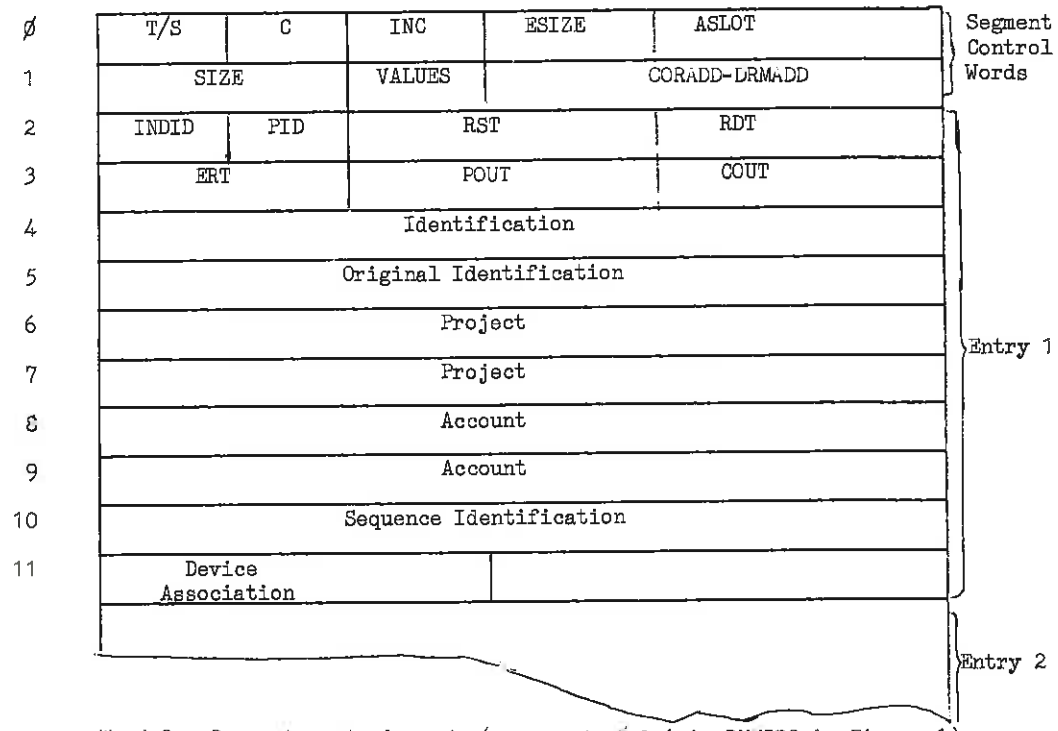
S2 - Recovery flag = Run status indicator

1 = Run in scheduling queue
2 = Run is open
4 = Run is closed
5 = Run in scheduling queue (START entry)

S3 - Output file count = Files yet to be serviced

H2 - Device association = Identification of input device

Figure 1. RUNIDQ




Word 0: Segment control words (see words Ø & 1 in RUNIDQ in Figure 1)
& 1:

Word 2: S1 - INDID = Run specifications

BIT 31 = Indicates 'C' option
BIT 32 = Indicates 'P' option
BIT 33 = Indicates 'T' option

S2 - PID = Alphabetic priority value (@RUN statement)
T2 - RST = Start time (2400 value in minutes)
T3 - RDT = Deadline time (2400 value in minutes)

Word 3: T1 - ERT = Estimated running time
T2 - POUT = Page output estimated
T3 - COUT = Card output estimated

NOTE: T1, T2, T3 are @RUN statement values if specified; otherwise account or system standards.

Word 4: Identification - Unique run identification

Word 5: Original Identification - Run identification as specified on @RUN statement

Figure 2. (1 of 2) RUNQ

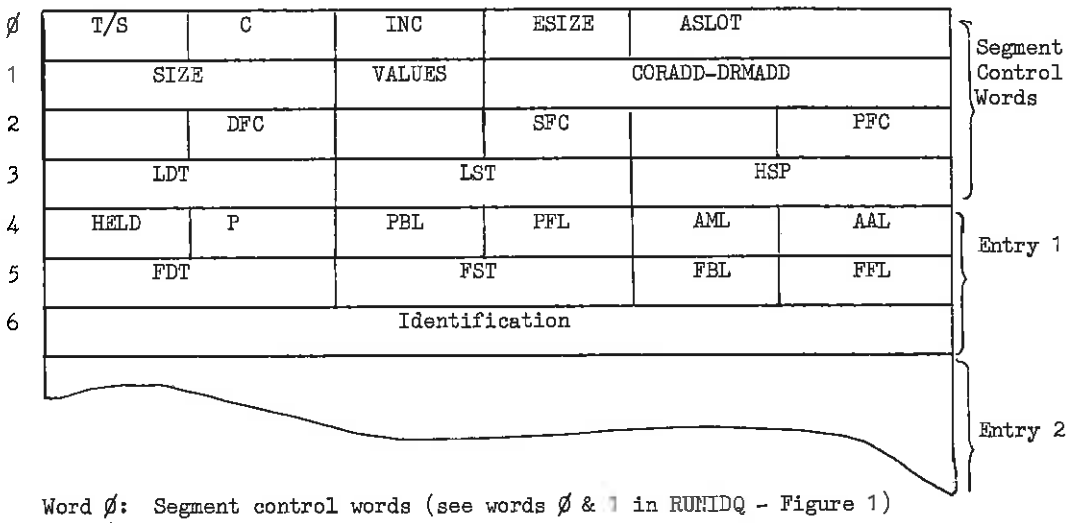
Word 6: & 7: Project - Project identification from @RUN statement

Word 8: & 9: Account - Account identification from @RUN statement

Word 10: Sequence Identification - Run identification of previous run from same input device within same probe activation

Word 11: Device Association - Identification of input device

Figure 2. (2 of 2) RUNQ




Word Ø: Segment control words (see words Ø & 1 in RUNIDQ - Figure 1)
& 1:

Word 2: Segment Control Word

S2 - DFC = Index to the most critical of the 'START' entries in next segment

S4 - SFC = Index to the most critical of the 'START' entries In next segment

S6 - PFC = Index to the most critical of the 'PRIORITY' entries in next segment

Word 3: Segment Control Word

T1 - LDT = Most critical (lowest) 'DEADLINE TIME' value in next segment

T2 - LST = Most critical (lowest) 'START TIME' value in next segment

T3 - HSP = Most critical (lowest) 'PRIORITY' value in next segment

Word 4: S1 - HELD = Hold indicator to inhibit a run's opening
Ø1 = "CSHOLD RUNID" keyin on console
Ø4 = This run has a 'S' option and the previous run from this input device has not terminated.
Ø1Ø = This run is from a remote 1ØØ4 & must be held until @FIN statement is read.

S2 - P = Current priority value for this run (Initially set to the alphabetic priority on the @RUN statement minus 3)

Figure 3. (1 of 2) SCHQ

S3 - PBL = Link (Index) to next less ciritical priority entry
Ø77 = none (Note PBL not used when FST ≠ Ø)

S4 - PFL = Link (Index) to next more critical priority entry
Ø77 = None (Note PFL not used when FST ≠ Ø)

S5 - AML = Maximum real time priority level allowed
(obtained from account # or system standards)

S6 - AAL = PCT size (obtained from @RUN statement)

Word 5: T1 - FDT = Flexible deadling time of this entry. (The difference in minutes, between critical time and CLT time; where CLT is the time (2400) of the last update by CSU )

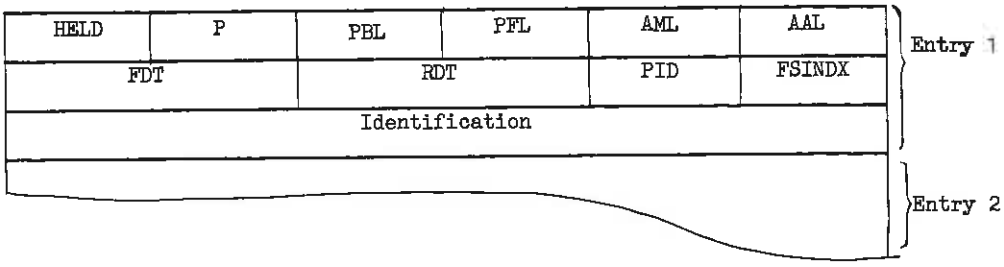
T2 - FST = Flexible start time of this entry (The difference in minutes, between the specified START TIME on @RUN statement and CLT )

S5 - FBL = When FST ≠ Ø: FLB is the link (index) to the next less critical 'START' entry in this segment.
When FST = Ø and FDT ≠ Ø: FBL is the link (index) to the next less critical 'DEADLINE' entry in this segment.

S6 - FFL = When FST ≠ Ø: FFL is the link (index) to the next more critical 'START' entry in this segment.
When FST = Ø and FDT ≠ Ø: FFL is the link to the next more critical "DEADLING entry in this segment.

Word 6: Identification - Unique run identification

Figure 3. (2 of 2) SCHQ




Word Ø: S1 - HELD = 040 implies that FIMAIN was unable to satisfy the facility requests for this entry.

S2 - P = Current priority value for this run

S3 - PBL = Link (Index) to the next less critical 'PRIORITY' entry
077 = None

S4 - PFL = Link (Index) to the next more critical 'PRIORITY' entry
077 = None

S5 - AML = Maximum real time priority level allowed

S6 - AAL = PCT size

Word 1: T1 - FDT = Flexible deadling time of this entry

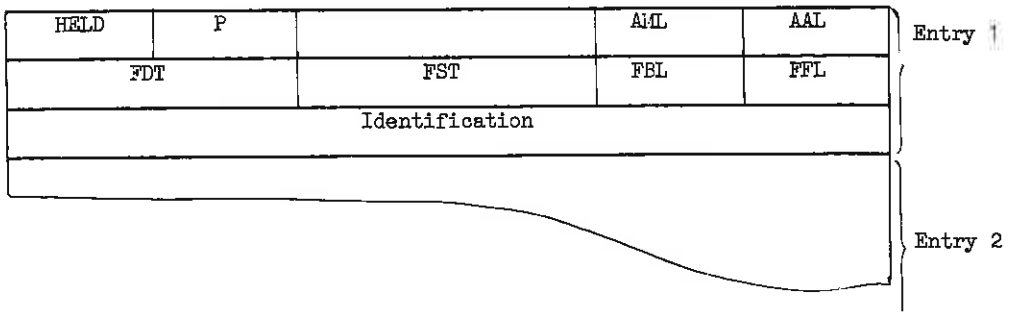
T2 - RDT = Deadline time (2400 value in minutes)

S5 - PID = Alphabetic priority value

S6 - FSINDX = Facility indicator
Ø1 = currently obtaining a facility synopsis

Word 2: Identification - Unique run identification

Figure 4. PRIORQ




Word Ø: S1 - HELD = 1 implies "CSHOLD RUNID" keyin

S2 - P = Current priority value for this run

S5 - AML = Maximum real time priority level allowed

S6 - AAL = PCT size

Word 1: T1 - FDT = Flexible deadline time for this entry

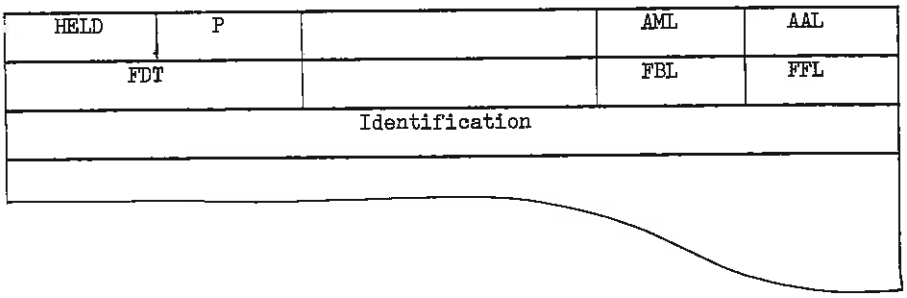
T2 - FST = Flexible start time of this entry

S5 - FBL = Link (Index) to the next less critical "START" entry in this queue.

S6 - FFL = Link (Index) to the next more critical "START" entry in this queue

Word 2: Identification = Unique run identification

Figure 5. STARTQ




Word Ø: S1 - HELD = 1 implies "CSHOLD RUNID" keyin.

S2 - P = Current priority value for this run

S5 - AML = Maximum real time priority level allowed

S6 - AAL = PCT size

Word 1: T1 - FDT = Flexible deadline time of this entry

S5 - FBL = Link (Index) to the next less critical "DEADLINE" entry in this queue

S6 - FFL = Link (Index) to the next more critical "DEADLINE" entry in this queue

Word 2: Identification - Unique run identification

Figure 6. DEADQ

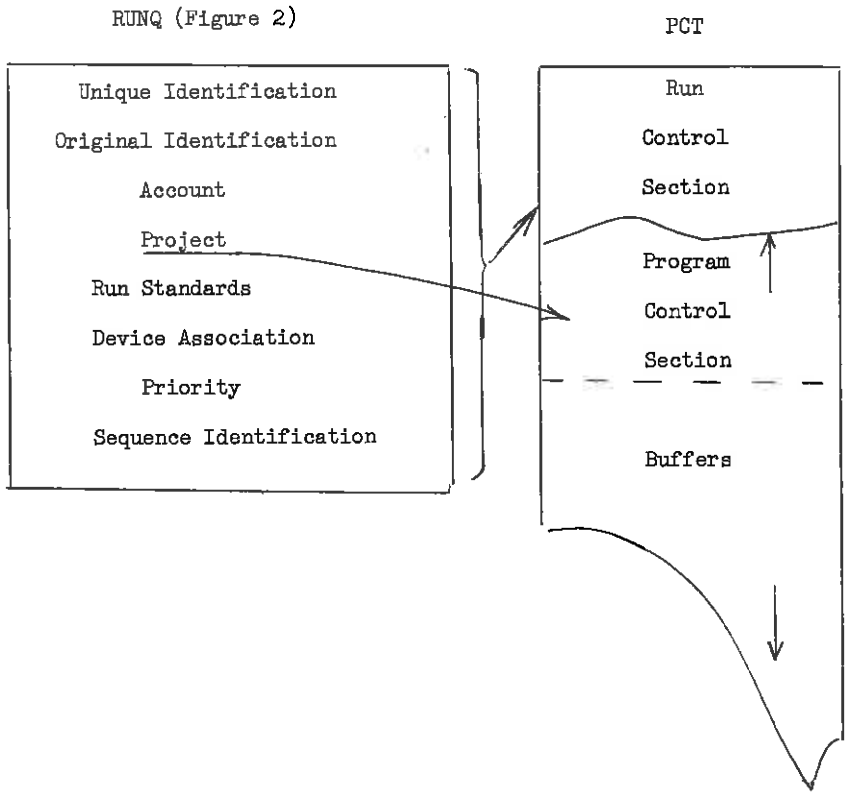
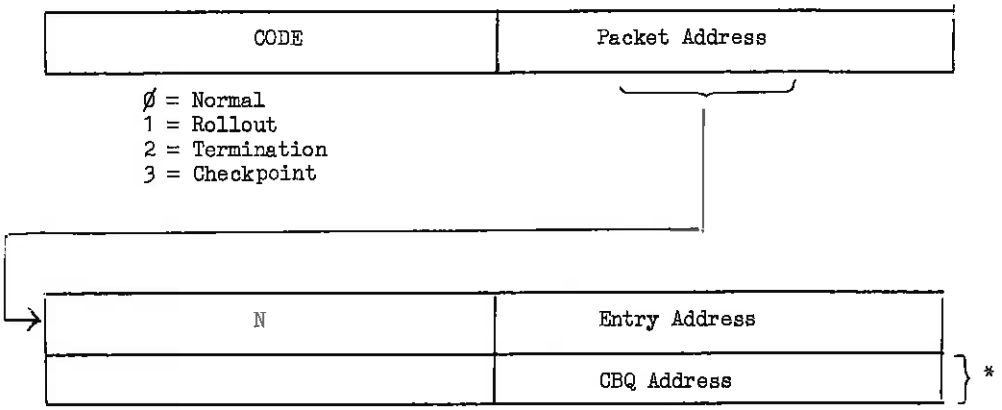

RUNQ (Figure 2)
Unique Identification
Original Identification
Account
Project
Run Standards
Device Association
Priority
Sequence Identification
PCT
Run
Control
Section
Program
Control
Section
Buffers

Figure 7. PCT

Passed To Fimain




*This is the first word of the DFTAB Entry (Figure 8B)

Figure 8A. Packets to FIMAIN

FSTAB Entry

| T/S | TCNT | COND | Location of 1st INFOR |
|---|---|---|---|
| Identification | | | |
| RD$DT | | | CBQ |

PFTAB Entry (within RCT)

| FBC | | | FFC |
|---|---|---|---|
| T/S | TCNT | COND | Location of 1st INFOR |
| Identification | | | |

DFTAB Entry

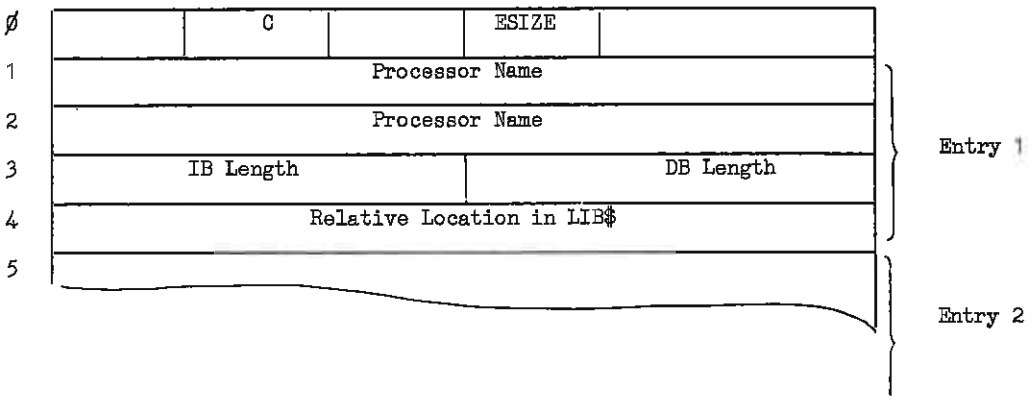
| T/S | TCNT | COND | Location of INFOR |
|---|---|---|---|
| | | | Switch List Identification |

TCNT: Total number of words including the segment control word of the next INFOR segment

COND: for communication between Coarse Scheduler and FIMAIN

Ø3 = Assign when possible
Ø4Ø = facilities assigned
Ø42 = error
H2 = error message address

Figure 8B. Table Entries



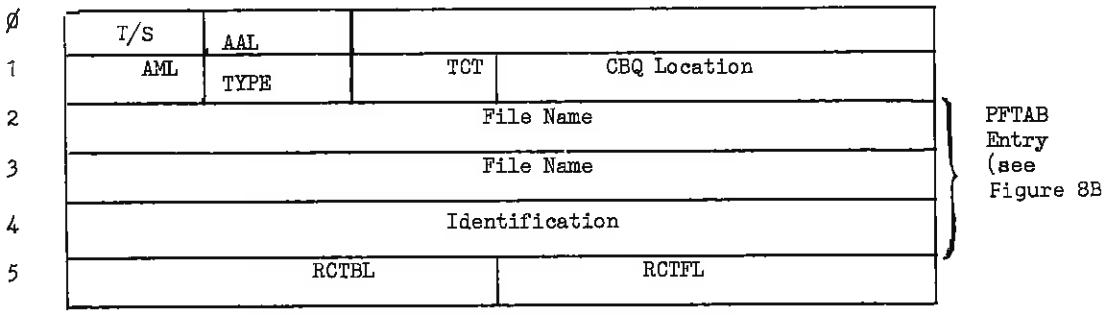

Word Ø: S2 - C = Number of entries in segment
S4 - ESIZE = Entry size

Word 1: & 2: Processor name

Word 3: H1 - IB Length = I bank core requirements (number of words)
H2 - DB Length = D bank core requirements (number of words)

Word 4: Relative location in LIB$ (Header Table Address)

Figure 9. LIBT




Word Ø: S1 - T/S
S2 - AAL - PCT size

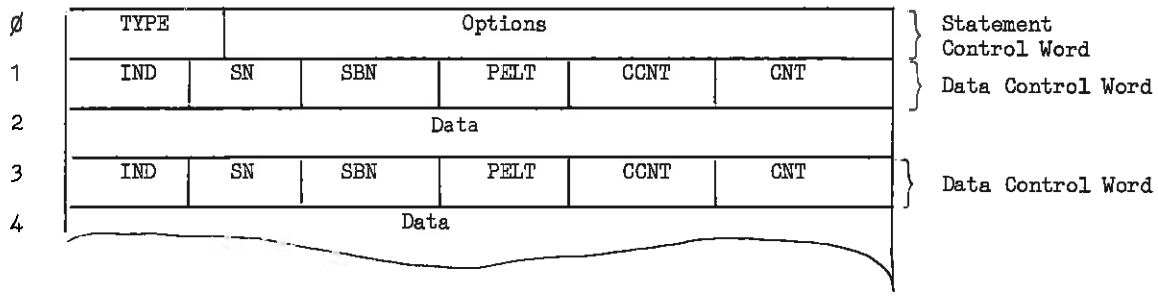
Word 1: S1 - AML = Max. real time level allowed; Ø = not allowed
S2 - TYPE = Ø4 = DEMAND; Ø5 = DEADLINE; Ø6 = BATCH
S3 - TCT = Task control tag
ØØ = Abort keyin
Ø2 = Previous task was a collection implying CSA must look in TPF$.
Ø4 = Hold run from further execution
Ø1Ø = Checkpoint keyin has occurred - call checkpoint function.
Ø2Ø = Restart keyin has occurred - call restart function.
T3 - CBQ location

Word 2: & 3: File Name - for D.A. for Initial Load

Word 4: Identification

Word 5: H1 - RCTBL = RCT backward link
H2 - BCTFL = RCT forward link

Figure 10. Run Control Table (RCT)




Word Ø: Bits 35-26 - Type = Statement type number
i.e. Ø37 = @RUN
ØlØØ = all FUR/PUR commands
Ø5Ø = @XQT
Ø51 = @MAP, @FOR, @COB etc.

Bits 25-0 - Options = Option mask where a bit set indicates option present.
i.e. Bit 25 set implies A option
Bit Ø set implies Z Option

Word 1: Data Control Word = Defines the data image which follows (i.e. Word 2)

S1 - IND = defines data image type
ØØ = subfield of the option field if type ≠ ØlØØ or ≠Ø51
ØØ = command subfield if type = Ø100 or = Ø51
Ø1 = subfield of a specification field
S2 - SN = specification field number (Always = 1 for option subfield)
S3 - SBN = Subfield number
S4 - PELT= Ø75 implies that an element name has been preceded only by a "." (period) implying no file is stated. i.e. .ABC
ØØ implies normal condition. i.e. FILE.ABC
S5 - CCNT= The number of characters (1-6), left justified and space filled, that are present in the last word of the data image defined.
S6 - CNT = The number of words in this data image.

Word 2: Data - Variable length data image in field data

Figure 11. INFOR FORMAT

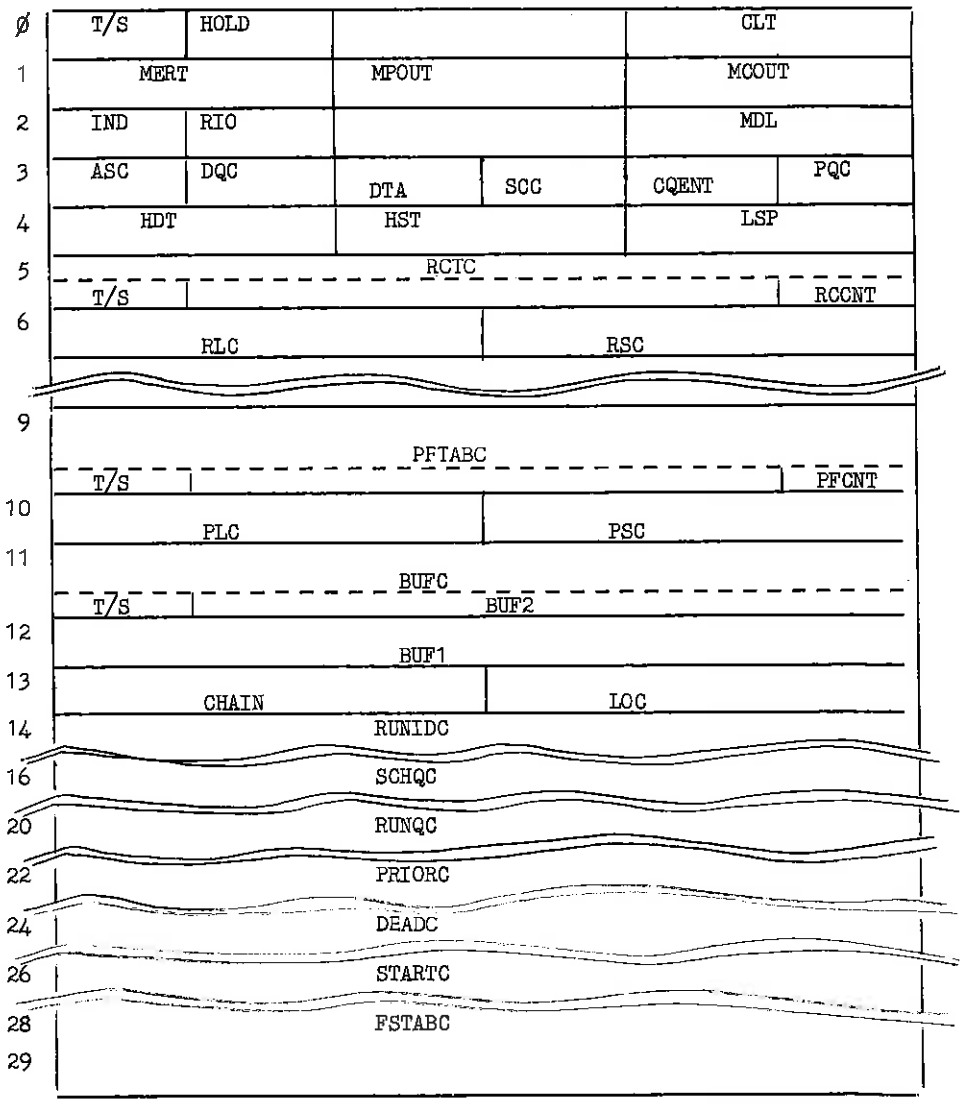



Figure 12. (1 of 3) LOCTAB

Word Ø: S1 - T/S
S2 - HOLD = Master hold on all runs
T3 - CLT = Time of last update

Word 1: T1 - MERT = Maximum time
T2 - MPOUT =Maximum pages
T3 - MCOUT =Maximum cards

Word 2: S1 - IND = Maximum time, maximum pages, maximum cards indicators
S2 - RIO = Adjustment ratio to eliminate abuse of deadline time
T3 - MDL = Minimum deadline time allowed

Word 3: S1 - ASC = Minimum number of entries for PRIORQ
S2 - DQC = Index to most critical DEADQ entry
S3 - DTA = Constant used to determine priority revision for 'DEADLINE'

S4 - SQC = Index to most critical STARTQ entry
S5 - CQENT = Minimum number of CQE's (implies runs open)
S6 - PQC = Index to most critical PRIORQ entry

Word 4: T1 - Highest deadline time value (FDT) in DEADQ
T2 - Highest start time value (FST) in STARTQ
T3 - Lowest priority value (P) in PRIORQ

Word 5: RCTC - RCT control word
S1 - T/S
S6 - RCCNT - number of RCT's

Word 6: H1 - RLC = Location of last RCT
H2 - RSC = Location of first RCT

Word 9: PFTABC - PFTAB control word
S1 - T/S
S6 - PFCNT - number of PFTAB's

Word 10: H1 - PLC = Location of last PFTAB
H2 - PSC = Location of first PFTAB

Word 11: BUFC
S1 - T/S
S2 - S6 = BUF2 (see BUF1)

Word 12: BUF1 - (along with BUF2) form a 64 bit field representing the 28 word buffers (Sectors) of the 1792 word track on mass storage.
Bit = 1 means buffer in use
= 0 means buffer is available

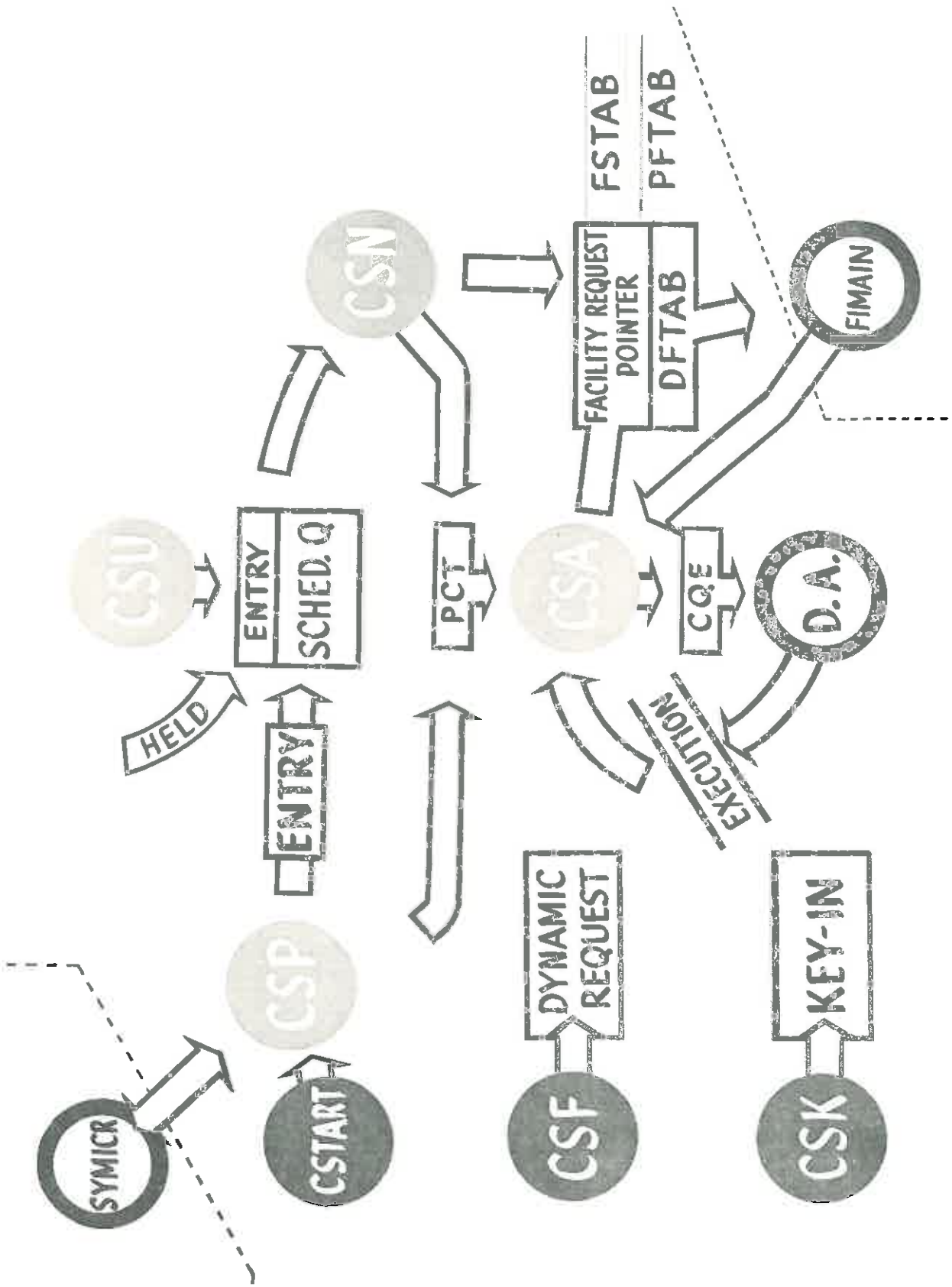
Word 13: H1 - CHAIN = Absolute core address (EXPOOL) which holds the control words for the next track of 1792 words.
H2 - LOC = The relative starting address of this 1792 word track.

Figure 12. (2 of 3) LOCTAB

```
Word 14:  *RUNIDC - RUNIDQ segment control word
   & 15:
Word 16:  SCHQC - SCHQ segment control words see SCHQ (Figure 3)
   - 19:
Word 20:  RUNQC - RUNQ segment control words - see RUNQ (Figure 2)
   & 21:
Word 22:  *PRIORC - PRIORQ segment control words
   & 23:
Word 24:  *DEADC - DEADQ segment control words
   & 25:
Word 26:  *STARTC - STARTQ segment control words
   & 27:
Word 28:  *FSTABC - FSTAB segment control words
   & 29:
```

*see definition of RUNIDQ (Figure 1) segment control words

Figure 12. (3 of 3) LOCTAB


FSTAB
PFTAB
CSN
FACILITY REQUEST POINTER
DFTAB
FIMAIN
CSU
ENTRY
SCHED. Q
PCT
CSA
CQE
D.A.
HELD
ENTRY
EXECUTION
CSP
DYNAMIC REQUEST
KEY-IN
SYMICR
CSTART
CSF
CSK

# 4. DYNAMIC ALLOCATOR

General

The Dynamic Allocator has the responsibility for the distribution of core space among user and EXEC tasks in a multi-programming environment.

The main elements of the Dynamic Allocator are:

1. DA
   a. Entry Routines
   b. Storage Control
2. DACCC (Core Contents Control)
3. DASFI and related elements for control of Swap File space

The main functions of the Dynamic Allocator are:

1. Allocation of Core
2. Release of Core
3. Initial Load of Tasks
4. Swap-out of Tasks
5. Reload of Tasks Previously Swapped Out

The tables, packets and queues used by the Dynamic Allocator are:

| | | |
|---|---|---|
| 1. | Core Queue Entry (CQE) | Figure 1 |
| 2. | Chained Core Request (CCR) | Figure 2 |
| 3. | Level Control Word (LCW) | |
| 4. | Core Request Queue (CRQ) | |
| 5. | Instruction Packets | Figure 3 |
| 6. | I/O Packets | Figure 4 |
| 7. | Core Maps | |
| 8. | Mass Storage Maps | |

Entry Routines

For each type of requirement, the requestor enters DA at the entry routine for that specific request. The requestor defines the request to DA in the Core Queue Entry for the subject run.

The entry routine builds a Chained Core Request based on the information passed to it via the CQE, makes the proper entries in the LCW and CRQ, and deactivates the requestor. Control is then passed to the Storage Control Section of DA.

Storage Control

The Storage Control Section of the Dynamic Allocator (DASC) controls the actual distribution of core space.

The principle functions of the Storage Control Section are as follows:

1. Check level control words for outstanding core requests.
2. Select possible candidate on priority basis.
3. Index into core request queue for address of request.
4. Search for memory requested.
5. Build inst. packets for DACCC.
6. Update core maps.
7. Activate DACCC.
8. Deactivate if no more requests outstanding.

Within the Dynamic Allocator a queueing scheme is used. To maintain the core requests, this scheme is broken down into type and level. First, the types are used to define the request type being requested of the DA. All requests to the DA for core will fit into one of these types.

Within each of these types are levels at which core requests could reside: core requests are chained independently on each of these levels. To help reference this priority of queue, there are Level Control Words. One word represents each type and one bit in each word represents each level of that type.

The Level Control Words are comprised of two memory locations for each priority type.

The LCWs are tested for the presence of bits which would indicate outstanding requests. Upon finding bits in the first half of the LCWs,Storage Control calculates a value for indexing into the Core Request Queue.

Assuming core space is found for a request, the following action is taken by DASC:

1. Builds instruction packet for DACCC.

LWCs, 2. Updates LWCs, CRQ and core maps.

3. Activates DACCC if inactive.

Core Contents Control

The Core Contents Control Section of the Dynamic Allocator controls the I/O operations as required.

The input or output packet as appropriate, is built by DACCC from information contained in the 6 word instruction packet passed to it by storage control; the I/O operation is performed, and the instruction packet is released from EXPOOL.

DACCC consists of a control section and processing sections for the major categories of operations it is asked to perform.

1. The control section CCC is an independent activity which scans the instruction packet list on a first in first out basis and passes control to the proper processing sections. CCC recognizes the following requests:

| | |
|---|---|
| a. Reload | ITRL |
| b. Swap Out | ITSO |
| c. EXEC D Bank Acquisition | ITED |
| d. EXEC I Bank Acquisition | ITEI |
| e. Fixed Block Acquisition | ITFB |
| f. Initial Load | ITIL |
| g. Re-entrant Processor Load | ITRP (Not Now in DACCC) |
| h. PCT Expansion | ITEP |
| i. Program D Bank Expansion | ITPD |
| j. Program I Bank Expansion | ITPI |
| k. PCT Acquisition | ITAPCT |
| l. Absolute Load | ITABS (Not Now in DACCC) |
| m. Suspend | ITSP (Not Now in DACCC) |

2. Reload (ITRL) is handled by the independent activity CCCR. After reloading the task and reactivating it, the activity requesting the reload is activated. During the housekeeping phase of CCCR, the PCT address is updated in the core queue entry, and the PCT is changed to reflect the programs current positions in core.

3. Swap-out (ITSO) is handled by the routine CCCS. The task's I & D Banks are swapped out into the area specified in the instruction packet.

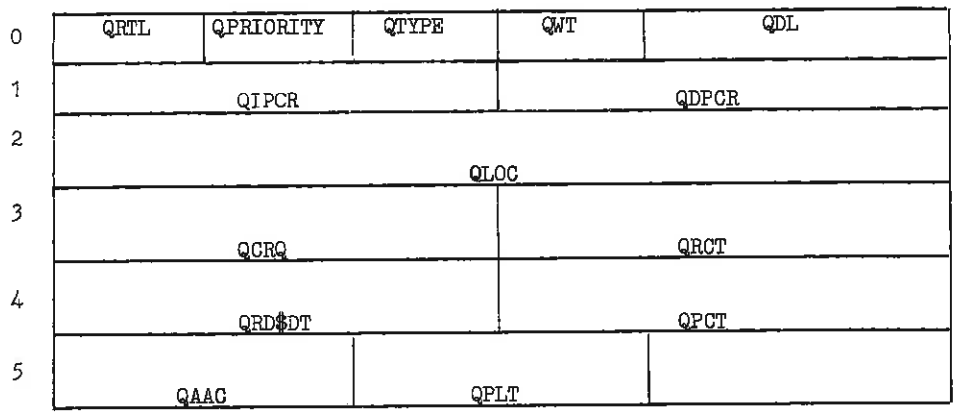
4. EXEC D Bank, I Bank and fixed block acquisition (ITED, ITEI and ITFB) are handled by CCEC with separate entrances for each type (CCED, CCCEI, and CCCF). CCEC activates the requestor.

5. Initial Load (CCCL) is handled by the independent activity CCCLOO. CCCLOO loads the PCT of the task to be loaded, modifies the PCT temporarily so that the remainder of the load will be an independent activity of the task to be loaded, then builds the Switch List Entry and activates the task. When the task itself is loaded and the PCT has been properly set up for running, the initial load requestor is activated and control is passed to the DISPATCHER.

6. PCT expansion (ITEP) initializes the additional PCT area and activates the requestor.

7. Program D and I Bank expansion (ITPD and ITPI) are handled by a common routine which modifies the PCT to reflect the expansion and reactivates the requestor.

8. PCT acquisition (CCCPCT) initializes the PCT area and activates the requestor. Control is then returned to CCC.




Word 0:
- S1 - QRTL = Real time level
- S2 - QPRIORITY = Program's priority
- S3 - QTYPE = Program's type
- S4 - QWT = Deadline weight
- T3 - QDL = Deadline time

Word 1:
- H1 - QIPCR = I portion core requirements (Number of words)
- H2 - QDPCR = D portion core requirements (Number of words)

Word 2:
- QLOC = Relative file location for initial load

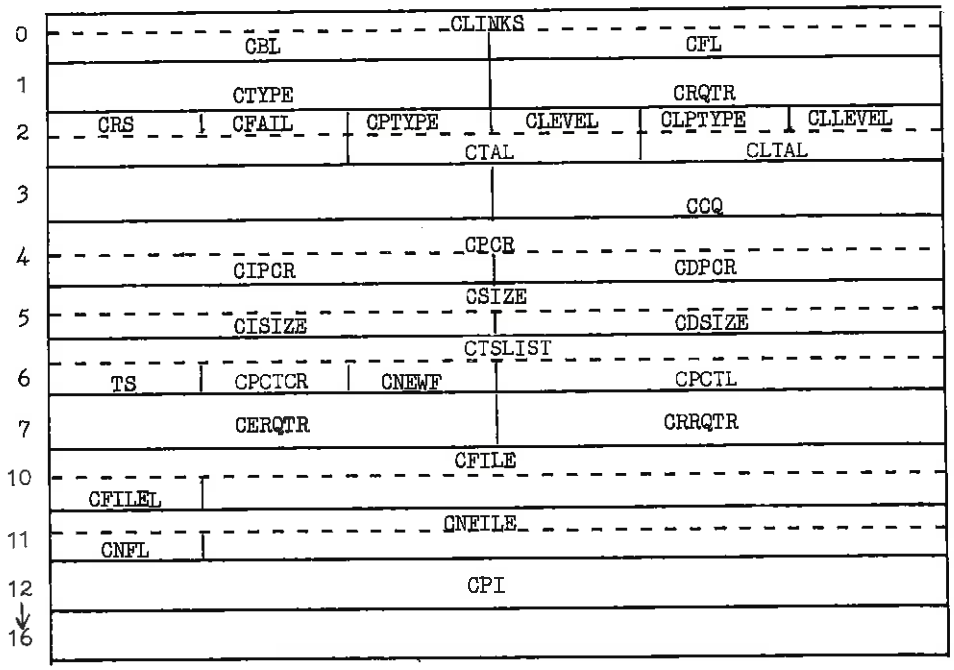
Word 3:
- H1 - QCRQ = Core request queue entry address (Relative)
- H2 - QRCT = Run control table address (Relative)

Word 4:
- H1 - QRD$DT = Read - Print control table address (Relative)
- H2 - QPCT = Program control table address (Absolute)

Word 5:
- T1 - QAAC = Active activity count
- T2 - QPLT = Present level total

} DA Modification Used to Determine Swapping Priority

Figure 1. Core Queue Entry/Core-Swap Queue Entry




Word 0 : CLINKS = Forward & Backward links
H1 - CBL = Backward link (Relative address of previous entry)
H2 - CFL = Forward link (Relative address of next entry)

Word 1
H1 - CTYPE = Request type (See "ITYPE" in instruction packet)
H2 - CRQTR = Requestor list (Switch list entry during PCT acquisition only; Also is used for temporary storage)

Word 2
S1 - CRS = Request State
00 - In core
-01 - Swapped out (RSSO)
-02 - Reload in progress (RSIP)
S2 - CFAIL = Last unsuccessful pass number
S3 - CPTYPE = CRQ type
S4 - CLEVEL = CRQ level
S5 - CLPTYPE= Last CRQ type
S6 - CLLEVEL= Last CRQ level
T2 - CTAL = CRQ type & level (Dependent upon kind of request; who made request, etc,. It is not the program's actual type & level, but a calculated type & level for priority placement in level control word)
T3 - CLTAL = Last CRQ type & level

Word 3
H2 - CCQ = Core queue entry (Relative address)

Word 4 : CPCR = Program core requirements
H1 - CIPCR = I Portion core requirements (Number of blocks including PCT requirements)
H2 - CDPCR = D Portion core requirements (Number of blocks)

Figure 2.(1 of 2) Chained Core Request

```
Word 5 : CSIZE    = Current program size
    H1 - CISIZE   = I portion size (Number of blocks including PCT requirements)
    H2 - CDSIZE   = D portion size (number of blocks)

Word 6 : CTSLIST  = Test and set indicator (List)
    S1 - TS (For word 7 usage)
    S2 - CPCTCR   = PCT core requirements (Number of blocks
    S3 - CNEWF    = New swap file necessary (If set release old swap file)
    H2 - CPCTL    = PCT length (Number of words)

Word 7
    H1 - CERQTR   = Expand requestor list  } Switch list address chain
    H2 - CRRQTR   = Reload requestor list  } For reactivating requestor

Word 10 : CFILE   = Swap file relative address
    T1 -  CFILEL  = Swap file length (Number of tracks)

Word 11 : CNFILE  = Next swap file relative address
    T1 -  CNFL    = Next swap file length

Word 12 : CPI       Pseudo-interrupt bit map
```

Figure 2. (2 of 2) Chained Core Request

| | | |
|---|---|---|
| 0 | ITYPE | INI |
| 1 | IIPCD | |
| 2 | IDPCD | |
| 3 | IRQTR | ICQ |
| 4 | IPCT | |
| 5 | IFILE | |

Word 0:
H1 - ITYPE = Instruction Type
ITRL = 0000001 = Reload
ITSO = 0000002 = Swap-Out
ITIL = 0000004 = Initial load
ITEI = 0000010 = EXEC I Bank acquisition
ITED = 0000020 = EXEC D Bank acquisition
ITFB = 0000040 = EXEC Fixed block acquisition
ITRP = 0000100 = Re-entrant processor
ITPI = 0000200 = Program I portion expansion
ITPD = 0000400 = Program D portion expansion
ITAPCT = 0001000 = PCT acquisition
ITABS = 0002000 = Absolute load
ITEP = 0004000 = Expand PCT
ITSP = 0100000 = Suspend
H2 - INI = Next instruction packet address (Relative)

Word 1 : IIPCD = I portion core descriptor

CORE DESCRIPTOR FORMAT

| Start Bank | Start Block | Number of Blocks | Number of Banks | Last Bank | Last Block |
|---|---|---|---|---|---|

S1 - Start bank number (Module number)
S2 - Start block number within start bank
S3 - Number of blocks within start bank excluding first bank
S5 - Last bank number (Module number)
S6 - Last block number within last bank
S4 - Number of entire banks between start bank and last bank

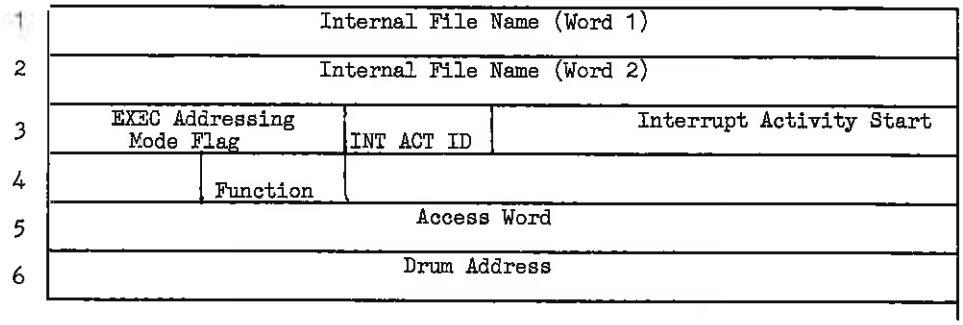
Word 2 : IDPCD = D portion core descriptor (See format for word 1)

Word 3 :
H1 - IRQTR = Instruction requestor (Same as "CRQTR" in core request entry)
H2 - ICQ = Core queue entry address (Relative)

Word 4 :
H1 - IPCT = Program control table address (Absolute)

Word 5 : IFILE = File packet address (Same as "CFILE" in core request entry)

Figure 3.Core Contents Control Instruction Packet




Word 1 and Word 2 : The internal file name used in all references to the file. This name is either the same as some external file name of the @ASG statement or is attached to an external file name by a @USE statement.

Word 3 : T1 - Used by the EXEC to determine mode of addressing.
S3 - Interrupt activity identification
H2 - Interrupt activity starting address

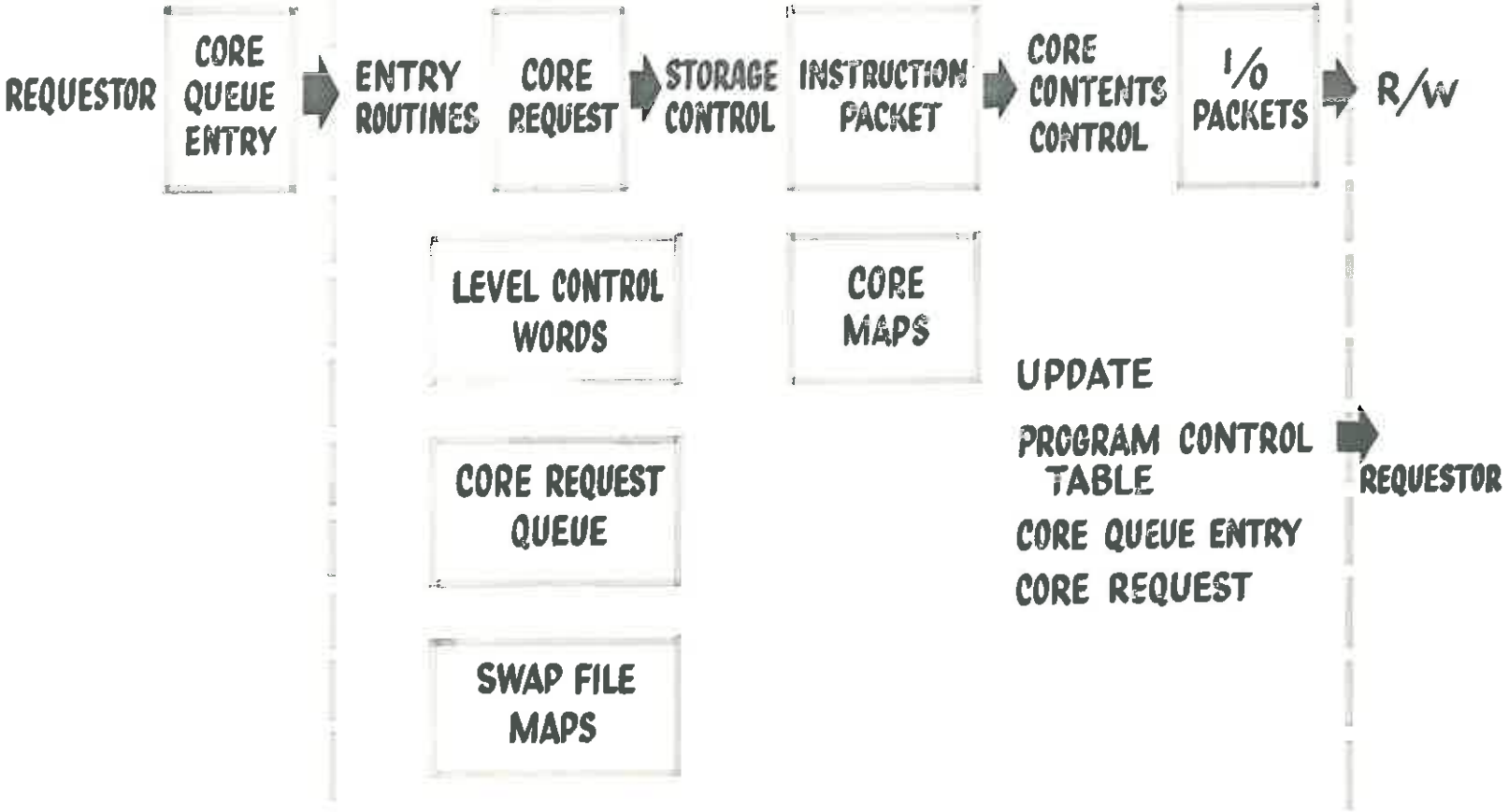
Word 4 : S2 - The code denoting the function to be performed.

Word 5 : The fifth word of the packet is an I/O access word specified in the format defined in the UNIVAC 1108 multi-processor system description, UP-4046 REV. 2 That is, Bits 35-34 are the increment decrement designator, bits 33-18 contain the number of words to transfer, and bits 17-00 contain the address at which transfer is to begin. For GW$, SCR$, and SCRB$ functions, this word contains the number of access words and the address at which the string of access words begin.

Word 6 : For magnetic drum files, this word contains the mass storage address at which the described I/O operation is to start. This address is relative to the start of the mass storage file; the handler provides for determining the absolute position. For fastrand files (or simulated fastrand on other type drums the address is the start of a sector and consecutive addresses are 28 words apart.

Figure 4. I/O Packet

# DYNAMIC ALLOCATOR (D.A.)



# 5. DISPATCHER

In a multi-programming environment time slices are granted dynamically by the Dispatcher according to the priority and needs of the user's programs. The user to the Dispatcher is any activity requesting CPU time.

The tasks used to control CPU time are:

1. FORK - initiate new activities; build SWL.
2. ACT/DEACT - place/remove SWL into/from SWL queue.
3. DAPA - maintain Demand/Batch ratio.
4. Dispatcher - switching activities for time slices.

These tasks are used to maintain tables necessary to control system throughput. These tables are:

1. PCT - Program Control Table.
2. SWL - Switch List.
3. ASA - Activity Save/Status Area.
4. Minor Save Area.
5. Major Save Area.
6. Substatus Save Area.

The Dispatcher grants time slices to activities indicated in the switch list queue. CPU utilization is concerned primarily with the SWL. The tasks which build the SWLs are DACCC (Initial Load) and FORK. DACCC is responsible for building the program's initial SWL. Subsequent activities in a program are granted CPU time by the Dispatcher from their SWL which is built by the routine FORK.

The portion of the initial load routine in DACCC concerned with building the programs first SWL is basically the same as the routine FORK. The following discussion will concern FORK in detail, and how it gets an activity into the SWL queue ready for time slices.

FORK performs the following functions.

1. Build SWL for new activity.
2. Save register information in save areas.
3. Activate new SWL.

To activate a new SWL the routine ACT is called. ACT performs the following tasks:

1. Mark SWL ready for execution.
2. Link SWL into SWL queue.
3. Set SWL control word.

In a multi-programming system there must be a method of dynamically adjusting the priority at which a program or a group of programs will execute. Through the system parameter DMAX and the routine DAPA, EXEC 8 will adjust and maintain a specified ratio of Demand/Batch jobs. The parameter DMAX is the percentage of total CPU time which Demand will be given for execution. DAPA will adjust the queue of SWL to maintain the proper Demand/Batch ratio of CPU time.

The other main function of DAPA is to insure that the routine CSN is activated periodically. CSN is the routine used to select the next Batch job to be opened. Demand activities are adjusted according to the parameter DMAX to maintain the proper Demand/Batch balance. This is used in the Dispatcher to help determine the highest priority SWL to be granted a time slice.

The Dispatcher looks only at the SWL queue for activities to be switched. The functions of the Dispatcher are:

1. Assure all interrupts are processed.
2. Update program accumulated time.
3. Select next activity to switch into execution.
4. Save stopped activity.
5. Load new activity.
6. Switch to activity.

PCT (Program Control Table)

Various tables were discussed individually as they were used by EXEC 8.

1. RD$DT - Symbiont Complex
2. PCT - Coarse Scheduler
3. CQE/CCR - Dynamic Allocator
4. SWL - Dispatcher

The EXEC needs these tables to give itself the flexibility necessary to dynamically handle interrupts, allow for real time processing, allocate/release central memory and perform system I/O. To obtain greater system efficiency, there is a control table for all user/EXEC interactions. This is the PCT (Program Control Table.)

Run Control Section

| | | | |
|---|---|---|---|
| *Original Run Identify | | | |
| *Generated Run Identity (Unique ID) | | | |
| Total Run Time for Completed Program | | | |
| *Estimated Run Time (200 Microsec. Inc.) | | | |
| *Acct Prty | FI-Abort | *T Option | Core Queue Entry Address |
| Qual.Tbl.Start | | | Activity Name Tbl |
| Program Contigency Control | | | |
| Console Requests | | Log Requests | ESI Accnt Nmbr |
| Total I/O Requests | | | |
| Total I/O Data Transfers | | | |
| 432 - Accumulation of Drum Usage Time | | | |
| 880 - Accumulation of Drum Usage Time | | | |
| F2 - Accumulation of Drum Usage Time | | | |
| FB - Accumulation of Drum Usage Time | | | |
| Run Start Time and Data (TDATES$ Format) | | | |
| XQT Options (Bit 25=A ---Bit 0=Z) | | | |
| Termination Conditions For Program | | | |
| * Most Recent Qualifier | | | |
| (Other Qualifiers Linked in PCT Buffers) | | | |
| *Account Number (12 Characters) | | | |
| * 2nd Word Acct. No. | | | |

Acct Prty - Priority established for this account number
FI-Abort - An Abort flag for facility inventory
T-Option - Termination options from the run control image

Figure 1. (1 of 3) Program Control Table

Program Control Section

| | | |
|---|---|---|
| Amount of Quantum Used (Demand) | | Quantum (200 Microsec. Inc.) |
| Total Accumulated Run Time (200 Microsecond Inc.) | | |
| Real Time Activity Count | | Swap Lock Counter |
| Segment Load Control | | |
| Activities Released Via AWAIT$ (Bit 34=ACT 1--Bit 0=35) | | |
| Mask Control Word of Existing Activities | | |
| Activity Mask of Activity 1<br>Used For<br>Naming Activities<br>and Performing<br>AWAIT$ Requests (See PRM)<br>Activity Mask of Activity 35 | | |
| Pointer to ASA Chain | | Current Activity Count |
| Cond of Run | Swap-Ind | Linkage to CPOOL$ Ref |
| | Gr Tbl Flg | Prog Term Flg |
| | | |
| Ibank Core Descriptor | | |
| Dbank Core Descriptor | | |
| Relative Ibank Value | | IB/DB Separator Value |
| Swap Indicator | | |
| Program File Name Used | | |
| by LOAD$ and PMD | | |
| PCT Format in ERR FL | | Header Table Addr. (Current Task) |
| Termination Counter | | Initial Time Quantum |
| **ESI Contingency Control | | |

Figure 1. Program Control Table
(2 of 3)

| | |
|---|---|
| Swap Count | Count of Outstand I/O Reqsts |
| Item Chain Start | Item Chain End |
| DACT Er Flg \| PCT TM Flg | ER DACT Chain |
| | ESI Activity Count |
| ESI Activity Storage Limits | |
| Suspend Flag | Buffer Addr. For PMD |
| PCT Linking Control | |
| Name Item Control | |
| File Name (Word 1) | |
| File Name (Word 2) | |
| Assign and Name Item Buffers | |
| Buffer | Link to 1st 4 Wd Buf or 0 |
| Control | Link to 1st 8 Wd Buf or 0 |
| Words | Link to 1st 16 Wd Buf or 0 |
| For | Link to 1st 32 Wd Buf or 0 |
| PCT | Link to 1st 64 Wd Buf or 0 |
| Buffer | Link to 1st 128 Wd Buf or 0 |
| Area | Link to 1st 256 Wd Buf or 0 |
| | Link to 1st 512 Wd Buf or 0 |
| PCT Buffers | |

*Initial Information established by the Coarse Scheduler from the RUNQ Entry.
**See PRM for Contigency Explanation.

Figure 1. (3 of 3) Program Control Section

| Next SWL | | | Last SWL | | |
|---|---|---|---|---|---|
| ASA Address (ABS) | | | PCT Address (ABS) | | |
| RD$DT Address | | | Swap IND | CKPT IND | Original Type |
| Wait Indicators | | | SAM | Key IND | Location |
| Present Type | Present Level | Original Level | Core Queue Address | | |
| Wait Time for Timed Queue | | | | | |
| Contingency Mask | Contingency Indicator | | Activity Contingency Address or Ø | | |

Word 5 is also available for non-permanent information.

Wait Indicators:
- Bit 18 - EXPOOL
- 19 - I/O control
- 20 - I/O control
- 21 - Console Control, PCT expansion hold
- 22 - Checkpoint
- 23 - Waiting for reload
- 24 - Waiting for Core, EXPOOL expansion
- 25 - Symbiont
- 26 - Block buffering
- 27 - Program file package
- 28 - Termination
- 29 - EMODE
- 30 - AWAIT$
- 35 - Stop

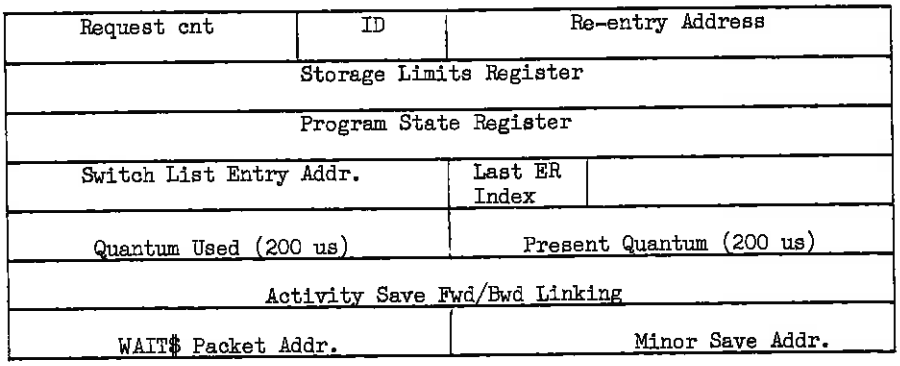
ESI Indicators:
- Bit 12 - ESI SAM
- 13 - Post processing completed: 0=no, 1=yes
- 14 - (not used)
- 15 - Mode: 0=single, 1=multiple
- 16 - Entry used as ESI activity
- 17 - Presently executing in system

Key Indicators:
- Bit 06 - Swap in
- 10 - Name defined
- 11 - Activity interrupted

Location Indicators:
- 4 - Job in timed activity queue
- 3 - Job presently executing or in ER queue
- 2 - Job available for execution
- 1 - Job presently waiting

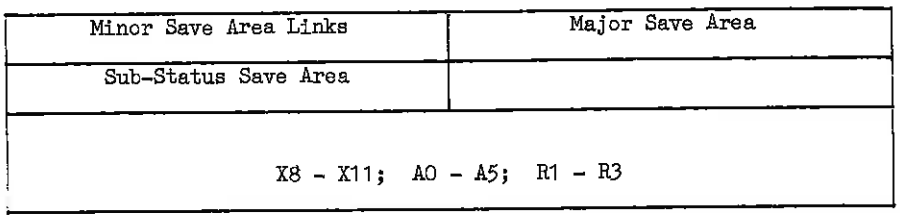
Figure 2. Switch List

Activity Status Area (ASA) *



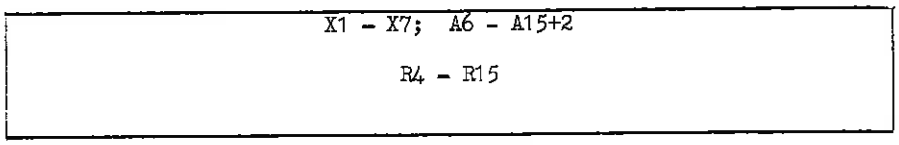

7 Words In Length

Minor Register SAVE AREA *



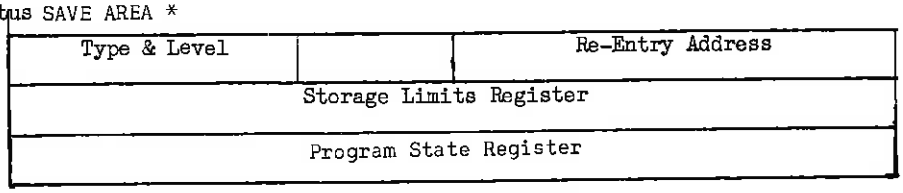

15 Words In Length

Major Register SAVE AREA *




31 Words In Length

Sub-Status SAVE AREA *




3 Words In Length

* Build to PCT Buffer

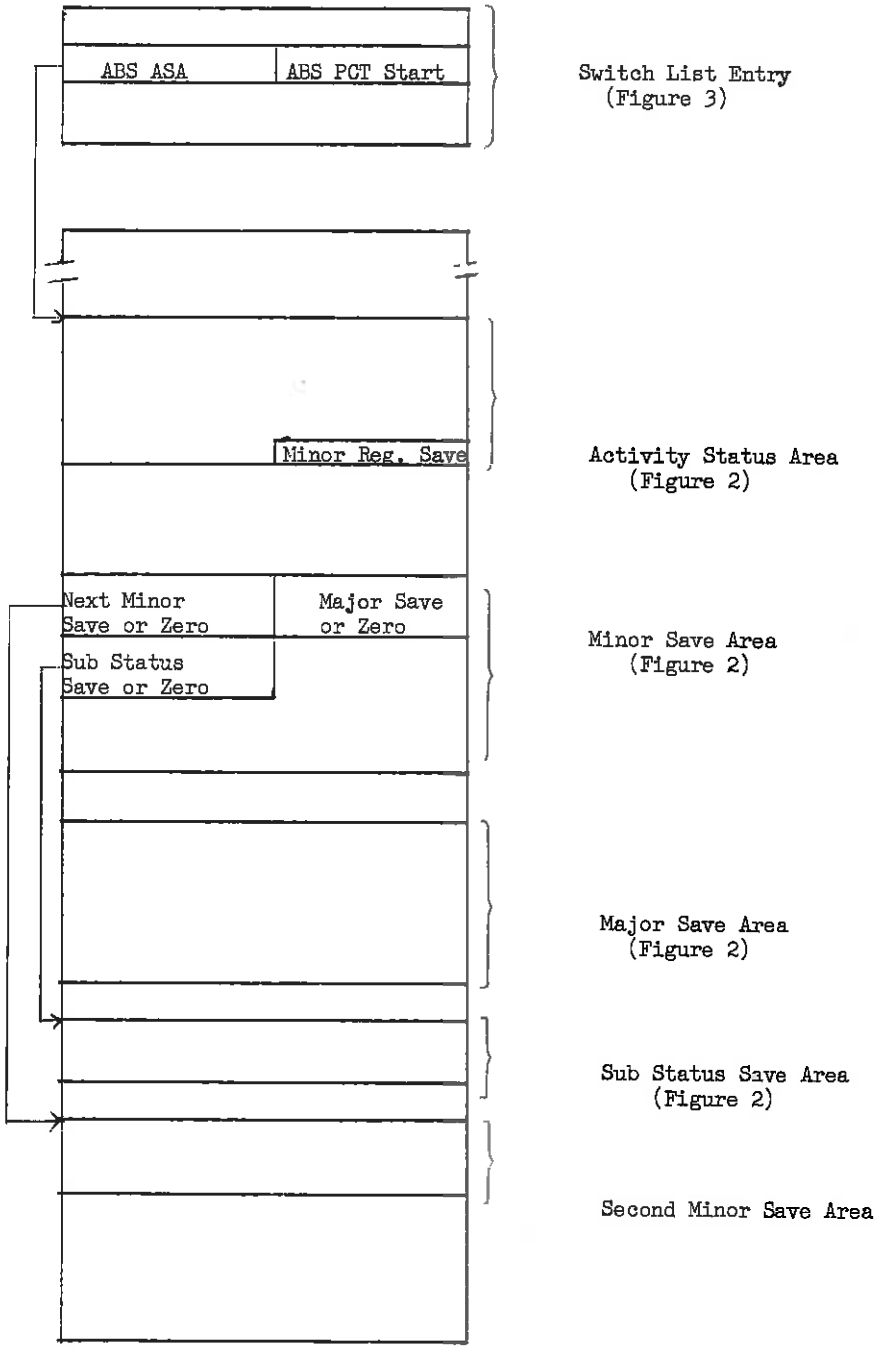
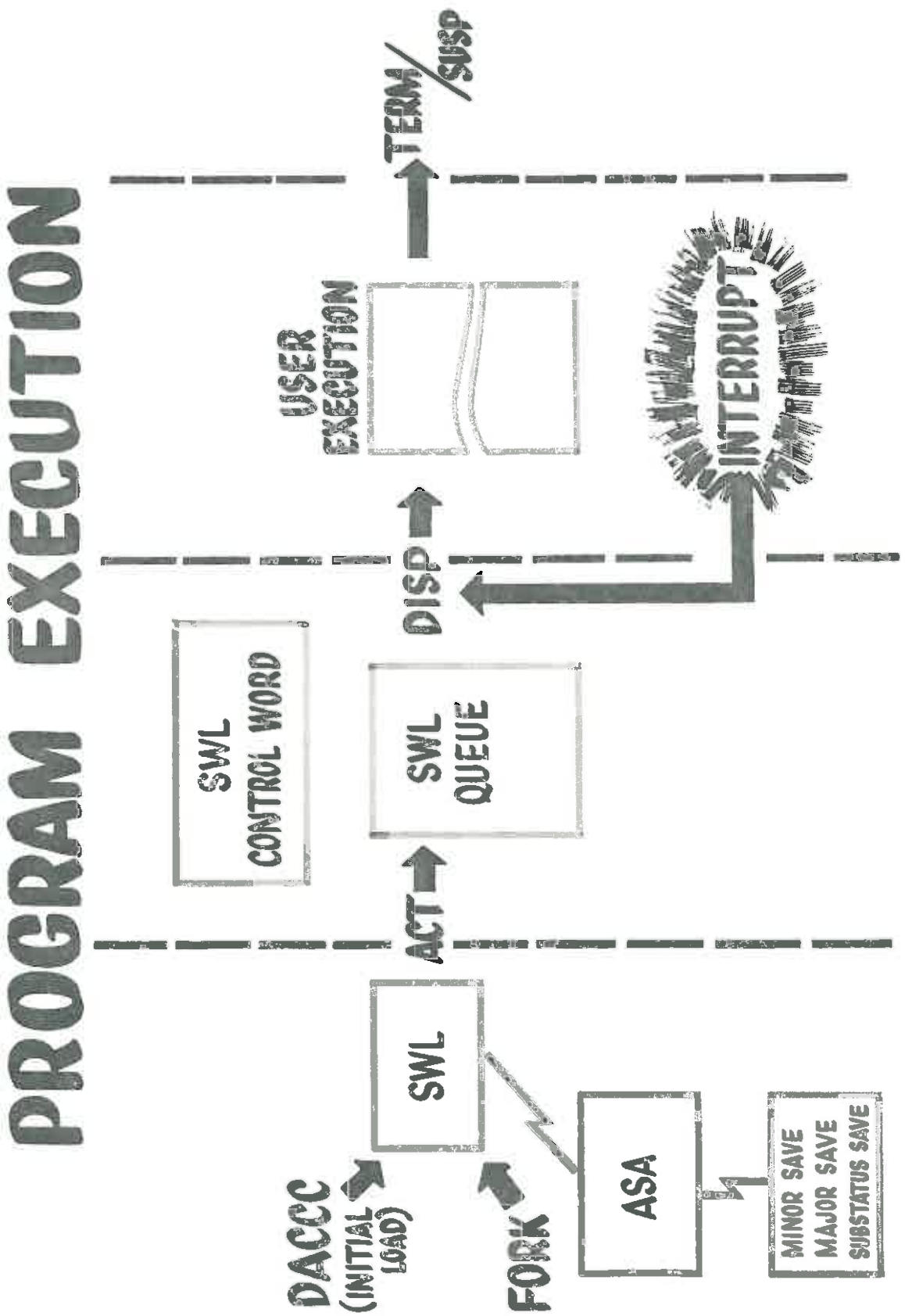



Figure 4, General Layout of Activity


PROGRAM EXECUTION
TERM/SUSP
USER EXECUTION
INTERRUPT
DISP
SWL CONTROL WORD
SWL QUEUE
ACT
SWL
DACCC (INITIAL LOAD)
FORK
ASA
MINOR SAVE
MAJOR SAVE
SUBSTATUS SAVE

# 6. TERMINATION

A RUN passes through various phases of a 'Life Cycle' from the time it enters the system until it finally is removed from the system. The primary events in this cycle are the creation or removal of:

1. The unique RUN ID, established by the Coarse Scheduler (CS) upon the introduction of the RUN to the system.
2. The RUN with its initial Program Control Table (PCT) established by the CS when the run is selected.
3. The PROGRAM, opened by the CS is loaded by the Dynamic Allocator at the request of the Coarse Scheduler.
4. The ACTIVITY, created during initial load by the Dynamic Allocator.

The expansion of this 'Life Cycle' is referred to as the initiation phase. TERMINATION may be defined as the orderly and sequential reversal of the initiation phase.

Elements

There are six elements involved in the termination complex.

Their names and their functions are:

1. TERM : The normal and abnormal termination of all activities.
2. EXITFØ : The termination and cleanup of all programs.
3. REMID : The termination of all runs and removal of the run ID.
4. XKEY : The initiation of requests for termination from the console.
5. EIH : The initiation of error termination by hardware interrupt.
6. ERRFO : The diagnosis and control of error termination requests.

Normal Termination Requests

Only the user is in a position to know when an activity has executed normally. Therefore, only the user is allowed to request normal termination. This request is made through an ER to EXIT$. This request has no effect on any other activities.

Abnormal Termination Requests

Abnormal requests fall into two catagories.

The first is a request for error termination. The user makes the request through an ER to ERR$ and the EXEC makes its request through the tag EMODE found in the element TERM. As in the case of normal termination, only the requesting activity is terminated. The distinction between normal and error termination is the error message and the control register dump that is placed in the users PRINT$ file by the element ERRFØ.

The second type of request is for abort termination. The user makes his request through an ER to ABORT$. This will terminate the requesting activity and subsequently force all remaining activities into error termination. The EXEC has a slightly different method of aborting a program. It applies the same type of force which causes the error termination of all activities but does so through the independent use of a routine that modifies the values used in the storage limits register.

The Role of Storage Limits in Termination

Every activity has an Activity Save Area within the Run's PCT. One of the units of information found in the ASA pertains to the values that are loaded into the Storage Limits Register during execution. These values are used by the hardware to monitor all accesses to central memory. If an attempt is made to access memory outside of the boundaries imposed by these SLR values, a guard-mode interrupt occurs.

A guard-mode interrupt can be forced by every access to memory by replacing the valid storage limits values with a standard set of octal values calculated to cause just such an interrupt. This standard set is 000,777,000,777. In comparison with this set of values, any central memory address will be greater than 000 (the upper boundary) and less than 777 (the lower boundary) and therefore out of bounds.

A routine to affect this replacement is available for use by the EXEC and in effect to the user through a request for an ABORT$ termination.

TERM

The element TERM handles the termination of all activities, regardless of the manner in which termination was requested. The following steps are taken to remove each activity:

1. Clean Up I/O:
   a. The lock on all file items read with the Read and Lock function is released.
   b. All queued I/O requests are either removed from the queue or are allowed to complete.
2. Release Save Area. The following buffer areas are released back to the PCT:
   a. Activity Save Area.
   b. Minor Register Save Area.
   c. Major Register Save Area, if used.
   d. Sub-statues Save Area, if used.

3. Decrement Activity Count.

4. Ambiguity Check:
   1. Program lockup due to faulty programmer logic.
   2. All remaining activities are in a wait state.

5. Release Switch List Entry.

6. Make Log Entry.

ERRFØ

ERRFØ may be entered through a request for ERR$ or ABORT$ termination, from the EXEC calling for an EMODE termination or in response to a guard-mode interrupt being handled by the element EIH.

The functions of ERRFØ are to:

1. Verify Residence; insure the residence of a program prior to termination of an activity.

2. Check Storage Limits; if they have been collapsed, steps 3 and 4 will be bypassed.

3. Notify Operator; 'RUNID ERROR' or 'RUNID ABORT'.

4. Dump Registers to PRINT$

5. Check for ABORT$; collapse storage limits values in every ASA associated with this program.

6. EXIT to TERM.

Given a program with three activites, consider a normal termination request for activity 1, ABORT$ termination for activity 3, and that in turn would have forced activity 2 into error termination due to the replacement of its storage limits values. As each activity passed through TERM its activity count would have been decremented and would be at zero now. Since there are no more activities to terminate the element EXITFØ would then be called in to terminate or cleanup the program.

EXITFØ

The element EXITFØ receives control from the element TERM only when the activity count reaches zero. EXITFØ essentially cleans up the details resulting from the support of the activities just terminated. This program cleanup is accomplished by the following steps:

1. Assume RD$DT; for diagnostic messages by attaching to the user's PCT.

2. Disable Communications.

3. Write Dynamic Dumps.

4. Write 'PMD' Dumps.

5. Release I/D Bank.

6. Activate CSA.

Appraisal of Program Performance by CSA

The element CSA in the Coarse Scheduler has the responsibility of deciding whether or not the run is to be continued. As a result of analyzing certain performance flags, CSA will reach one of three possible conclusions:

1. The program terminated normally. CSA will inspect and honor the next run stream control statement.

2. The program terminated in error. Only conditional statements and requests for PMD will be honored. Any other control statement will result in the termination of the run.

3. The program terminated in abort mode. No control statements will be honored and run termination will result with no further reference to the run stream.

REMID

The element REMID is a multi purpose element providing services to the termination complex and others. Those steps taken on behalf of run termination are as follows:

1. Check for 'SO' Flag and remove Hold. The presence of an 'SO' flag indicates that some subsequent run is being held in the scheduling queue awaiting the termination of this run.

2. Make Log Entry, Run Termination Entry. This specific type of log entry triggers the accounting function to purge the run temporary log.

3. Release Run Tables:
   a. Chained Core Request
   b. Core Queue Entry
   c. Run Control
   d. Program Control Table

4. Activate CSN/CSU

Accounting

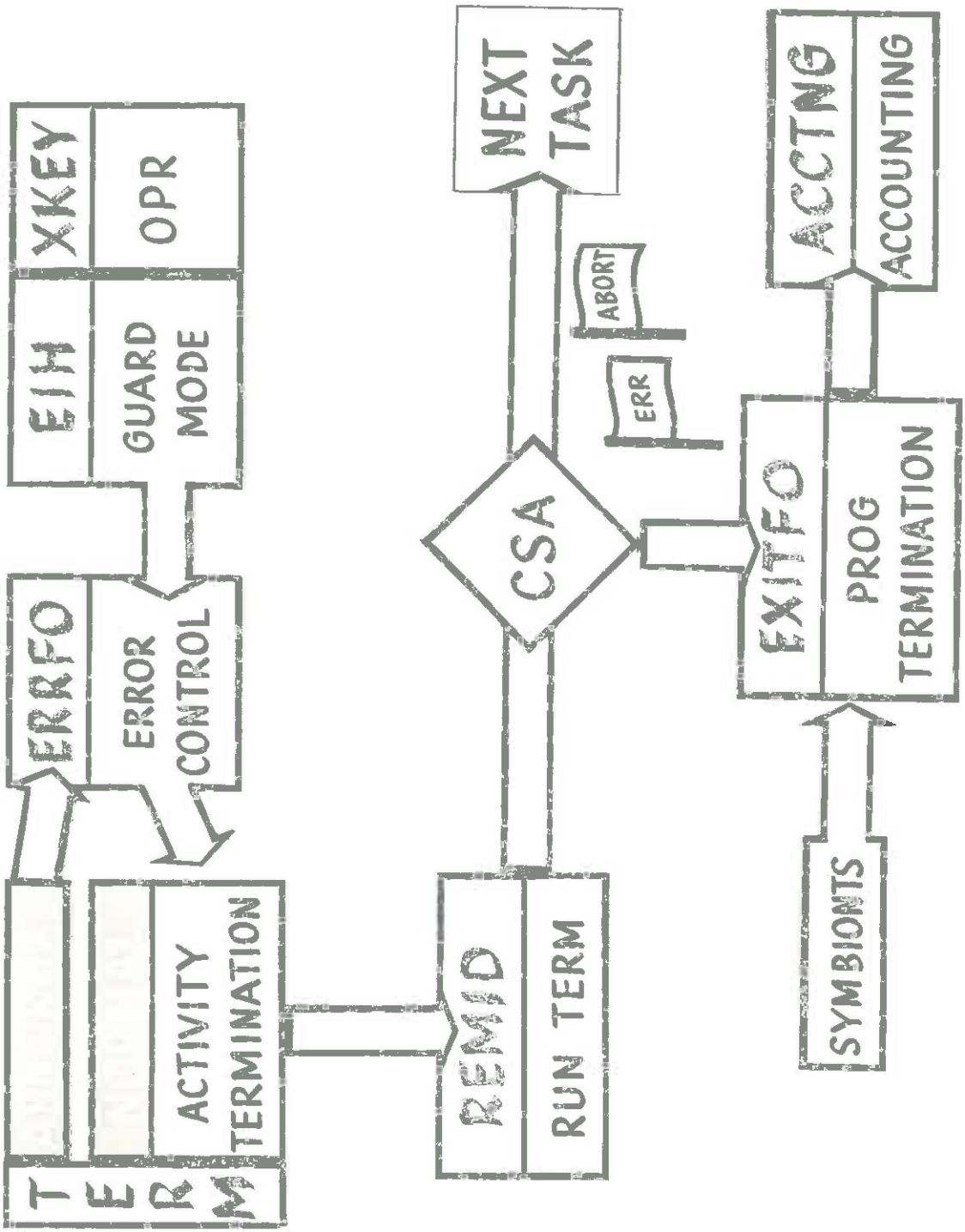
Throughout the execution of a run, the function LOGM is called on to remove log entires from core and place them in a run temporary log file. All entires for any given run remain there until the run termination entry is made. This entry calls in the ACTNG function to purge the run temporary file of all log entries made on behalf of that run. ACTNG chronologically removes each entry and writes it in the master log, updates the summary account file and makes final entries in the PRINT$ file.

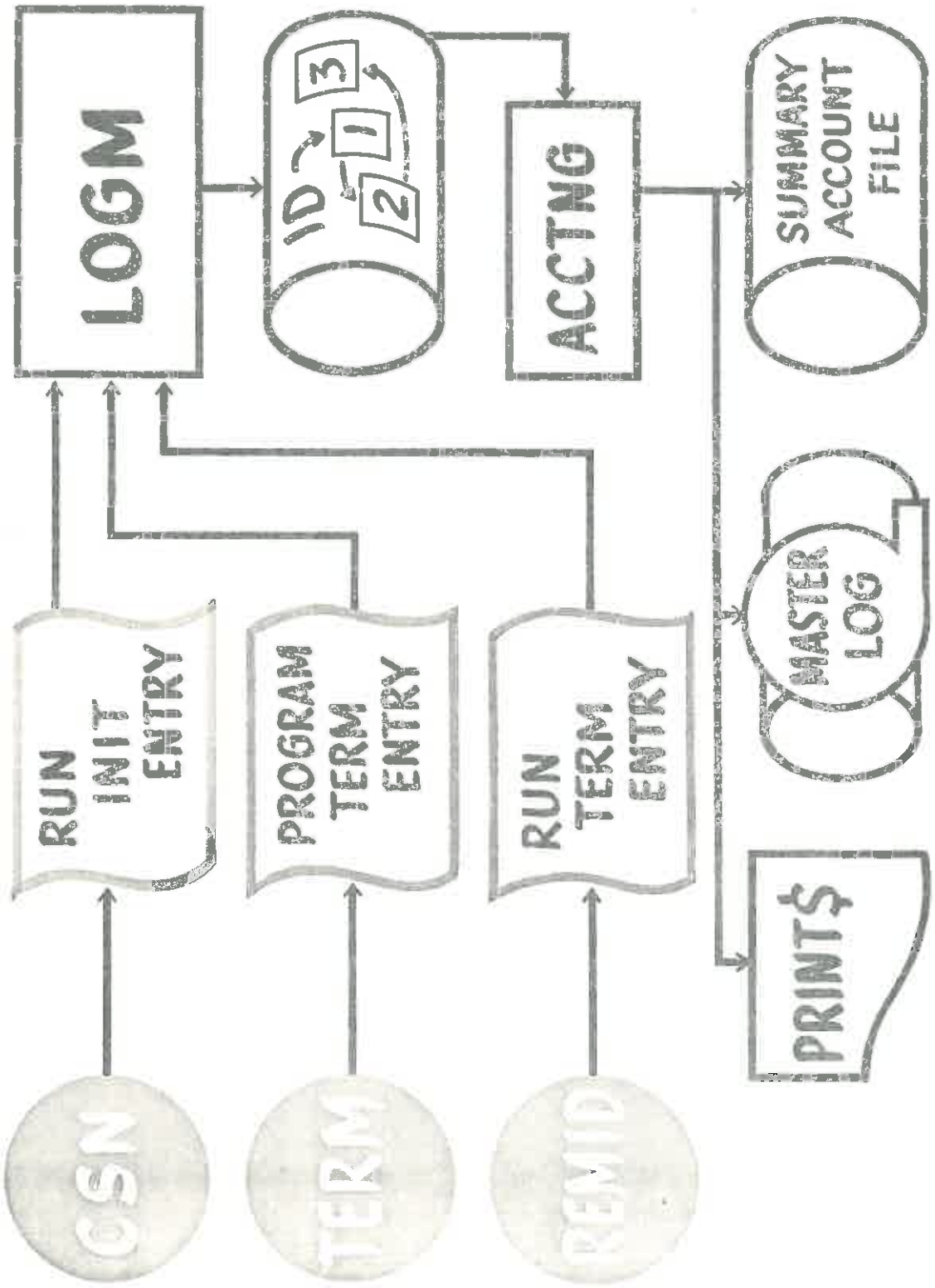
Removal of the RUNID Entry

As the Symbiont complex completes the processing of each output file generated by the run, it enters REMID to decrement the output file count which is carried in the RUNIDQ entry. When that output file count reaches zero, REMID will remove the RUNIDQ entry from the queue and that completes the termination of the run.

| ELEMENT NAME | RECIPIENT | CAUSE AND TEXT |
|---|---|---|
| EXITFØ | PRINT$ | I/O Error in Writing Out Dynamic Dumps:<br>'Error - Dynamic Dumps Not Closed' |
| EXITFØ | PRINT$ | I/O Error in Writing Core to DIAG$:<br>'I/O Error in Termination PMD Not Initialized' |
| EXITFØ | PRINT$ | Inability to Further Expand PCT:<br>'PCT/PROG Overflow' or 'Dynamic PCT Expansion for RT Prog Not Allowed' |
| CSA | PRINT$ | Run Placed in ABORT Mode:<br>'Remaining Control Statements Ingored' |
| XKEY | LOG | 'X' or 'E' Keyin From The Console<br>'Operator Killed Run' |
| XKEY | CONSOLE | RUNID Not Found in Run Control Table:<br>'Run Not Active' |
| ERRFØ | PRINT$ | All Remaining Activities in WAIT State:<br>'Prog Abort - DEACT/AWAIT Ambiguity' |

Figure 1.Termination Messages


XKEY
OPR
EIH
GUARD MODE
ERRFO
ERROR CONTROL
TERM
ACTIVITY TERMINATION
NEXT TASK
ABORT
ERR
CSA
RSMD
RUN TERM
ACCTNG
ACCOUNTING
EXITFO
PROG TERMINATION
SYMBIONTS


LOGM
ID
1
2
3
ACCTNG
SUMMARY ACCOUNT FILE
MASTER LOG
PRINT$
RUN INIT ENTRY
PROGRAM TERM ENTRY
RUN TERM ENTRY
OSN
TERM
REMID

# 7. RUN STREAM ANALYSIS

This seminar has covered the basic functions of EXEC 8, showing the interrelationships between elements and the use of tables in this environment. The various elements are related in that each may call on others to perform functions. A typical run stream presented to the executive can be defined as follows:

Typical Run Stream

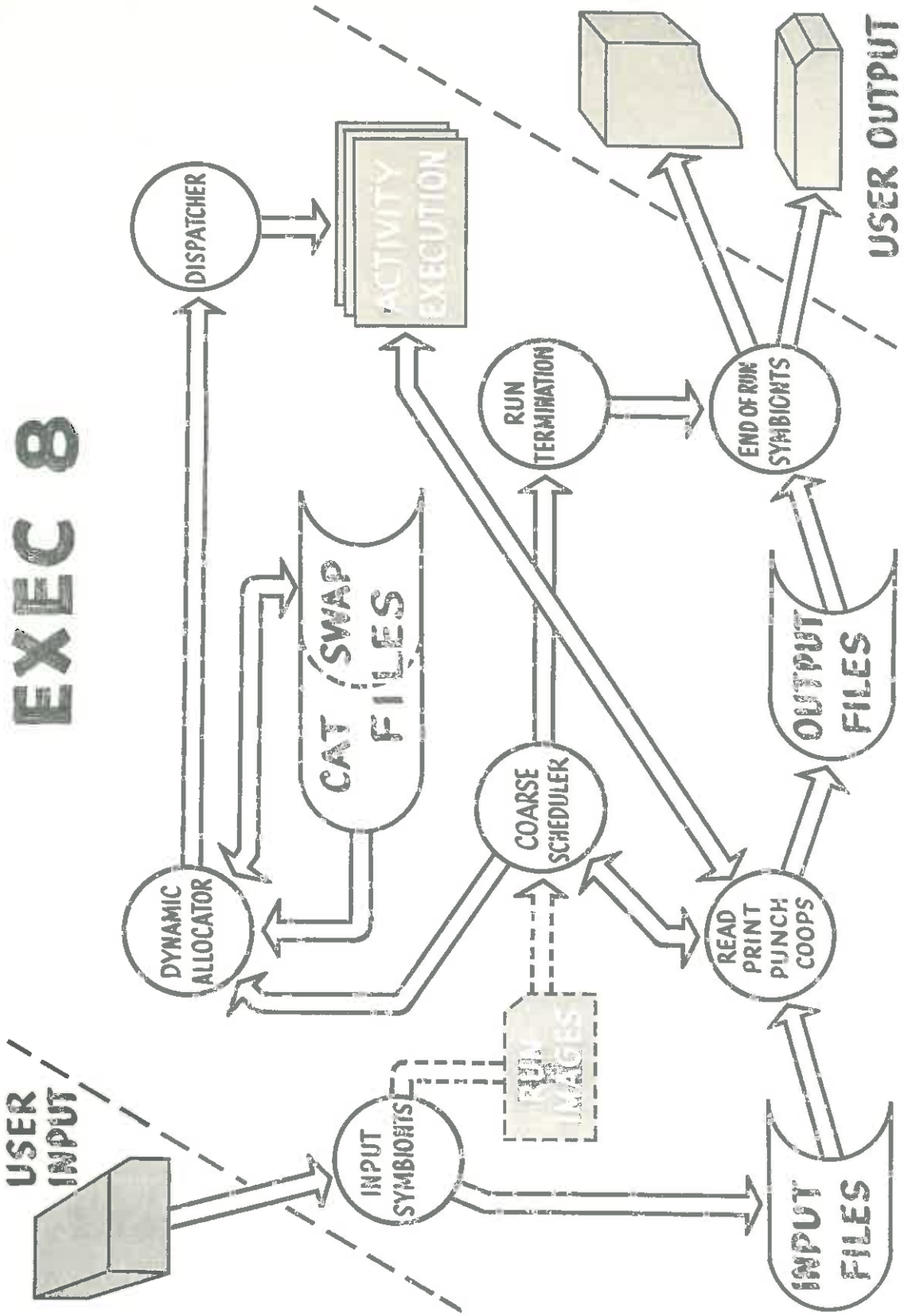
```
@Run One
@ASG,A FILEA
@Start FILEB
@XQT FILEA,ABC
@FIN
```

The Symbiont Complex (Input Symbiont) will recognize a run image and pass the image to the Coarse Scheduler for scheduling/activation. The Coarse Scheduler will obtain facilities through the Facility Inventory complex and perform a program file search for all tasks as they occur. The Dynamic Allocator is called to allocate central memory and load the program initially. The Dynamic Allocator activates this task and the Dispatcher places it into execution by granting time slices to this task. The program executes, and upon completion termination is effected, bringing the Coarse Scheduler back into action. This continues until the @FIN is encountered causing run termination, and output files are removed via the end of run symbionts. After the run is removed from central memory, the Coarse Scheduler is called in to analyze the Scheduling Queue for another entry to activate.

This analysis is meant only to relate these sections to each other; to understand these sections better, we suggest a review of the previous sessions.

This seminar conveys basic EXEC 8 information. UNIVAC has recommended this seminar a pre-requisite for further training and understanding of EXEC 8.

Requests for additional information or any questions concerning this seminar should be discussed with the local UNIVAC representative, who is aware of the information currently available and will be notified as additional training sessions are released.


EXEC 8
USER INPUT
USER OUTPUT
DISPATCHER
ACTIVITY EXECUTION
RUN TERMINATION
END OF RUN SYMBIONTS
CAT / SWAP FILES
OUTPUT FILES
COARSE SCHEDULER
DYNAMIC ALLOCATOR
READ PRINT PUNCH COOPS
INPUT SYMBIONTS
INPUT FILES

UNIVAC

SPERRY RAND