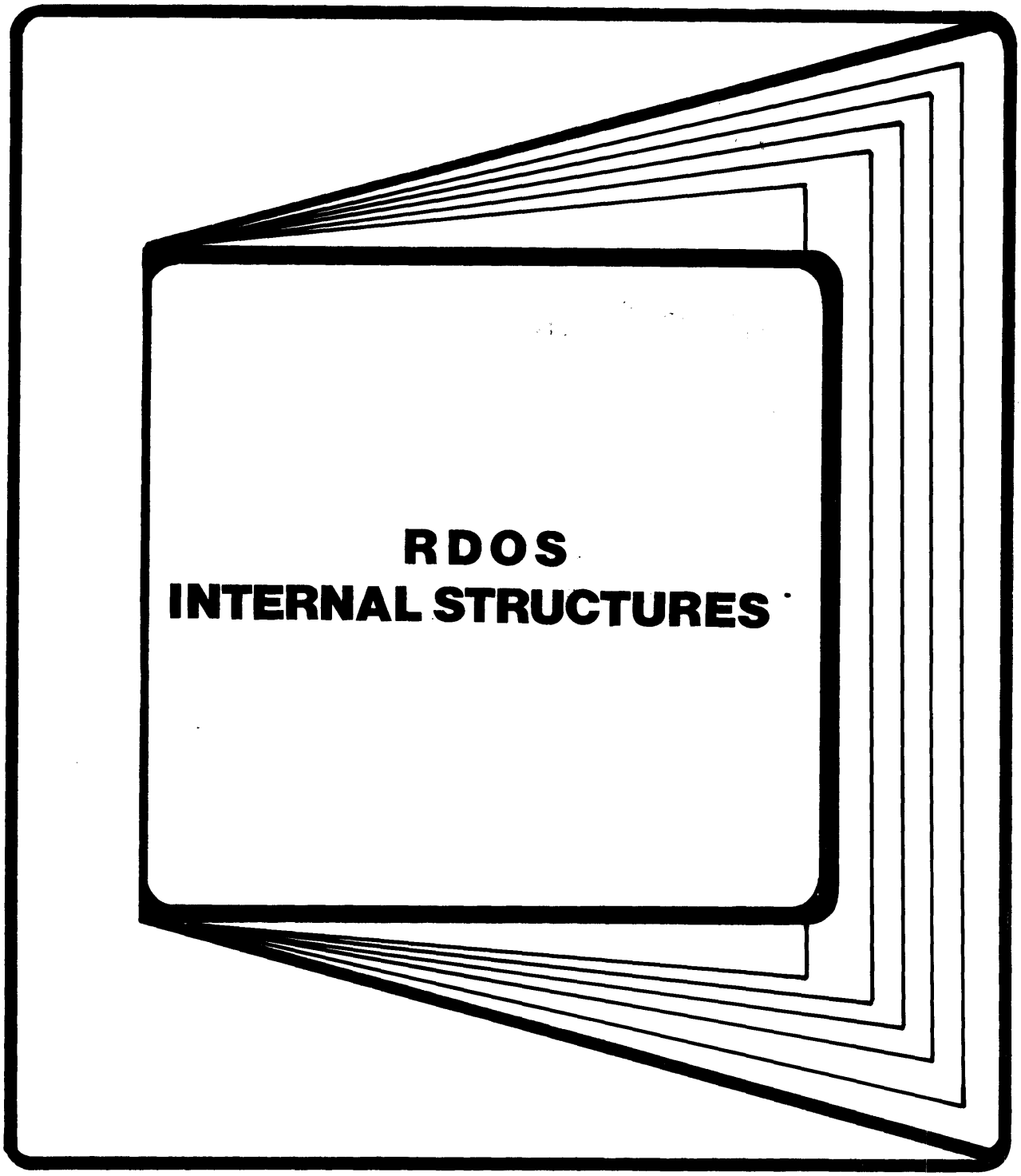
Data General



Student Handbook

Educational Services

019-000002-01

NOTICE

Data General Corporation (DGC) has prepared this manual for use by DGC personnel, licensees, and customers. The information contained herein is the property of DGC and shall not be reproduced in whole or in part without DGC's prior written approval.

Users are cautioned that DGC reserves the right to make changes without notice in the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including, but not limited to typographical, arithmetic, or listing errors.

The following are trademarks of Data General Corporation, Westboro, Massachusetts:

U.S. Registered Trademarks	Trademarks
ECLIPSE	INFOS
NOVA	DASHER

RDOS

INTERNAL STRUCTURES

019-000002-01

TABLE OF CONTENTS

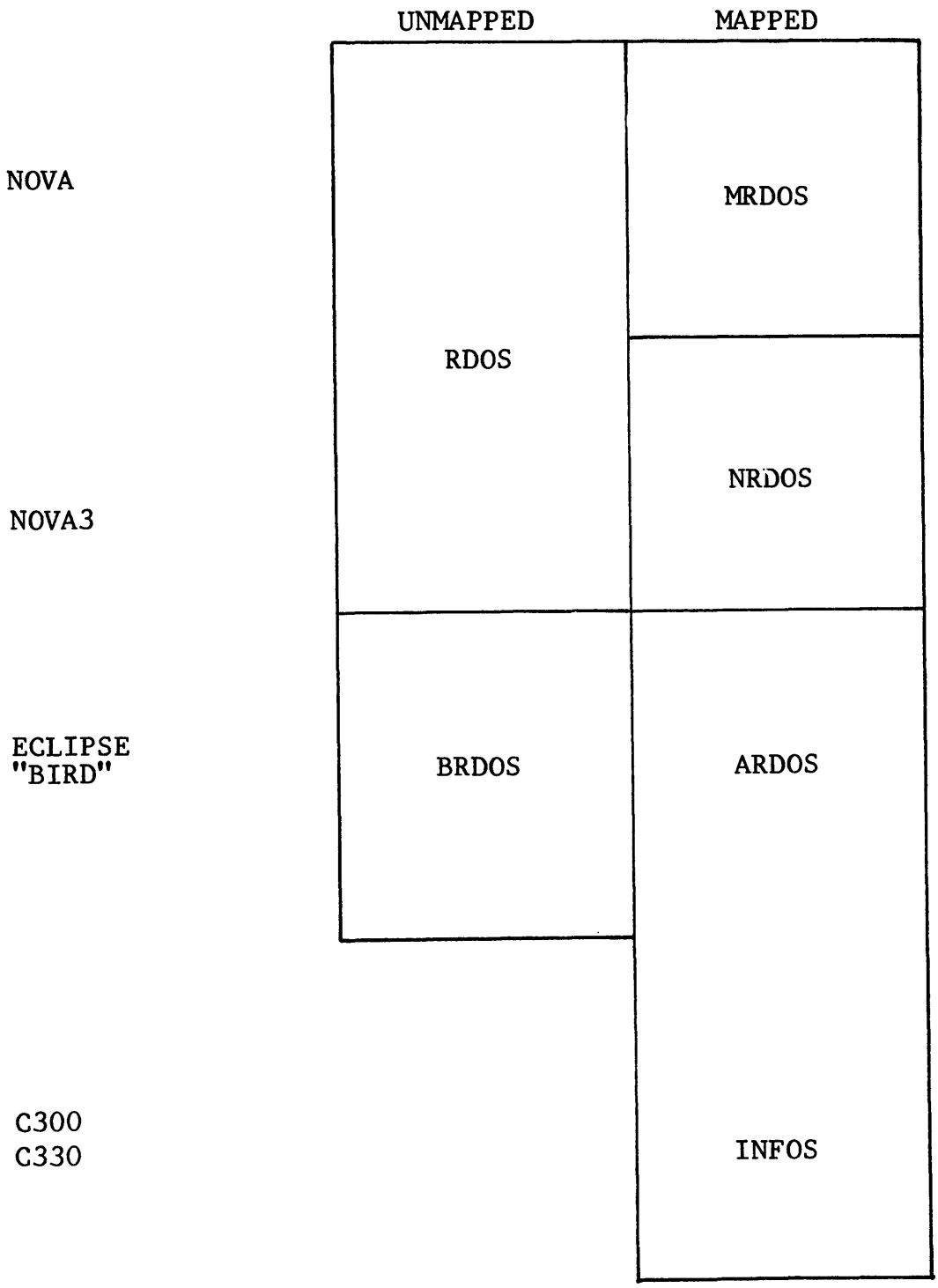
THIS BOOK

This course is an introduction to a large assembly language program called RDOS. We'll look closely at its bookkeeping tables to find out what RDOS does. To find out how RDOS works we'll outline some of the major code as well. This is but the first step toward the ability to modify system code. This course doesn't involve reading source listings, but rather prepares you to dig through the sources afterwards. In fact, the best way to fix the course information in your mind is to dive into RDOS source code immediately after the course.

Covering all the different machines that RDOS runs on posed a problem to both the people who wrote RDOS and those who wrote this book. The authors of RDOS solved this through a set of switch valves that control optionally assembled code. This allows most of the sources for all flavors of RDOS to be the same, but to produce different binary code under different switch settings.

In this book, we stress global facts that are applicable to all flavors and only differenciate CPUs where it severly impacts bookkeeping tables. The parameter files are the ultimate source for bookkeeping structures; the listings are the ultimate source for code. This manual is simply a guide to the internals of RDOS.

FLAVORS of RDOS



Section 1

SYSTEM OVERVIEW

DEFINITION

RDOS

IS AN

INTERRUPT DRIVEN

MULTI TASKING

SYSTEM OF

BOOKKEEPING TABLES

AND

ROUTINES

MANAGING HARDWARE

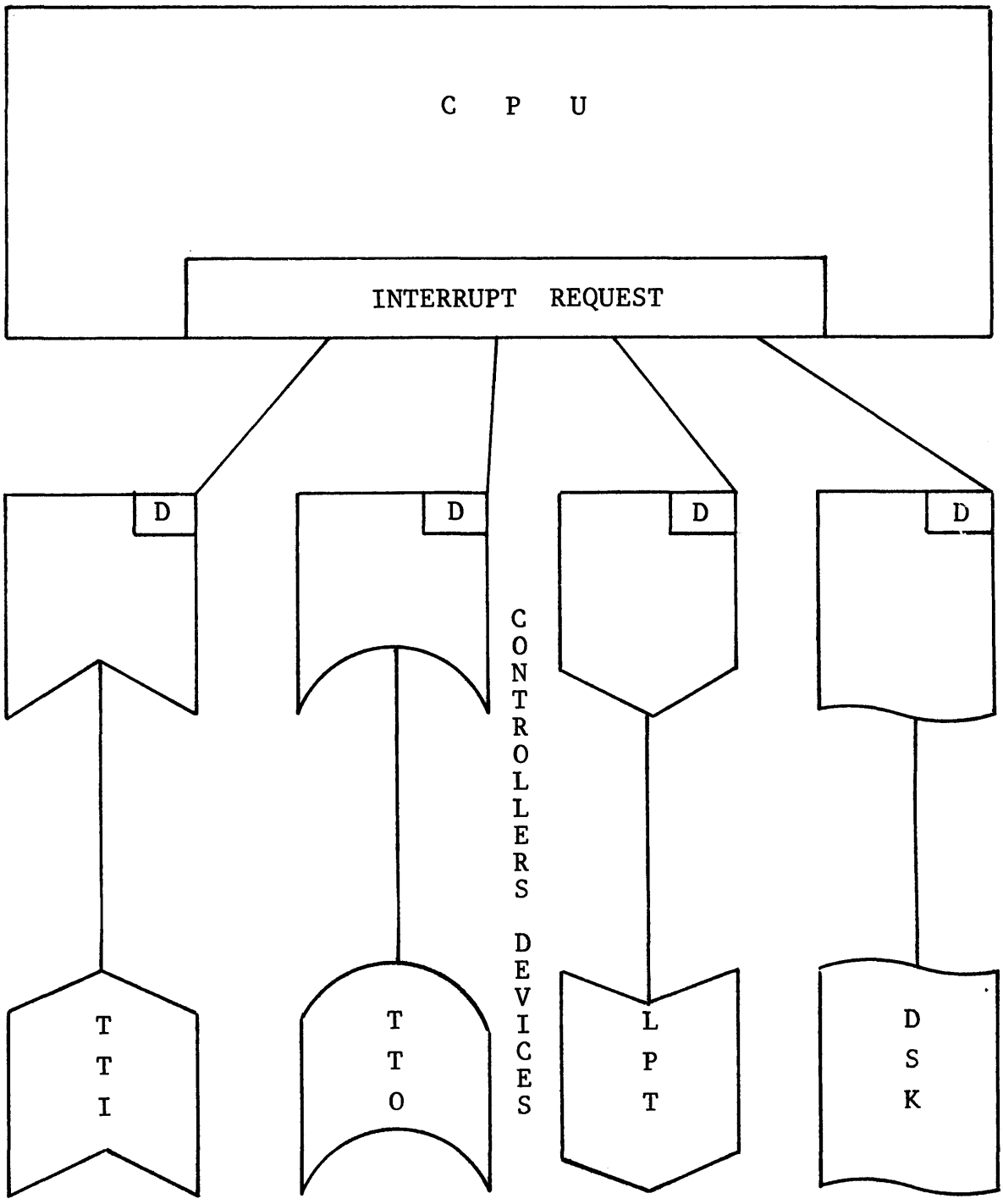
RESOURCES

ACCORDING TO

USER

SOFTWARE

INTERRUPT HARDWARE



INTERRUPT HARDWARE

CPU

TELLS CONTROLLER TO

TRANSFER DATA

TO/FROM DEVICE

CONTROLLER

(DESIGNED TO MATCH DEVICE SPEED)

TAKES RESPONSIBILITY

FOR TRANSFER.

LETS CPU WORK ON INDEPENDENTLY

CONTROLLER

INTERRUPTS CPU BY

SETTING "DONE" FLAG

AFTER DEVICE TRANSFER

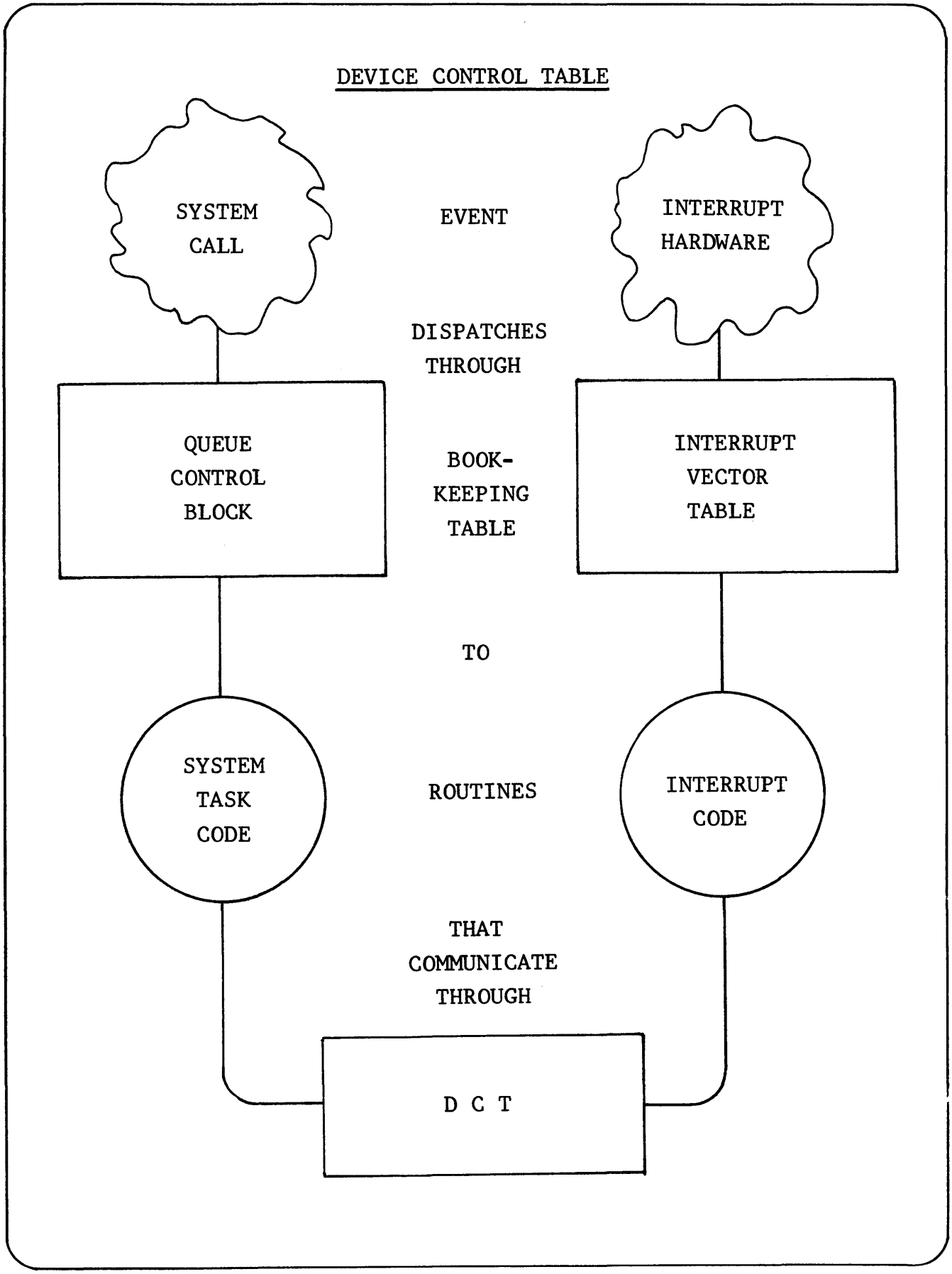
CPU

REACTS TO

TRANSFER COMPLETE

OR

ERROR STOP



INTERRUPT SOFTWARE

WHY 2 KINDS
OF ROUTINES
?

MINIMUM INTERRUPT CODE
= MINIMUM CONTROLLER
IDLE TIME
= MAXIMUM DEVICE SPEED

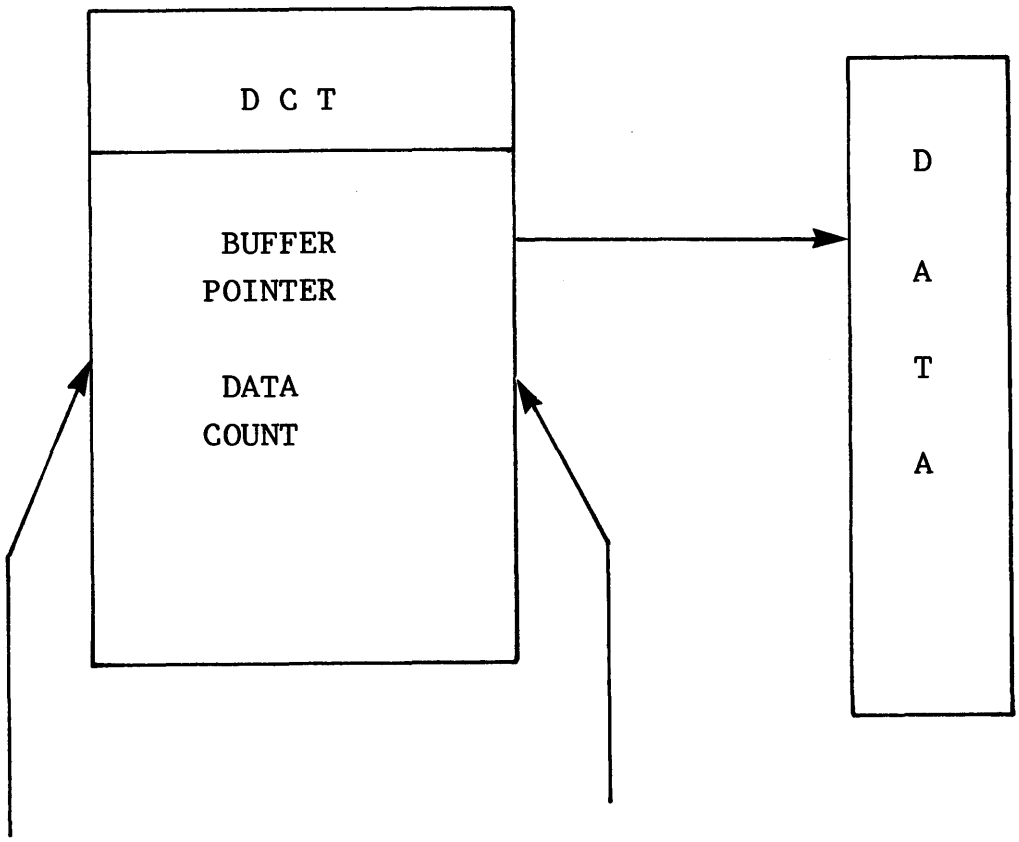
THEREFORE
MOVE AS MUCH WORK AS POSSIBLE
INTO
SYSTEM TASK ROUTINES
= NON - TIME - CRITICAL
= EXECUTES WHILE DEVICE
CONTROLLERS ARE BUSY

DEVICE CONTROL TABLE

INTERFACES

SYSTEM TASK CODE

AND INTERRUPT CODE



SYSTEM TASK
RUN BY SCHEDULER
SOFTWARE AND
SYSTEM CALLS

INTERRUPT
SERVICE ROUTINES
RUN BY INTERRUPT
HARDWARE

PHASES OF I/O PROCESSING

SYSTEM TASK (SET UP)

PREPARES BUFFER
SETS DCT BUFFER POINTER
SETS DCT DATA COUNT
ISSUES CONTROLLER COMMAND

INTERRUPT SERVICE ROUTINE

IF ERROR, TRY AGAIN
ELSE UPDATE DCT
  IF DONE, TELL TASK
  ELSE START NEXT

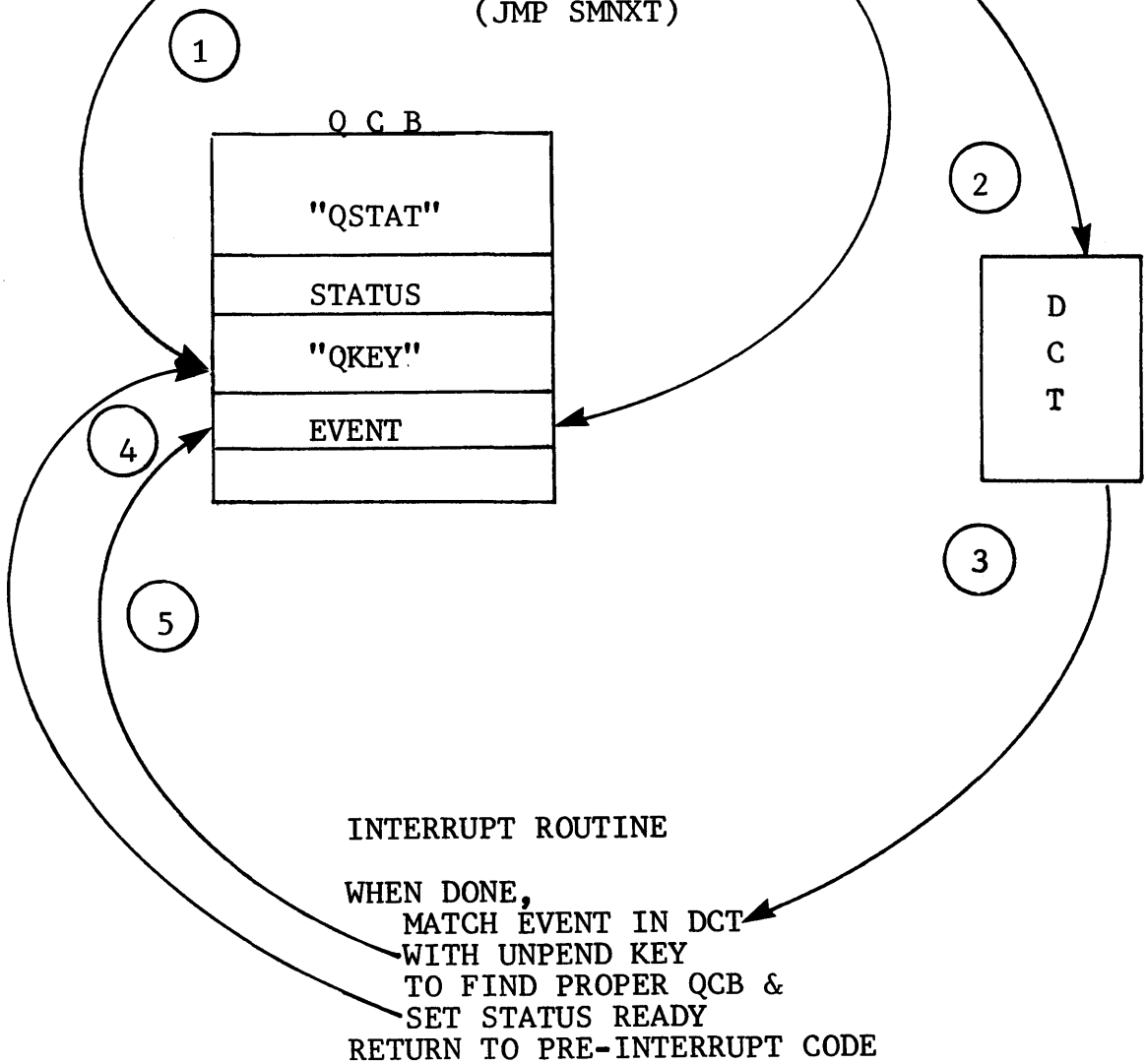
SYSTEM TASK (REACTION)

RESET DCT ENTRIES
PASS RESULTS TO USER
READY USER TASK
FREE UP RESOURCES ALLOCATED
  TO SYSTEM TASK (QCB,...)

PENDING A SYSTEM TASK

SYSTEM TASK (SET UP)

PREPARE FOR I/O EVENT
SET STATUS TO 'PENDED'
DESCRIBE UNPEND EVENT
START FIRST TRANSFER
PASS CONTROL TO SCHEDULER
(JMP SMNXT)



INTERRUPT ROUTINE

WHEN DONE,
MATCH EVENT IN DCT
WITH UNPEND KEY
TO FIND PROPER QCB &
SET STATUS READY
RETURN TO PRE-INTERRUPT CODE

UNPEND EVENTS

EACH EVENT FOR EACH DEVICE HAS
A UNIQUE KEY

EX. #1 A SERIES OF DATA
INTERRUPTS
FROM A PROGRAMMED I/O DEVICE
"BYTE DEVICE"

EX. #2 A SINGLE
INTERRUPT
FROM A DATA CHANNEL DEVICE
"BLOCK DEVICE"

EX. #3 SYSTEM OVERLAY
IS LOADED

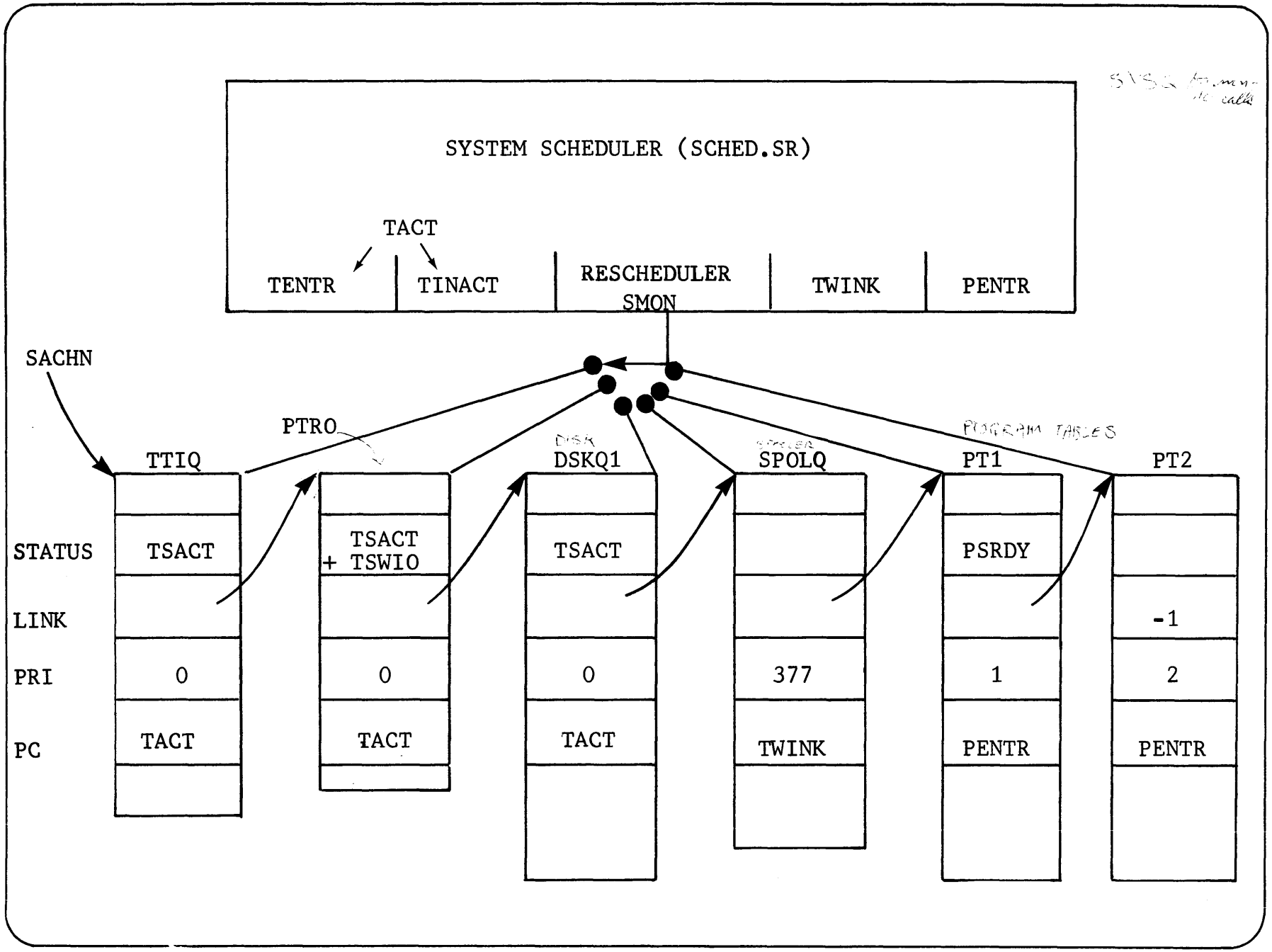
EX. #4 SYSTEM RESOURCE
FREED

notes

Section 2

QCB's

QUEUE CONTROL BLOCKS



SYSTEM TASK QUEUE CONTROL BLOCKS

Any processing path for a system call requires a minimum set of resources:

(1) System stack
(2) System task request cell
(3) System task queue control block

We will look at cells and stacks later on. Each I/O peripheral in the system has its own queue control block except the disks. There are two disk queues for all disks in the system (DSQ1 and DSQ2). There is also a system task queue control block for the spooler (SPOLQ). For processing system calls which are not I/O related there is a general processing task queue control block (SYSQ). This is also used for pre-processing of system calls to determine I/O device association. The two possible program tables PT1 and PT2 also form control blocks in the system task queue.

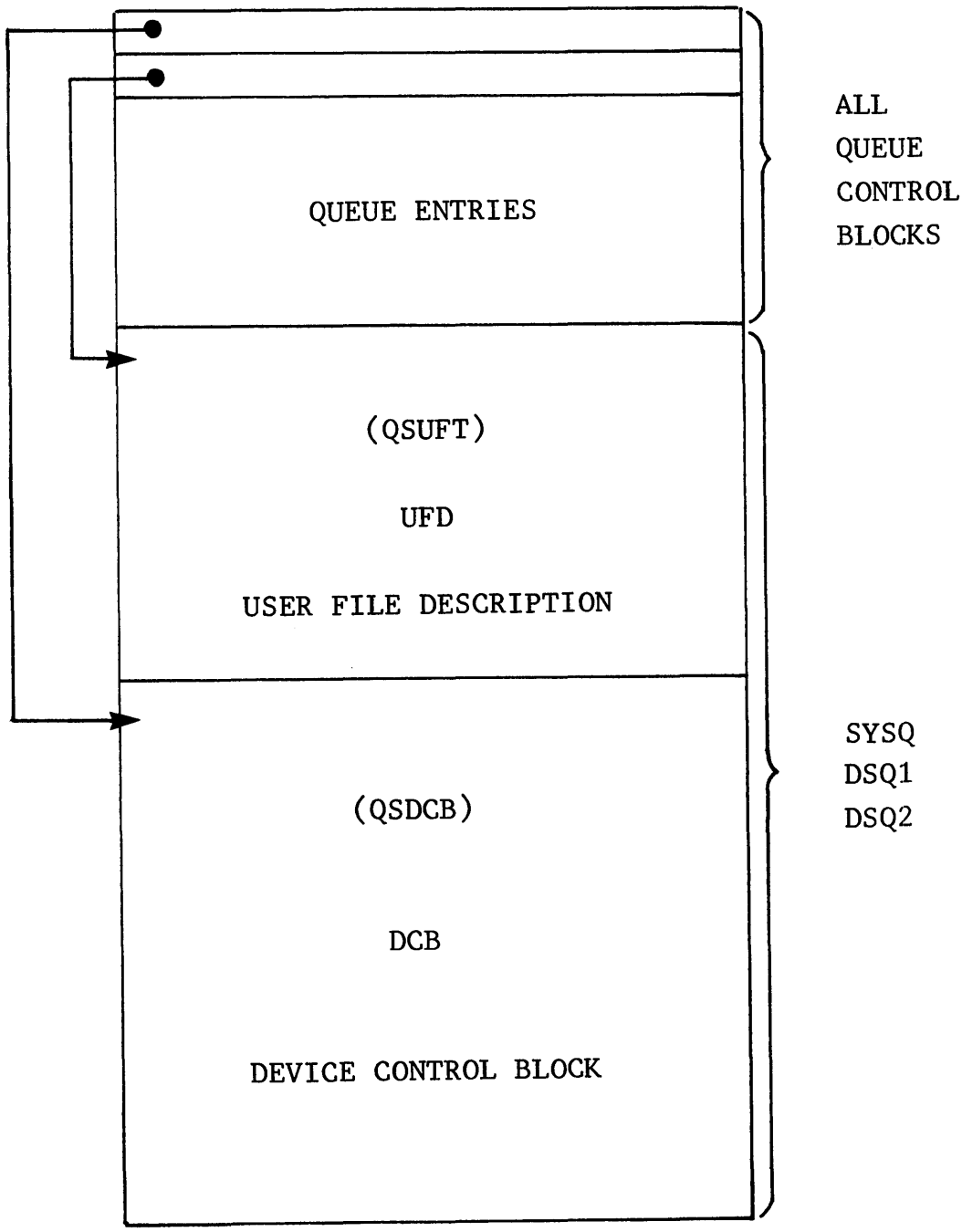
The system task queue control blocks are linked dynamically as system call processing is required. An essentially idle system (with respect to system work) would have only the spooler queue control block (SPOLQ) and the program table(s) in its active chain.

As system call processing is required by the user program, then the relevant system task queue control block is added to the head of the queue. The basic hierachy on the queue is summarized by:

(1) System Task Queue Control Blocks
(2) Spooler Control Block (SPOLQ)
(3) Foreground Program Table (PT1)
(4) Background Program Table (PT2)

Although the structure and content of these different types of QCBs vary dramatically, 4 words of each are identical in meaning. SMON (the system scheduler) uses only these 4 words, so to SMON, all QCBs look alike. The status and link words are (QSTAT & QALNK) PPRI, the priority word is a historical leftover from the days that ordered active QCBs by priority. It's still used to distinguish Foreground PT from background, but only by routines other than SMON. The last common word, PPC, holds the address of the state restore routine that SMON jumps to if the status is "ready". Since each QCB has its own PPC, this routine understands the unique structure of its QCB. SMON's main job is to link through generalized sections of QCBs to run a series of QCB-specific state restore routines.

SYSTEM TASK QUEUE CONTROL BLOCK



SYSTEM TASK QUEUE CONTROL BLOCK

QUEUE ENTRIES

0	POINTER TO QSDCB	QSDCP
1	POINTER TO QSUFT	QSUFP
2	TASK'S STACK POINTER	QSTKC
3	DCT ADDRESS IF ON QUEUE OR Ø	QDCT
4	PROGRAM STATUS	QSTAT = PSTAT
5	ACTIVE CHAIN LINK	QALNK = POLNK
6	PROGRAM PRIORITY	PPRI
7	PROGRAM PROCESSING PC	PPC
10	CURRENT CELL	QCURR
11	UNPEND KEY	QKEY
12	CELL LINK WORD	QLNK
13	CURRENT OVERLAY SEGMENT	QCRSG
14	CURRENT STACK FRAME POINTER	QSTK
15	NEXT QUEUE ADDRESS	QNXT
16	TIMEOUT CONSTANT	QTIME
17	ON QUEUE COUNT	QCNT

SYSTEM TASK QUEUE CONTROL BLOCK STRUCTURE

Word 0, QSDCP
Word 1, QSUFP

Every system call related to disk I/O needs a UFT. Some apply directly to the users UFT but others (i.e., CREATE, RENAME) have to be created. Only SYSQ, DSQ1 and DSQ2 have a UFT portion. If UFT does exist, words 1 and Ø point to the UFT and DCB portion respectively.

Word 2, QSTKC

Stack control table address of the system stack being used. Maintained so that stack can be released easily. (1BØ indicates this task uses a stack)

Word 3, QDCT

If control block of a peripheral device, address of the associated DCT. Used to match or for ENQUE (i.e. where to add cell to a queue). Ø for SYSQ, SPOLQ, DSQ1 and DSQ2.

Words 4 - 7

Are common with the program table offsets.

Word 4, QSTAT

Status word. Only status bit in common with program tables is 1BØ. (Always zero - ready to run). Other bit settings are:

1B13	TSAB	Abort current system (CTRL/A detected) call.
1B14	TSACT	Task is active. i.e. on the active chain.
1B15	TSWIO	Pended waiting for I/O.

Word 5, QALNK

Contains next element in the active chain (i.e. next active control block in queue).

Word 6, PPRI

Calling program priority. This is zero for any peripheral queue control block except SPOLQ, which is 377. (This is of historical interest only as SPOLQ is higher than PT1/PT2 on the queue.) For the program tables:

	PRIORITY
Foreground (PT1)	1
Background (PT2)	2

Word 7, PPC

Processing address when this queue control block gets control. All symbols are entry points in the system scheduler module (SYSTE).

(1) TACT-calls → TINACT if task has no current cell being processed but has a cell attached (i.e. competing for a stack).
→ TENTR if there is a current request being processed. If PENDed it will do a timeout check (once per second). Else it will restore state variables and continue when the task regains control.

(2) TWINK if an activity has begun which requires the spooler to be started up (only applies to SPOLQ). Uses similar procedures as TINACT once a number of special actions have occurred.

(3) SMNXT if there is nothing for the spooler to do. It is a dummy (only used by SPOLQ) to cause SPOLQ to be immediately passed over.

Word 10, QCURR — Address of current cell being processed. This is loaded into (CC) whenever this task is being run.

Word 11, QKEY — Key for UNPEND logic. If a system resource has been exhausted. Keys listed below

key	description
14+DCT ADDR	Byte device
DCB ADDR	Block device
Ø77777	Cell to be free
+1	Buffer to be free
Buffer Addr	Waiting for sys.OL
Ø	Not PENDED

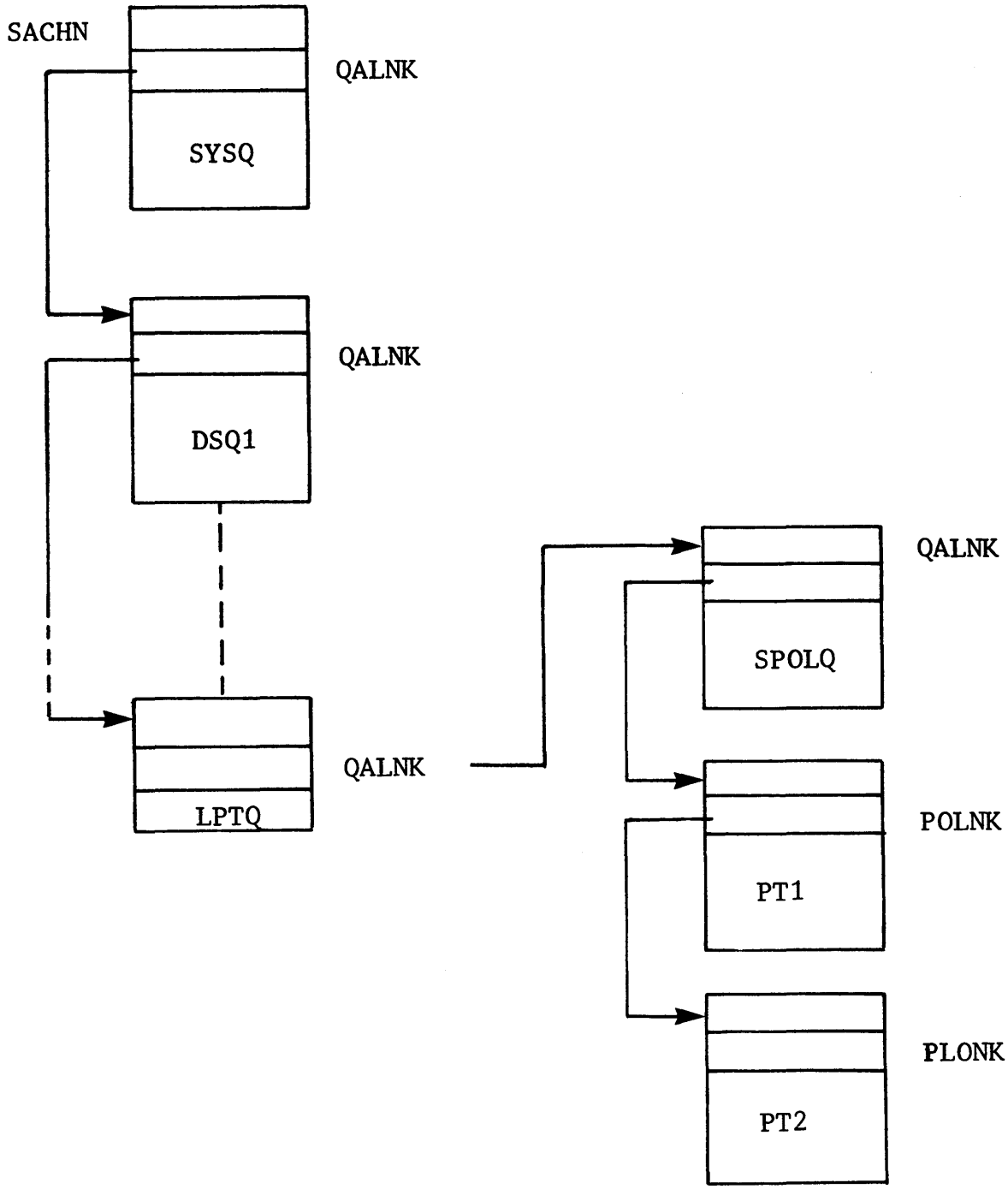
Word 12, QLNK — Link to the next cell in the task request chain. The requests are backed up to the task queue control block with respect to process priority. (i.e. new foreground program requests would be placed before any background program requests already on chain if the foreground has higher priority than background).

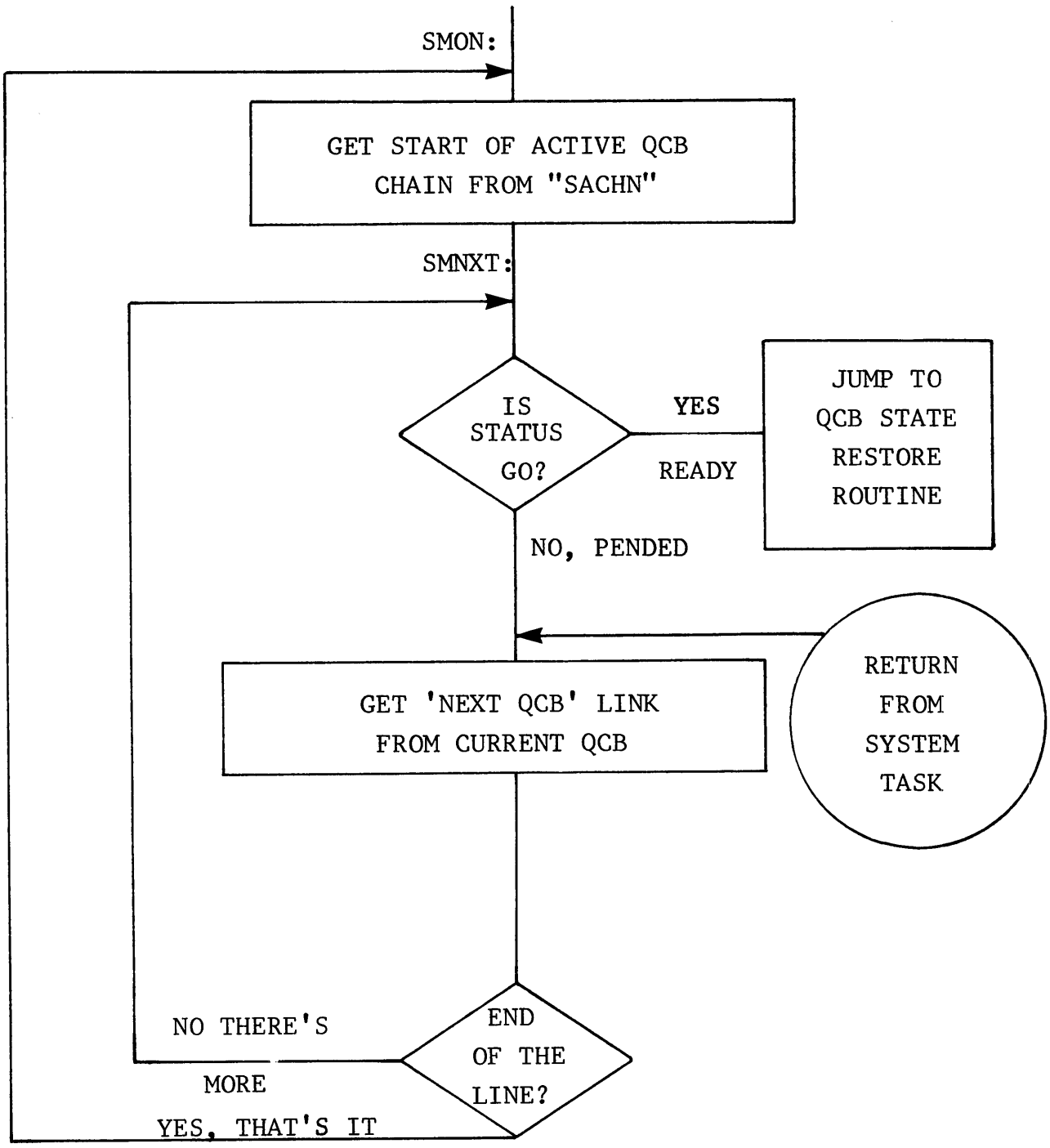
Word 13, QCRSG — Overlay table offset address of the currently executing segment. Placed in (CRSEG) when the task is started up.

Word 14, QSTK — Current stack frame pointer placed in (CSP) when the task is started up.

Word 15,	QNXT	Next queue address. This is an internal link of all known task queues (i.e. not necessarily active). This list is searched initially to associate a DCT with an I/O request.
Word 16,	QTIME	Timeout constant in seconds or -1 (wait forever).
Word 17,	QCNT	The number of requests on the queue control block. (i.e. the number of links on the chain pointed to by 12). If SPOLQ, then it is the processing address that TWINK will go to. In this case (QCURR) and (QLNK) are zero.
Word 2Ø,	QSUFT	Start address of UFT.
Word 42,	QSDCB	Start address of DCB portion of UFT.

SYSTEM TASK QUEUE



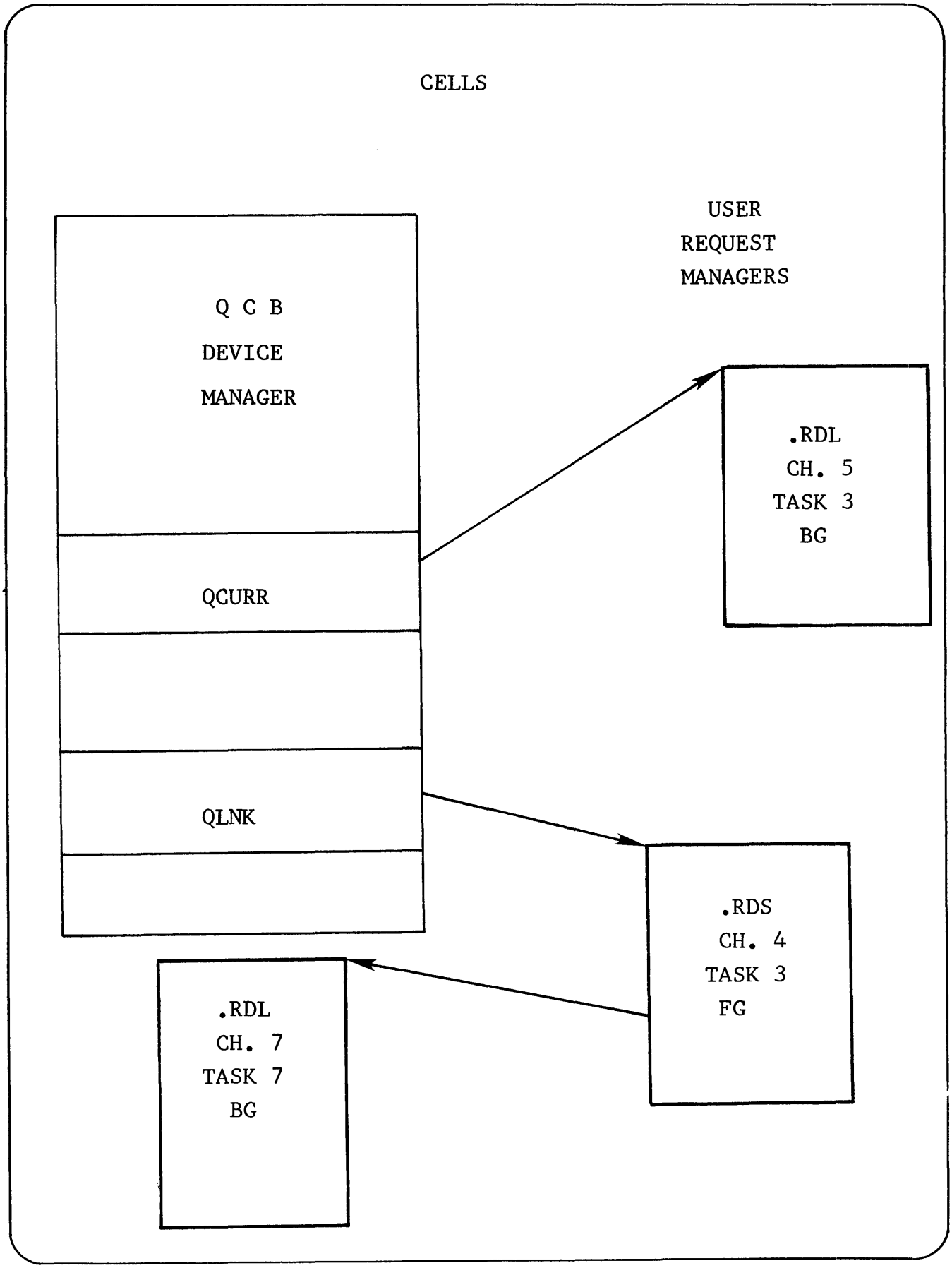


SYSTEM TASK
(NOT SCHEDULER)
HAS FREEDOM AND RESPONSIBILITY
OF SYSTEM TASK STATE SAVE
AND TIMING OF SYSTEM RESCHEDULE

notes

Section 3

CELLS AND SYSTEM CALL PROCESSING



1.10 SYSTEM TASK REQUEST CELLS

A system task is much like a user task. However, a system task is associated with a particular I/O device or with the spooler. Each device has a system queue entry in its driver module. Each system task uses this queue for its current information. A system call is allocated a system task request cell. The cell contains information on the system call being processed. The request cell is linked to the appropriate device queue.

System task request cells are also used by system routines for temporary storage. For instance, the spooler requires two cells for each actively spooling device. Also Direct I/O of Disk or MTA Data blocks require a cell to act as a fake buffer header. Because of the alternative uses of request cells, the length of the cell is longer than that required for it to act as a task request cell.

Since cells are a dynamic resource with a number of different uses it is possible for the system to exhaust its available pool. In this situation the user program tasks are held up until free cells become available.

Three cells are allocated for system use for each stack chosen at system generation time.

SYSTEM TASK REQUEST CELL

0	* USER TCB ADDRESS	CATCB
1	USER ACØ	CACØ
2	USER FILE POINTER ADDRESS TABLE	CAC1
3	UFT ADDRESS	CAC2
4	PROCESSING ENTRY POINT 1	CENT
5	*PROGRAM PRI. + FLAGS	CPROG
6	**PROCESSING ENTRY POINT 2	CENT2
7	UFT ADDRESS	CCHAN
10	TEMPORARY	CTEMP
11	*PROGRAM TABLE ADDRESS	CPTAD
12,13,14, 15,16	TEMPORARIES	CTMP2,CTMP3,..., CTMP6
17	CELL LINK	CLNK

* PRESERVED THROUGHOUT SYSTEM CALL

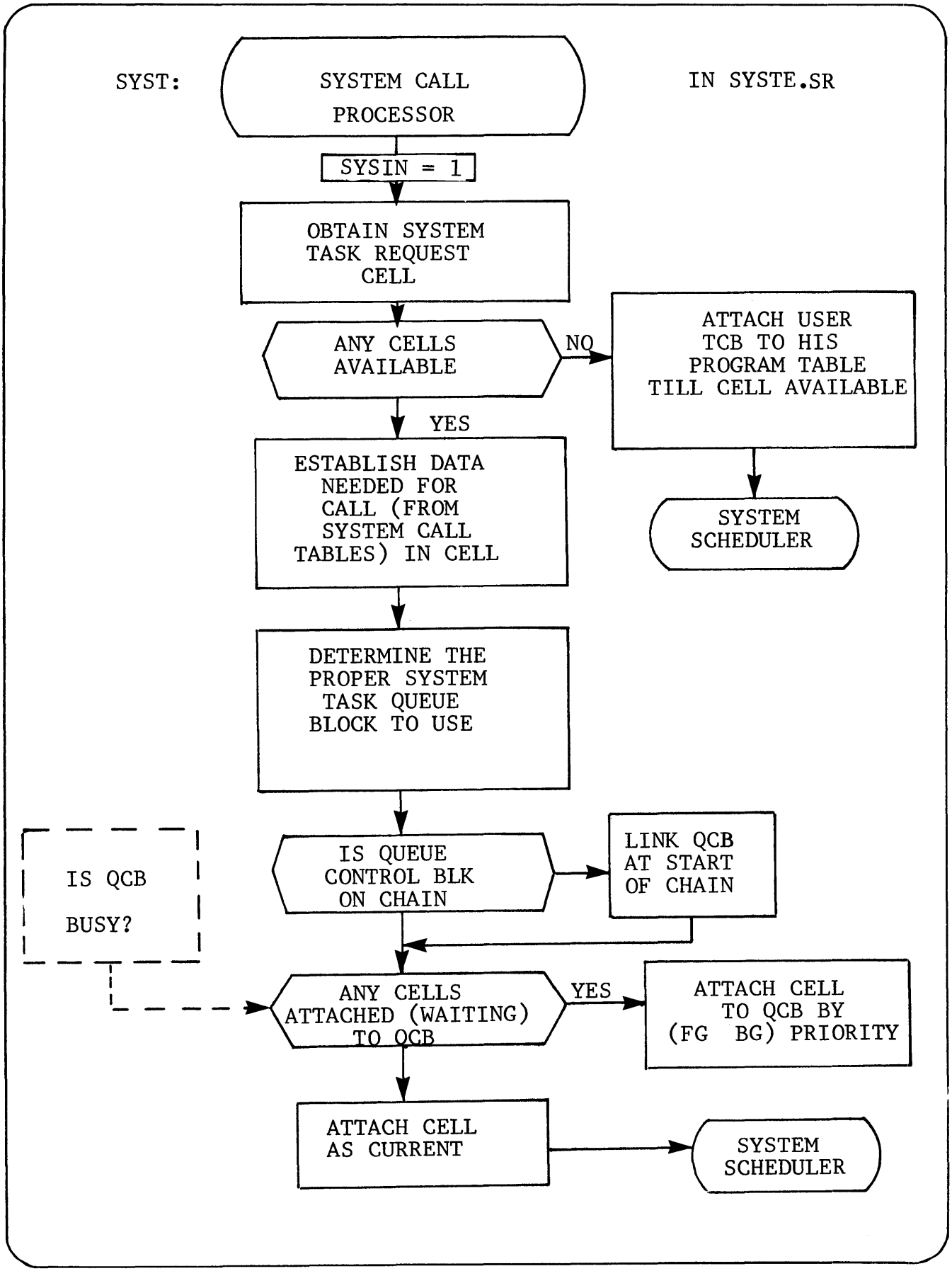
** PRESERVED THROUGH FIRST PROCESSING ENTRY CODE

SYSTEM TASK REQUEST CELL STRUCTURE

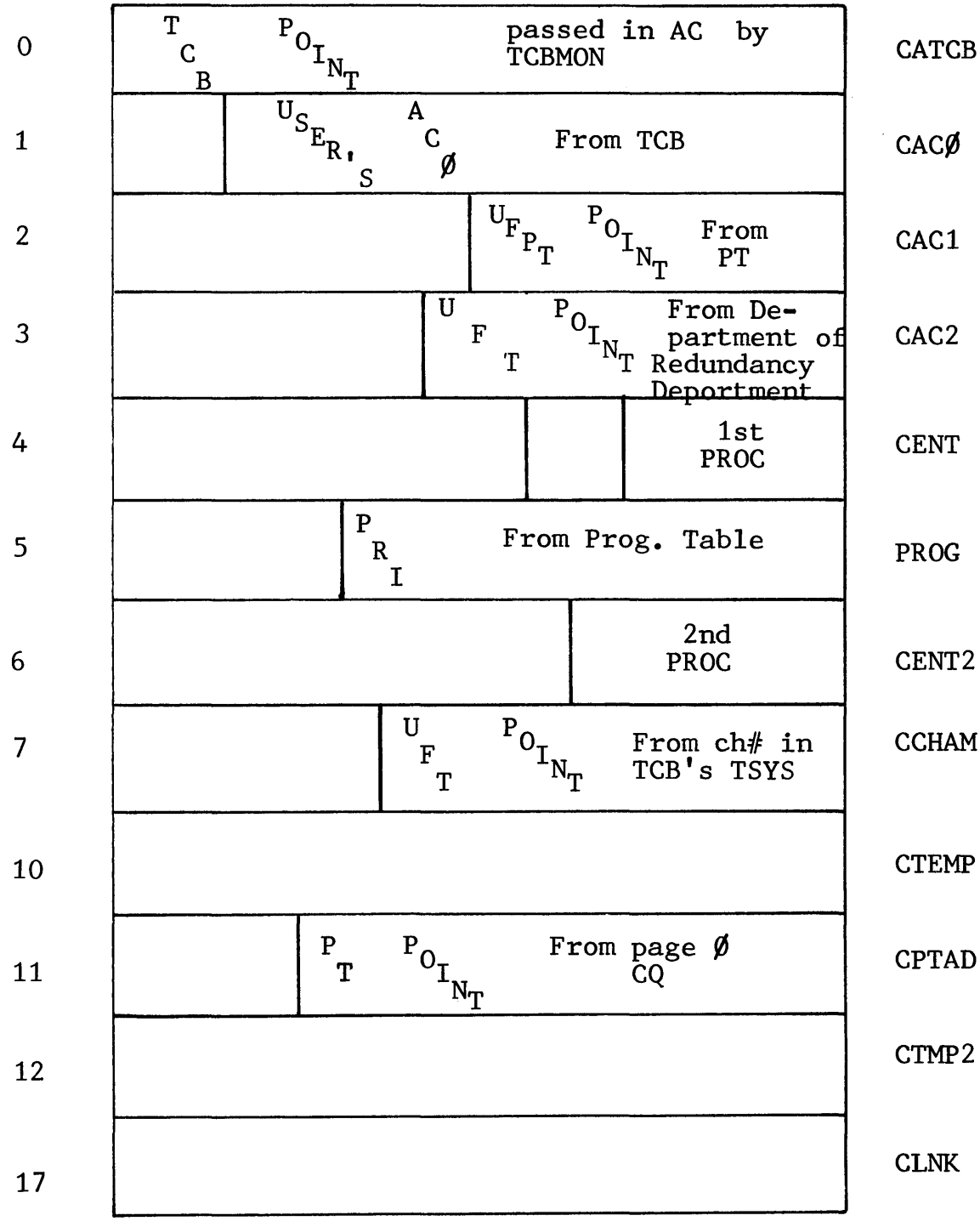
Word Ø,	CATCB	For an unmapped system this is the address of the user's TCB. For a mapped system this is the address of the users TCB + 76000. (i.e. map slot 31)
Word 1,	CACØ	User's ACØ, copied from user's TCB.
Word 2,	CAC1	Address of the UFPT pointer relating to the system call (Ø for non-I/O calls)
Word 3,	CAC2	Address of the UFT relating to the system call. (Ø for non-I/O calls)
Word 4,	CENT	First processing entry point. This code essentially determines DCT address for request queueing.
Word 5,	CPROG	Program priority: FG = 1, BG = 2 Flag: If 1BØ = 1, pass AC1 back to user.
Word 6,	CENT2	Second processing entry point. Code to carry out system call. For many calls only entry used (CENT = CENT2)
Word 7,	CCHAN	Address of the UFT relating to the system call.
Word 10,	CTEMP	Temporary location. Used by some system calls to control processing variations.
Word 11,	CPTAD	Address of program table (PT1/PT2)
Word 12, through Word 16,	CTMP2 CTMP6	Temporary locations. One use is to return an error code. CTMP2 = CERR
Word 17,	CLNK	Used to link cell to system task queue or the free chain.

NOTE: When a processing path is initially entered by RDOS, the AC's contain;

ACØ=CACØ,AC1 = CAC1,AC2 = CAC2,AC3 = CSP



SYSTEM TASK REQUEST CELL

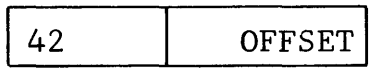


* PRESERVED THROUGHOUT SYSTEM CALL

** PRESERVED THROUGH FIRST PROCESSING ENTRY CODE

SYSTEM CALL CODES

Short Call format



Long Call format

Short Call

1 ENTRY, SYSQ, NO CHANNEL

LEFT BYTE = 42 (21XXX)

NON I/O CALLS = "SCALL"

Long Call - 3 types

Type 1 — 1 ENTRY, SYSQ

OFFSET IN CONTROL TABLE = 1

Type 2 — 2 ENTRIES, SYSQ

OFFSET IN CONTROL TABLE = ENTRY ADDRESS

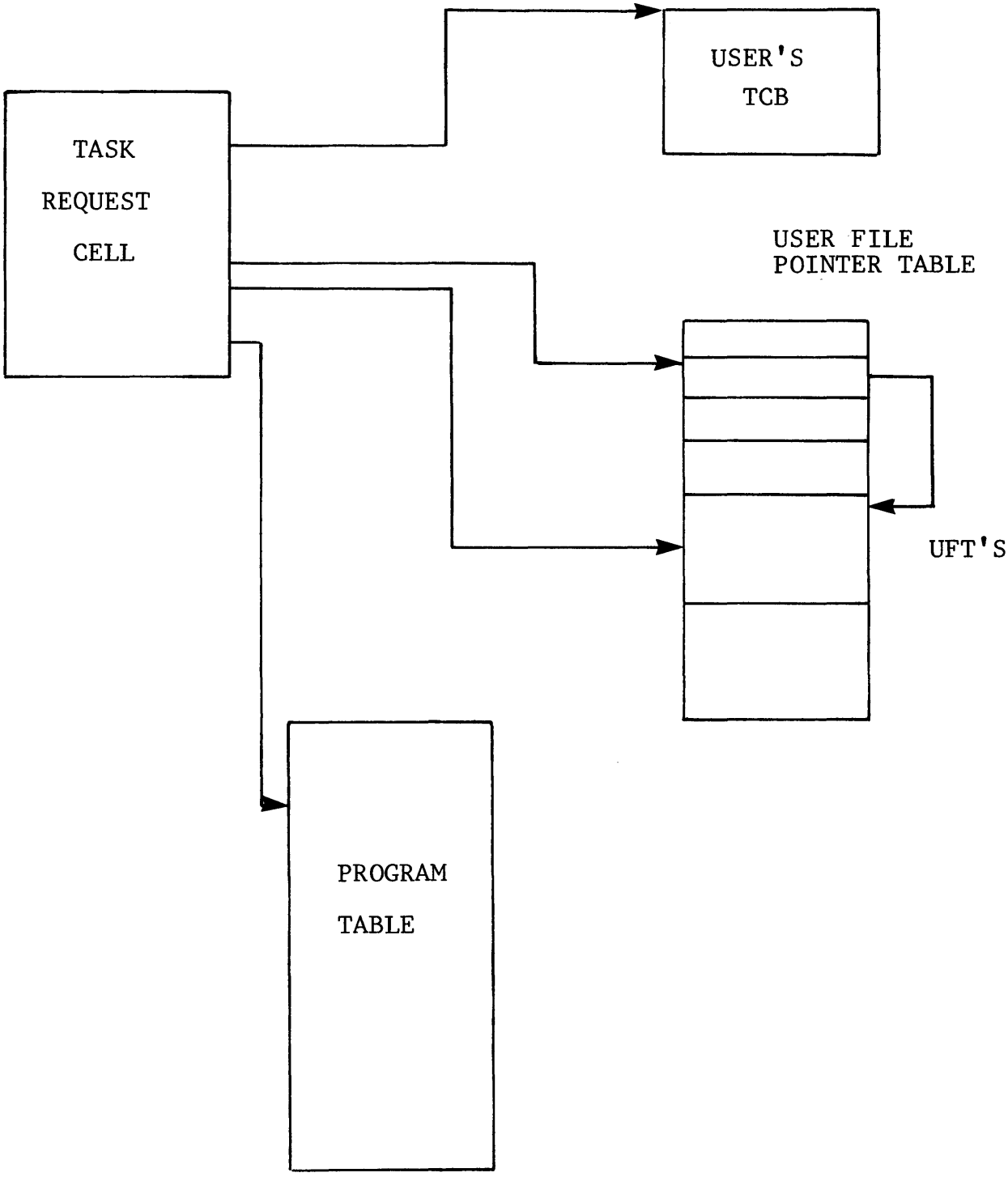
Type 3 — 2 ENTRIES, DEVICE SPECIFIC QCB

OFFSET IN CONTROL TABLE = 1BØ + ADDRESS

JUMP DIRECTLY TO ADDRESS

ROUTINE FINDS ITS OWN QCB FOR CENT2 PROCESSING

SYSTEM TASK REQUEST CELL



```
!0024 BSYST
01                ; SCALL TABLE
02 01114'077777 SCTBL:  RTCF            ; 0- GET FREQ RTC
03 01115'077777         UCLI            ; 1- SET USER CLOCK
04 01116'077777         UCLR            ; 2- REMOVE USER CLOCK
05 01117'077777         GTIME           ; 3- GET TOD
06 01120'077777         STIME           ; 4- SET TOD
07 01121'077777         SDAY            ; 5- SET DAY
08 01122'077777         GDAY            ; 6- GET DAY
09 01123'077777         IDEF            ; 7- DEV INTERRUPT DEFINE
10 01124'077777         IRMV            ; 10-INTERRUPT REMOVE
11 01125'077777         SPKIL           ; 11- KILL A SPOOLING OUTPUT
12 01126'077777         SPDIS           ; 12-DISABLE SPOLLING TO A DEV
13 01127'077777         SPEBL           ; 13- ENABLE SPOOLING TO A DEV
14 01130'077777         USRLST          ; 14- RSTAT, STATUS OF
15                                      ; RESOLUTION ENTRY
16 01131'077777         CPART           ; 15- CREATE PARTITION
17 01132'077777         CDIR            ; 16- CREATE DIRECTORY
18 01133'077777         LINK            ; 17- CREATE DIRECTORY LINK ENTRY
19 01134'077777         EQUIV           ; 20- REASSIGN LOGICAL DIRECTORY
20                                      ; SPECIF
21 01135'077777         GDIRS           ; 21- GET DIRECTORY SPECIF
22 01136'000445         SYSI            ; 22- SUS COMPATIBILITY
23 01137'177777         -1              ; 23- RTOS .WCHAR CALL
24 01140'077777         ICMN            ; 24- DEF COMMON
25 01141'077777         WRCMN           ; 25- WRITE TO COMMON
26 01142'077777         RDCMN           ; 26- READ FROM COMMON
27 01143'077777         ODIS            ; 27- DISABLE PROG CONT A
28 01144'077777         OEBL            ; 30- ENABLE CONT A
29 01145'077777         DEBL            ; 31- ENABLE MAPPED DEV
30 01146'077777         DDIS            ; 32- DISABLE MAPPED DEV
31 01147'077777         RDOPR           ; 33- READ OP MSG
32 01150'077777         WROPR           ; 34- WRITE OP MSG
33       000001         .IFE MSW
34 01151'000445'        SYSI            ; NOOP IF NOT MAPPED
35                      .ENDC
36       000000         .IFN MSW
37                      STMAP           ; 35-SET MAP FOR USER DCH
38                      .ENDC
39 01152'077777         GCIN            ; 36- GET CONSOLE INPUT DEV
40 01153'077777         GCOUT           ; 37- GET CONSOLE OUTPUT DEV
41 01154'077777         USRST           ; 40- STATUS OF FILE
42 01155'077777  ---->  ECLR            ; 41- CLEAR
43 01156'077777         TCRET           ; 42- TRANSPARENT CREATE
44 01157'077777         TCRND           ; 43- TRANSPARENT CRAND
45 01160'077777         TCCON           ; 44- TRANSPARENT CCONT
46 01161'077777         FGND            ; 45- SEE IF FG
47       000001         .IFE MSW
48 01162'177777         -1              ; 46- ILLEGAL IN UNMAPPED
49 01163'177777         -1              ; 47- ILLEGAL IN UNMAPPED
50                      .ENDC
51       000000         .IFN MSW
52                      GMEM            ; 46- GET MEM PARTITION
53                      SMEM            ; 47- SET MEM PARTITON
54                      .ENDC
55 01164'077777         HBOOT           ; 50- INVOKE HIPBOOT
56 01165'077777         GMDIR           ; 51- GET MASTER DIR. SPECIFIER
57 01166'077777         GCHN            ; 52- GET A FREE CHANNEL
58 01167'077777         UNLNK           ; 53- DESTROY A LINK ENTRY
59       000001         .IFE MSW
60 01170'177777         -1              ; 54-
```

```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
 0025 BSYST
01 01171'177777          -1                  ; WRITE EBL- .SCALL 55
02                       .ENDC
03       000000          .IFN MSW
04                       WRPR                ; 54- WRITE PROTECT
05                       WREBL               ; 55- WRITE ENABLE
06                       .ENDC
07 01172'077777          GSYS                ; 56- GET SYSTEM NAME
08 01173'077777          OVRP                ; 57- REPLACE OVERLAY
09 01174'077777          ABORT               ; 60- ABORT A TCB CALL
10 01175'077777          GMCA                ; 61- WHAT MCA AM I ?
11 01176'077777          SECI                ; 62- REQUEST TO RESCHEDULE
12                                           ; EVERY SECOND
13       000000          .IFN IOSW
14                ;**************************************************************
15                       HSTRU               ; IOCS - 63 - RUN HISTOGRAM
16                       HSTST               ; IOCS - 64 - STOP HISTOGRAM
17                ;**************************************************************
18                       .ENDC
19       000001          .IFE IOSW
20 01177'177777          -1
21 01200'177777          -1
22                       .ENDC
23 01201'077777          RDSW                ; 65- READ SWITCHES
24       000001          .IFE MSW
25 01202'177777          -1                  ; VMEM
26 01203'177777          -1                  ; MAP DEF
27                       .ENDC
28       000000          .IFN MSW
29                       VMEM                ; 66- VMEM
30                       MAPDF               ; 67- VIRTUAL DATA MAP DEF
31                       .ENDC
32 01204'077777          TUOFF               ; 70 - TURN TUNING OFF
33 01205'077777          TUON                ; 71 - TURN TUNING ON
34 01206'077777          INTAD               ; 72- WAIT FOR CONT A OR CONT C
35       000000          .IFN IOSW
36                       LBINI               ; 73- IOCS LABELED MAGTAPE INIT
37                       .ENDC
38       000001          .IFE IOSW
39 01207'177777          -1                  ; PLACE HOLDER FOR NON-IOCS
40                       .ENDC
41 01210'077777          CONNT               ; 74- CREATE CONTIGUOUS NOT DATA INIT
42
43       000074 SCTLN=   .-SCTBL-1
44
45
46
47                       .END

**00000 TOTAL ERRORS, 00000 PASS 1 ERRORS
```

```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
!0020 HSYST
01
02                ; MONITOR CALL CONTROL TABLE
03
04 00746'177777 MCCT:    @FILDQ            ; CREATE
05 00747'100746'          @FILDQ            ; DELETE
06 00750'100747'          @FILDQ            ; RENAME
07 00751'100750'          @FILDQ            ; MEMORY AVAILABLE
08 00752'177777           @BRKP1            ; BREAK
09 00753'077777           SDNCK             ; DEVICE RELEASE
10 00754'000753'          SDNCK             ; CHANGE DEFAULT DIRECTORY
11 00755'177777           @QCHK1            ; EXECUTE
12 00756'000754'          SDNCK             ; DEVICE INIT
13 00757'100755'          @QCHK1            ; RETURN
14 00760'000001           1                 ; I/O RESET
15 00761'177777           @BLKCA            ; BLOCK READ
16 00762'177777           @BLKCB            ; BLOCK WRITE
17 00763'100757'          @QCHK1            ; ERROR RTN
18 00764'100751'          @FILDQ            ; CREATE RANDOM
19 00765'177777           @TTYG             ; TTY GET CHARACTER
20 00766'177777           @TTYP             ; TTY PUT CHARACTER
21 00767'177777           @TIMEQ            ; DELAY
22 00770'100764'          @FILDQ            ; MEMORY INCREMENT
23 00771'077777           SOPNCK            ; OPEN FOR READING
24 00772'000771'          SOPNCK            ; INIT OVLYS
25 00773'000772'          SOPNCK            ; OPEN FOR APPENDING
26 00774'000001           1                 ; CHANGE ATTRIBUTES
27 00775'000001           1                 ; GET ATTRIBUTES
28 00776'000773'          SOPNCK            ; OPEN FILE
29 00777'077777           CLCHK             ; CLOSE FILE
30 01000'177777           @RDCK0            ; READ SEQUENTIAL
31 01001'177777           @RDCK1            ; READ LINE
32 01002'177777           @RDCK2            ; READ RANDOM
33 01003'177777           @WRCK             ; WRITE SEQUENTIAL
34 01004'101003'          @WRCK             ; WRITE LINE
35 01005'101004'          @WRCK             ; WRITE RANDOM
36        000001          .IFE MSW
37 01006'100770'          @FILDQ            ; OVERLAY REQUEST CHECK
38                        .ENDC
39        000000          .IFN MSW
40                        @LDOV             ; OVLY LOAD CHK FOR VIRTUAL NODE
41                        .ENDC
42 01007'101006'          @FILDQ            ; CREATE CONT
43 01010'000000           0                 ; NULL ENTRY FOR SCALL
44 01011'000001           1                 ; EXECF
45        000001          .IFE     IOSW
46 01012'177777           -1
47 01013'177777           -1
48                        .ENDC
49        000000          .IFN     IOSW
50                ;*****************************************************
51                PIOCS:  0                 ; IOCS - REGULAR PROCESS CALL
52                        1                 ; IOCS - OPEN CALL DIRECT PATH
53                ;*****************************************************
54                        .ENDC
55 01014'177777           @MTCHK            ; MTDIO
56 01015'000001           1                 ; SET FILE POSITION
57 01016'000001           1                 ; GET FILE POSITION
58 01017'000776'          SOPNCK            ; EOPEN
59 01020'001017'          SOPNCK            ; TOPN
60 01021'000001           1                 ; CHANGE LINK ACCESS ATTRIBUTES
```

1st Process addresses

@ means put directly on device queue (Don't use SysQ)

/ means skip 1st process

otherwise put call on SysQ

```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
 0021 BSYST
01 01022'000001          1                  ; GET CHANNEL STATUS
02       000001          .IFE MSW
03 01023'177777          -1
04                       .ENDC
05       000000          .IFN MSW
06                       @CPNT              ; CP INITIAL PROC
07                       .ENDC
08 01024'000777'         CLCHK              ; UPDATE FILE SIZE INFO
09                ;**************************************************************
10 01025'000001          1                  ; IOCS - GO DIRECTLY TO PRE-OPEN
11                                          ; CALL
12                ;**************************************************************
13 01026'177777          -1                 ; RESERVED 6067
14       000000          .IFN MSW
15                       @BLKEW             ; EXTENDED MEM WRITE
16                       @BLKER             ; EXTENDED MEM READ
17                       .ENDC
18       000001          .IFE MSW
19 01027'177777          -1                 ; EXTENDED WRITE
20 01030'177777          -1                 ; EXTENDED READ
21                       .ENDC
22
23       000063 CTLGT=   .-MCCT             ; LENGTH OF DISPATCH TABLE
```

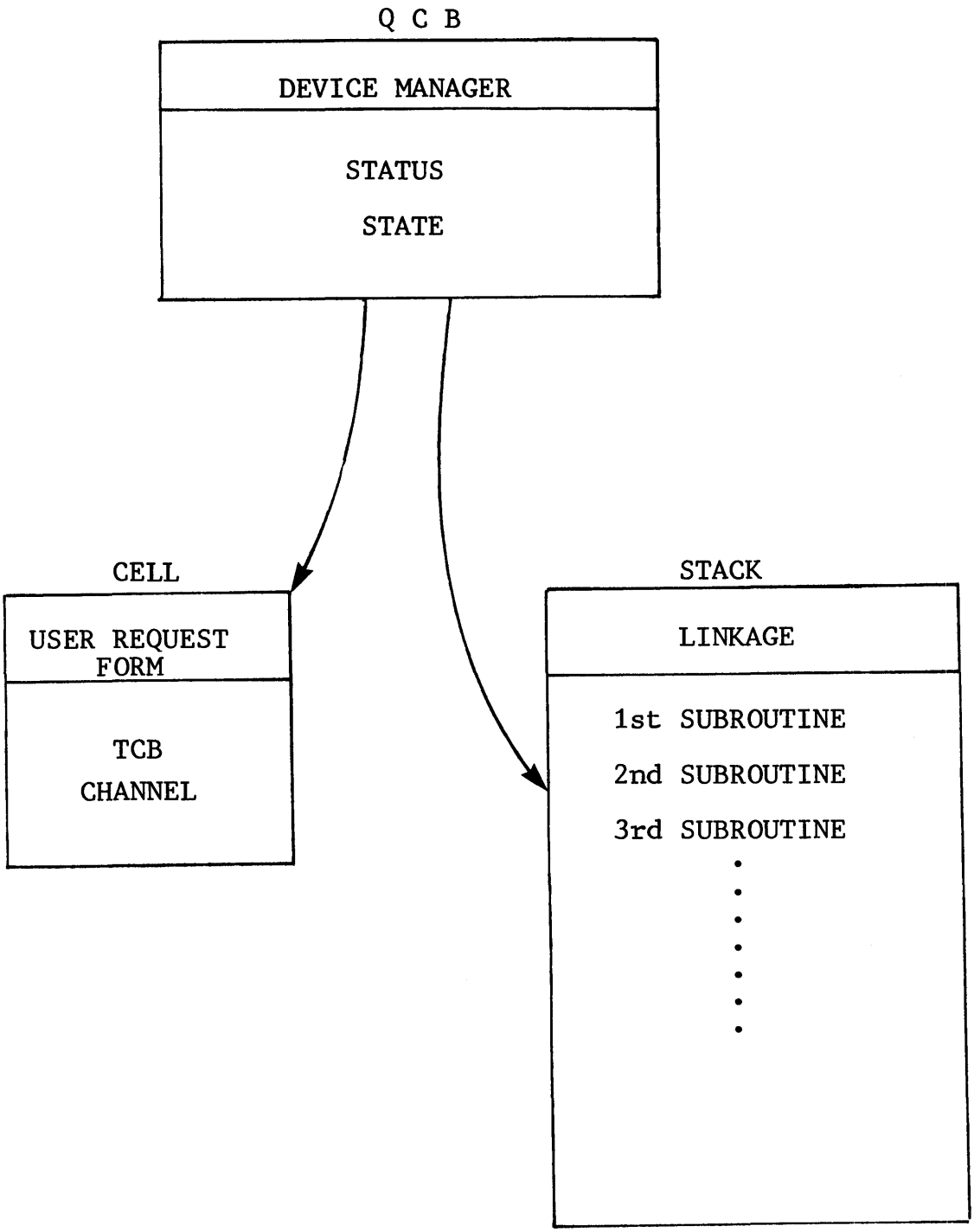
```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
10022 BSYST                                          Second process addresses
01                  ; MONITOR CALL SYSTEM DISPATCH TABLE
02
03 01031'077777 MCSDT:  CREATE              ; CREATE
04 01032'077777         DELFIL              ; DELETE
05 01033'077777         RNAM                ; RENAME
06 01034'077777         MEM                 ; AVAILABLE MEMORY
07 01035'000752'        BRKP1               ; BREAK
08 01036'077777         RLSE                ; DEVICE RELEASE
09 01037'077777         DDIR                ; DIRECTORY DEFAULT
10 01040'077777         CMN                 ; EXECUTE
11 01041'077777         INIT                ; DEVICE INIT
12 01042'077777         RETUR               ; RETURN
13 01043'077777         RSET                ; I/O RESET
14 01044'077777         RDB                 ; READ BLOCK
15 01045'077777         WRB                 ; WRITE BLOCK
16 01046'077777         ERTN                ; ERROR RETURN
17 01047'077777         CRAND               ; CREATE RANDOM
18 01050'077777         GCH                 ; TTY GET CHARACTER
19 01051'077777         PCH                 ; TTY PUT CHARACTER
20 01052'000001         1                   ; QUEUE TIME WAIT
21 01053'077777         MEMI                ; MEMORY INCREMENT
22 01054'077777         ROPN                ; OPEN FOR READING
23 01055'077777         OVOPN               ; INIT OVLYS
24 01056'077777         OPNA                ; OPEN FOR APPENDING
25 01057'077777         CHAT                ; CHANGE ATTRIBUTES
26 01060'077777         GTAT                ; GET ATTRIBUTES
27 01061'077777         OPEN                ; OPEN
28 01062'077777         CLS                 ; CLOSE
29 01063'000002         RS
30 01064'000003         RL
31 01065'000004         RR
32 01066'000005         WS
33 01067'000006         WL
34 01070'000007         WR
35 01071'077777         OVLOC               ; LOAD OVLY
36 01072'077777         CCONT               ; CREATE CONTIG
37 01073'000000         0                   ; NULL ENTRY FOR SCALL
38 01074'077777         CMNF                ; EXECF
39       000000         .IFN IOSW
40                  ;*************************************************************
41                      0                   ; IOCS - THESE PARE PCKED OFF
42                                          ; EARLIER
43                      IOCOP               ; IOCS - THE IOCS OPEN PROCESSOR
44                  ;*************************************************************
45                      .ENDC
46       000001         .IFE IOSW
47 01075'177777         -1
48 01076'177777         -1
49                      .ENDC
50 01077'077777         MTDIO
51 01100'077777         SFCP                ; SET FILE POS
52 01101'077777         GFCP                ; GET FILE POS
53 01102'077777         EOPN                ; EOPEN
54 01103'077777         TOPN                ; TRANSPARENT OPEN
55 01104'077777         CHLAT               ; CHANGE LINK ACCESS ATTRIBUTES
56 01105'077777         CHNST               ; GET CHANNEL STATUS
57       000001         .IFE MSW
58 01106'000000         0                   ; PLACE HOLDER IN RDOS
59                      .ENDC
60       000000         .IFN MSW
```

```
        LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
 0023 BSYST
01                      CMNB            ; EXBG
02                      .ENDC
03 01107'077777         UPDAT           ; UPDATE FILE SIZE INFO
04        000000        .IFN IOSW
05                    ;*****************************************************
06                      PREOP           ; IOCS - PRE-OPEN IOCS PROCESSOR
07                    ;*****************************************************
08                      .ENDC
09        000001        .IFE IOSW
10 01110'177777         -1
11                      .ENDC
12 01111'177777         -1              ; RESERVED 6087
13        000000        .IFN MSW
14                      WRB             ; EXTENDED WRITE
15                      RDB             ; EXTENDED READ
16                      .ENDC
17        000001        .IFE MSW
18 01112'177777         -1              ; EXTENDED WRITE
19 01113'177777         -1              ; EXTENDED READ
20                      .ENDC
```

Section 4

STACKS

A COMPLETE SYSTEM TASK



SYSTEM STACKS

REENTRANT CODING AND SUBROUTINE LINKAGE

One of the basic problems that arises in multi-level priority interrupt programming is that different levels require use of the same subroutine. This means that a higher priority interrupt could interrupt a lower priority one before it has completely used a common subroutine and argument pointers, temporary core storage locations, etc., with the result that the return point for the lower level interrupt would be lost. Reentrancy, then, is defined as that property of a subroutine whereby use of the routine does not modify any of its locations; temporary values are stored outside the routine. Thus one user may be prevented temporarily from completing use of a reentrant routine because a higher priority user needs to use it. When the higher priority user has finished with the routine, the lower priority user completes his use of the routine at the point where he was interrupted.

A method of reentrant coding using a system stack has been devised for RDOS to allow one subroutine to be entered at any time and from any interrupt level without loss of results. The operating system has several stacks used for the saving of state variables whenever a call to a system subroutine is executed. Each of these stacks is of a fixed length and stack frames are defined in the same manner for each stack. The stack frame is the basic increment of storage on a system stack.

SYSTEM STACK

In general, each system stack available allows the execution of one system call simultaneously with others. A new stack frame is allocated every time a new subroutine or overlay is called to complete the system task. The state is saved in the frame (accumulators, and return address), as well as any temporary data storage required.

Operation of each peripheral is considered to be a single system task. Despooling is another system task. The Disk(s) have two tasks associated with their operation. These tasks queue up behind the device drivers which actually access the devices. Each task requires a system stack.

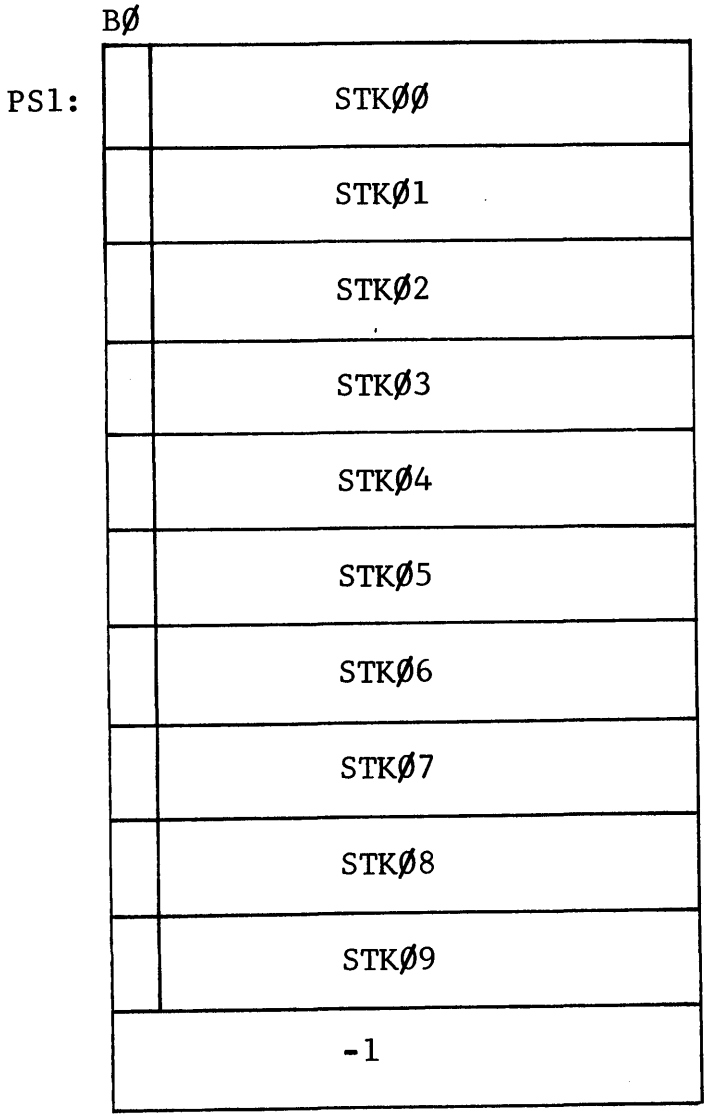
STACK ALLOCATION

Most system (.SYSTM) calls require a system stack to be allocated for the duration of the call. (i.e. the stack remains allocated until the system call is completed).

(Note that a .RDL, .RDS, or .GCHAR system call to $TTI or $TTI1 ties up a stack until the user types the required input).

STACK CONTROL WORDS

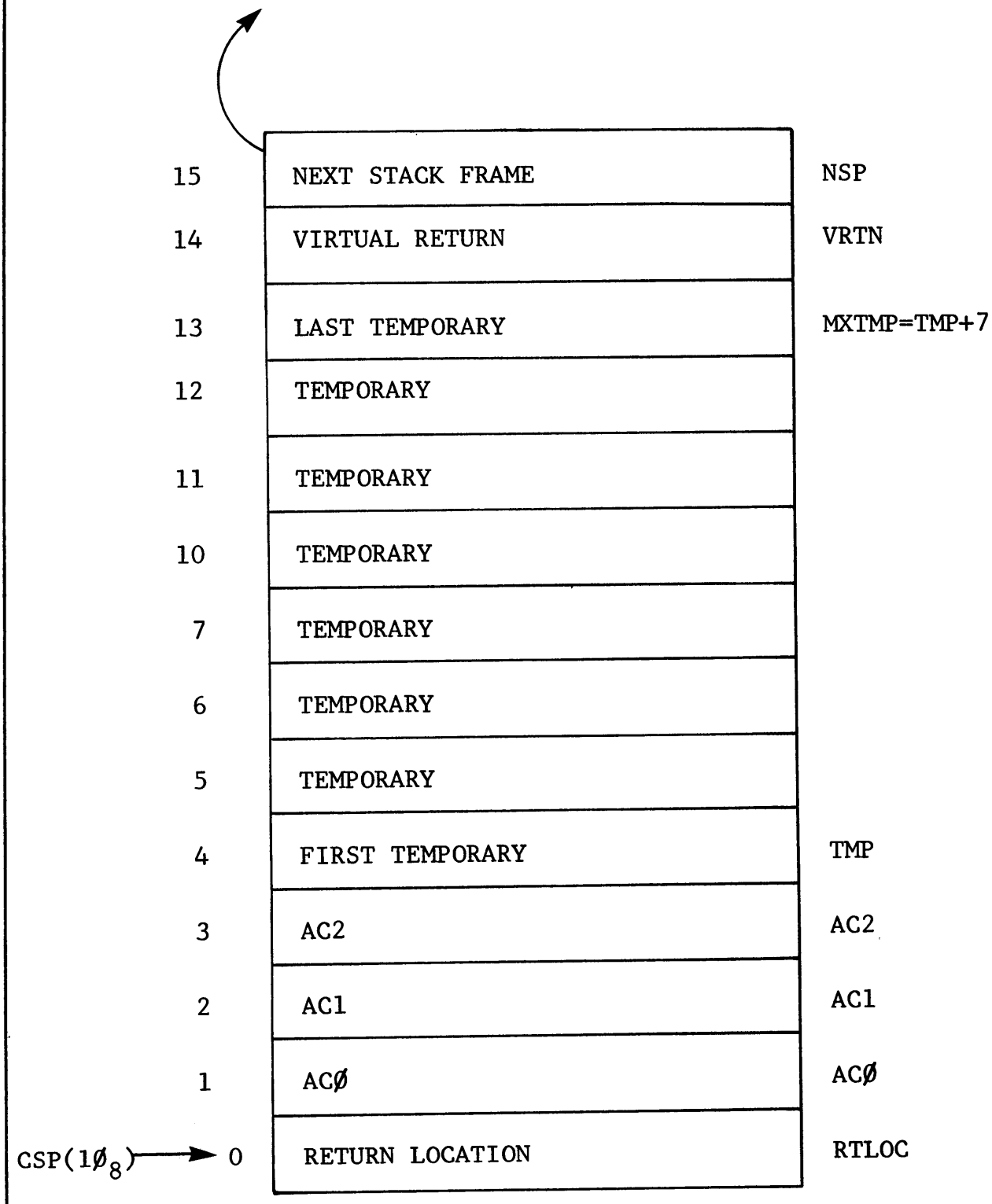
located in TABLE.SR



1BØ : STACK IN USE

ØBØ : STACK FREE

SYSTEM STACK FRAME



STACK FRAME STRUCTURE

Word 0;	RTLOC	Return location. This is the location at which processing will continue when a return is made from the called subroutine to the caller.
Word 1,	ACØ	Used to store caller's ACØ
Word 2,	AC1	Used to store caller's AC1
Word 3,	AC2	Used to store caller's AC2
Word 4,	TMP	First temporary location in stack frame
Word 13,	MXTMP	Last temporary location in stack frame
Word 14,	VRTN	Used to store virtual return address. This is used when the return is to be made to a system overlay (which is unlikely to be in the same system buffer later).
Word 15,	NSP	Next Stack pointer. This points to the start of the next stack frame. (Set to .+1) 1BØ set in word following last stack frame.

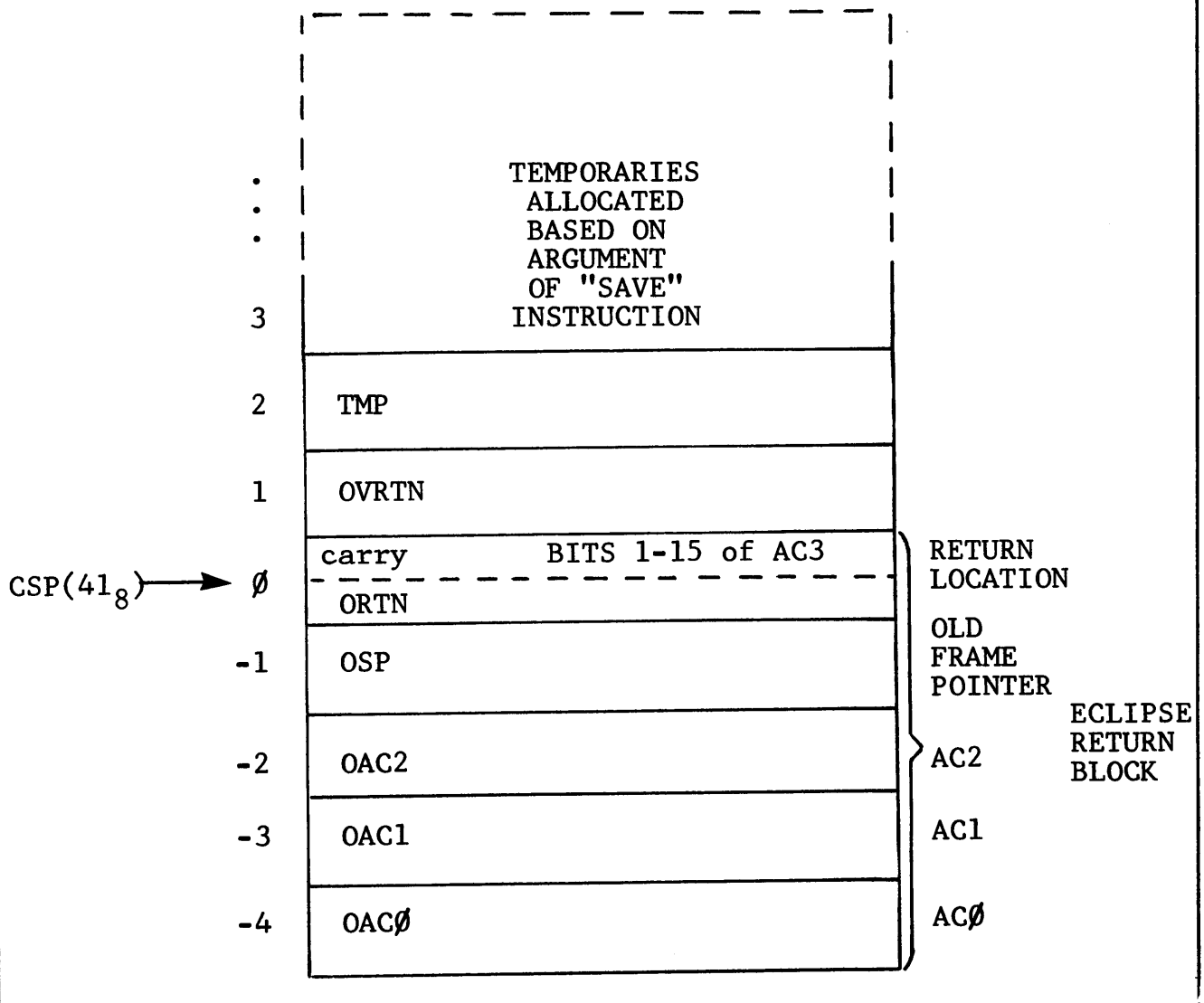
STACK FRAME STRUCTURE — ECLIPSE

Words Ø → - 4	ECLIPSE Return Block allocated by execution of ECLIPSE "save" instruction.
Word 1	Virtual return address. Same usage as above.
Words 2 → ?	Temporary storage locations allocated by execution of ECLIPSE "save" instruction. Quantity is variable from "one" to "many"!

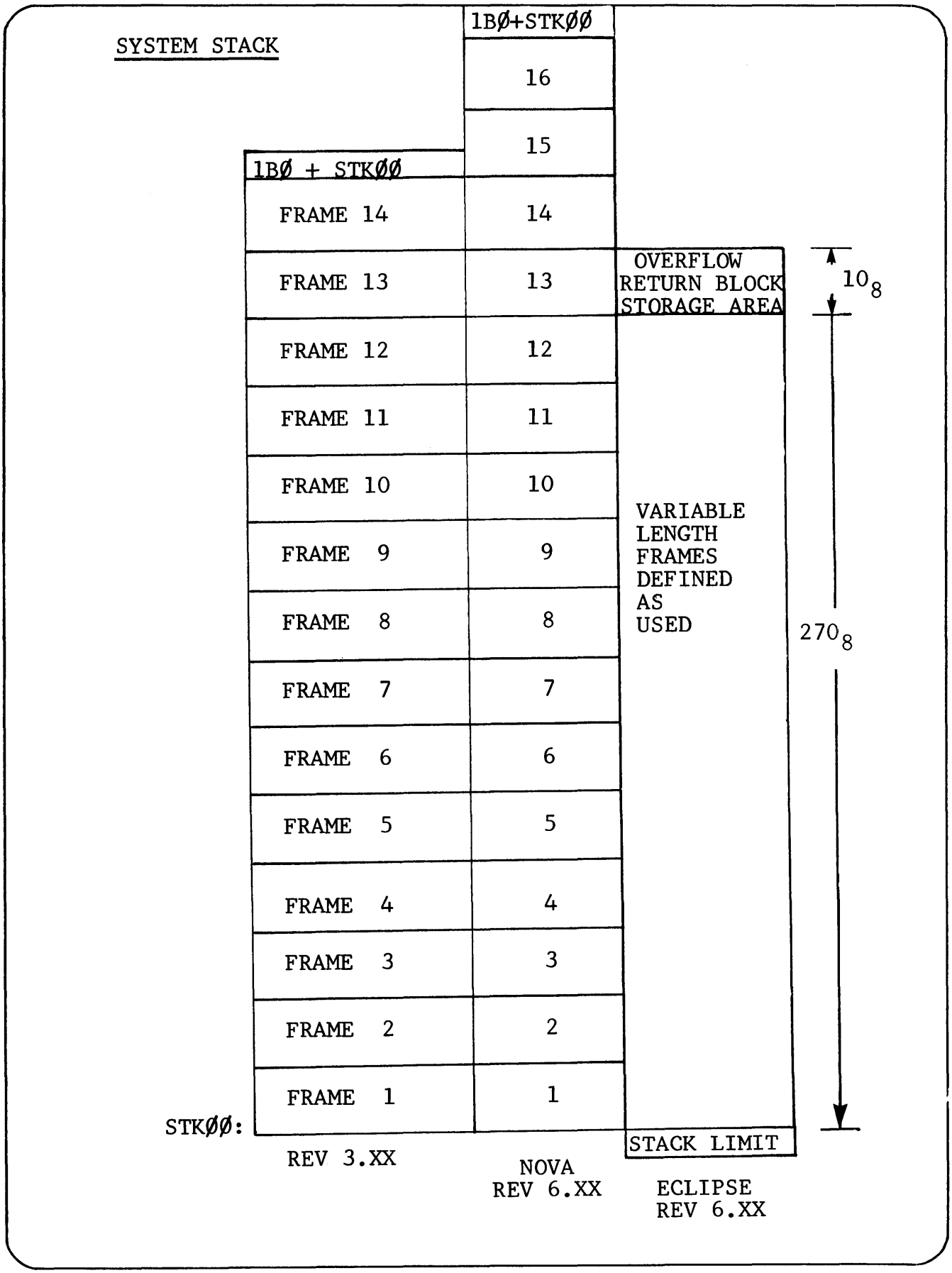
STACK CONTROL WORDS

To enable the system to be aware of whether a stack is in use or not, a stack control table is maintained. This table has a one-word entry for each system stack configured into the operating system. This one-word entry contains the base address of the system stack, and also indicates (as 1BØ) whether or not the stack is in use.

SYSTEM STACK FRAME



REV 6XX ECLIPSE



The only system calls that do not require a stack for the entire duration of the call are:

(1) QTY .RDS/.WRS & .RDL/.WRL CALLS
(2) .DELAY calls
(3) .RDOPR, .WROPR calls

A stack is required to set up the call, but it is then released.

(Note that the QTY driver will never require more than one stack.) "SET UP" phase, but no "REACTION" SYSTEM task.

If in any program environment fewer system stacks are available at any single moment than are required at that moment there will be a holdup. The system will service simultaneously only as many requests as it has available stacks. It will service requests in the order in which they are made by a program. If the foreground has priority over the background, its calls will be processed ahead of the background.

The current stack frame is pointed to by a page zero system location, CSP. with CSP in AC2 or AC3, the system can access locations in the frame using offsets defined in PARS.SR and explained on the following pages.

	REV 6.XX
NOVA	CSP = 10_8
ECLIPSE	CSP = 41_8

KNOWN AS ECLIPSE HARDWARE
"FRAME POINTER"

Access to frame locations:

```
LDA     3,CSP
LDA     Ø,ACØ,3     ; NOTE ECLIPSE STACK INSTRUCTION
                      NOT NEEDED
```

STACK LINKAGE - NOVA

Linkage to save state variables in a system or interrupt stack frame and restoration of these variables is provided by three routines (SAVE, RTRN and SRTRN) in the utility package within the system. The current Stack Pointer (CSP) is a page zero state variable, maintained by the system, for all stacks.

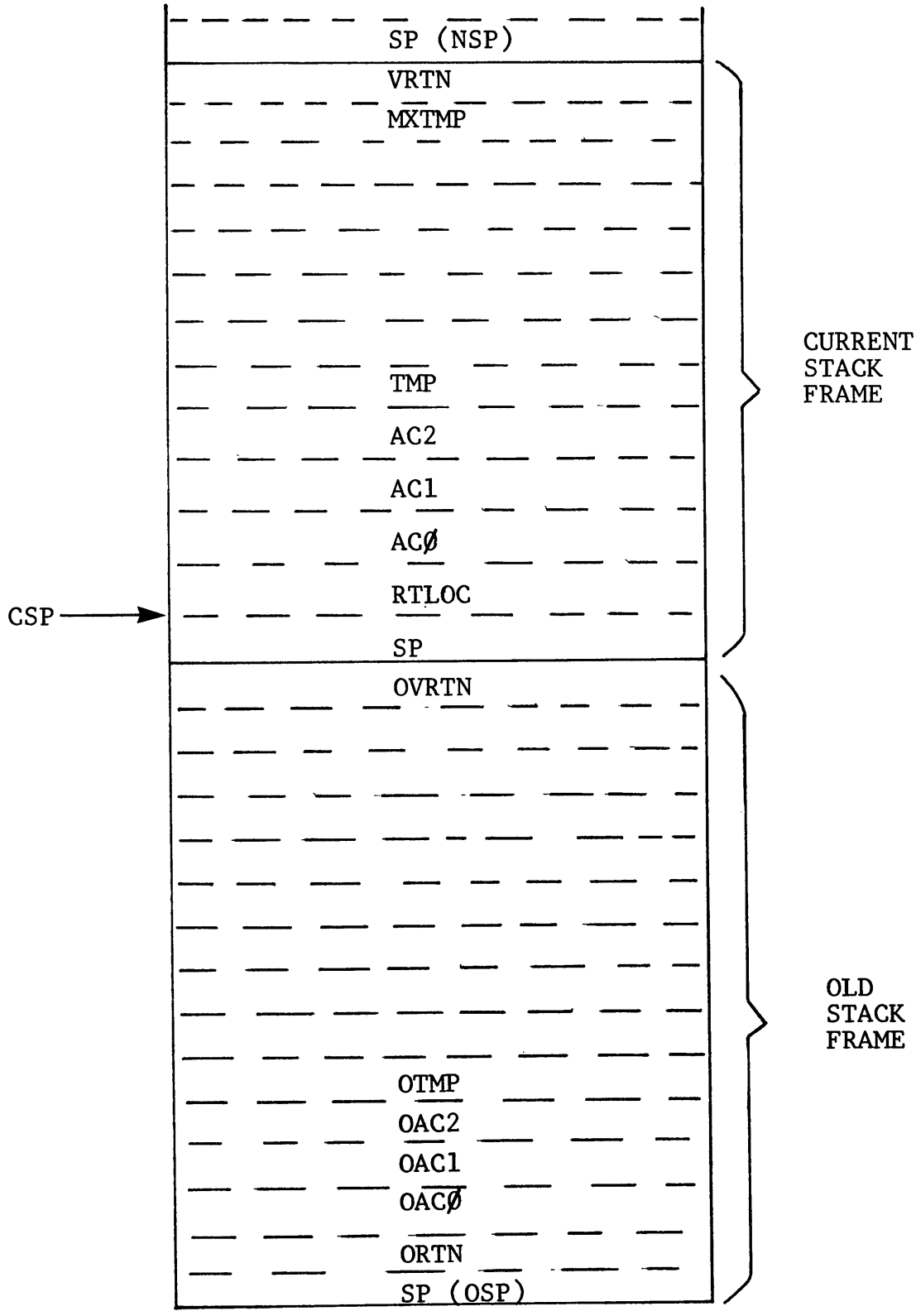
The initialization code for the operating system sets the pointers (SP) for each frame of the interrupt stack. These pointers are set at system load time for the system stacks. The pointer following the last frame of each stack has 1BØ. The system parameter tapes define offsets for both the current stack frame, as well as the old (last) frame. In this manner a subroutine can access the caller's frame's contents.

STACK LINKAGE - ECLIPSE

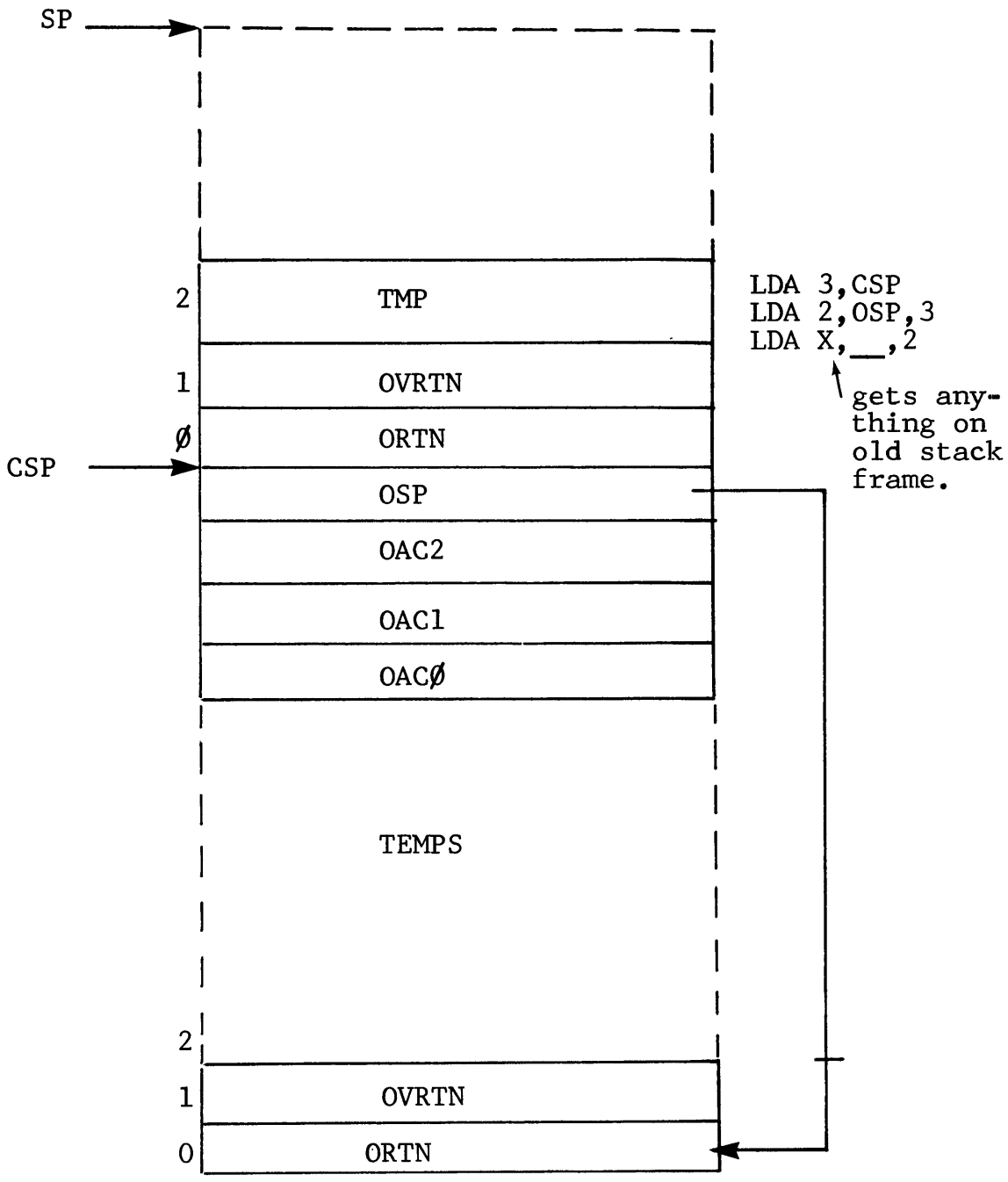
Linkage to save state variables is performed by the two ECLIPSE instructions "SAVE" and "RTN". Referencing old frame info can only be done by a task linking through successive "OLD FRAME" pointers.

LINKAGE ON STACKS

NOVA



LINKAGE ON STACKS ECLIPSE

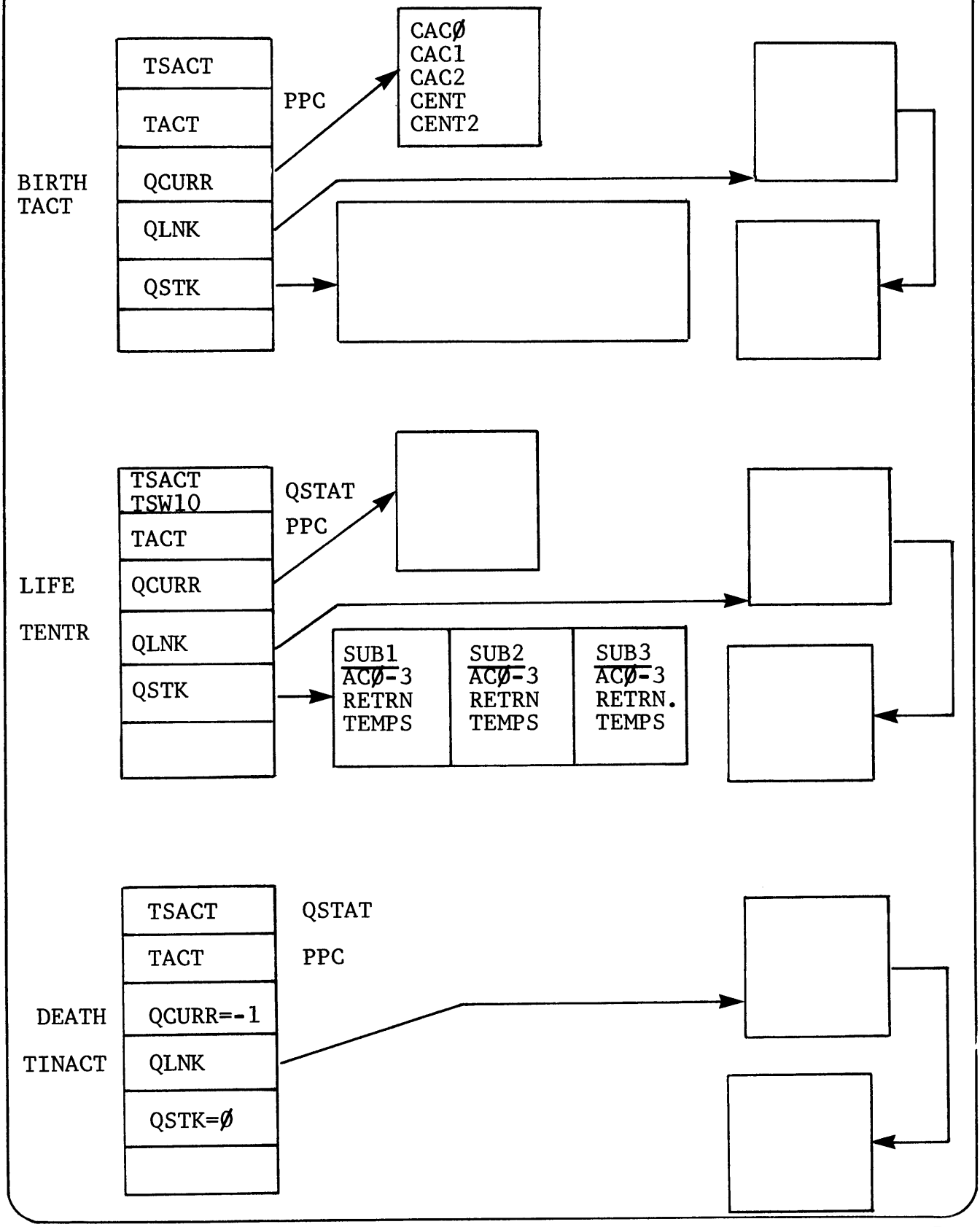


notes

Section 5

PAGE Ø
AND
TACT FLOW

AGES OF A SYSTEM TASK

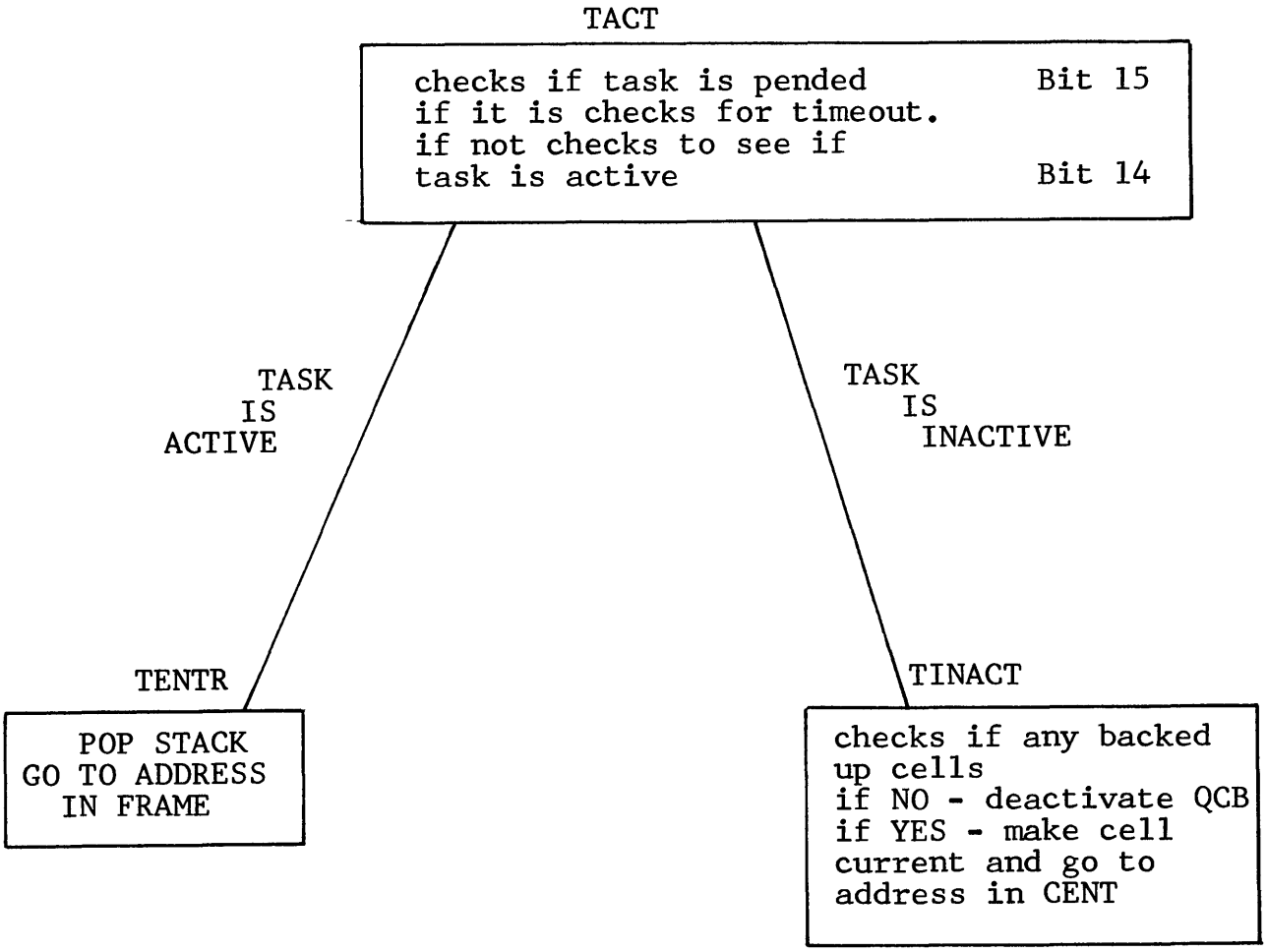


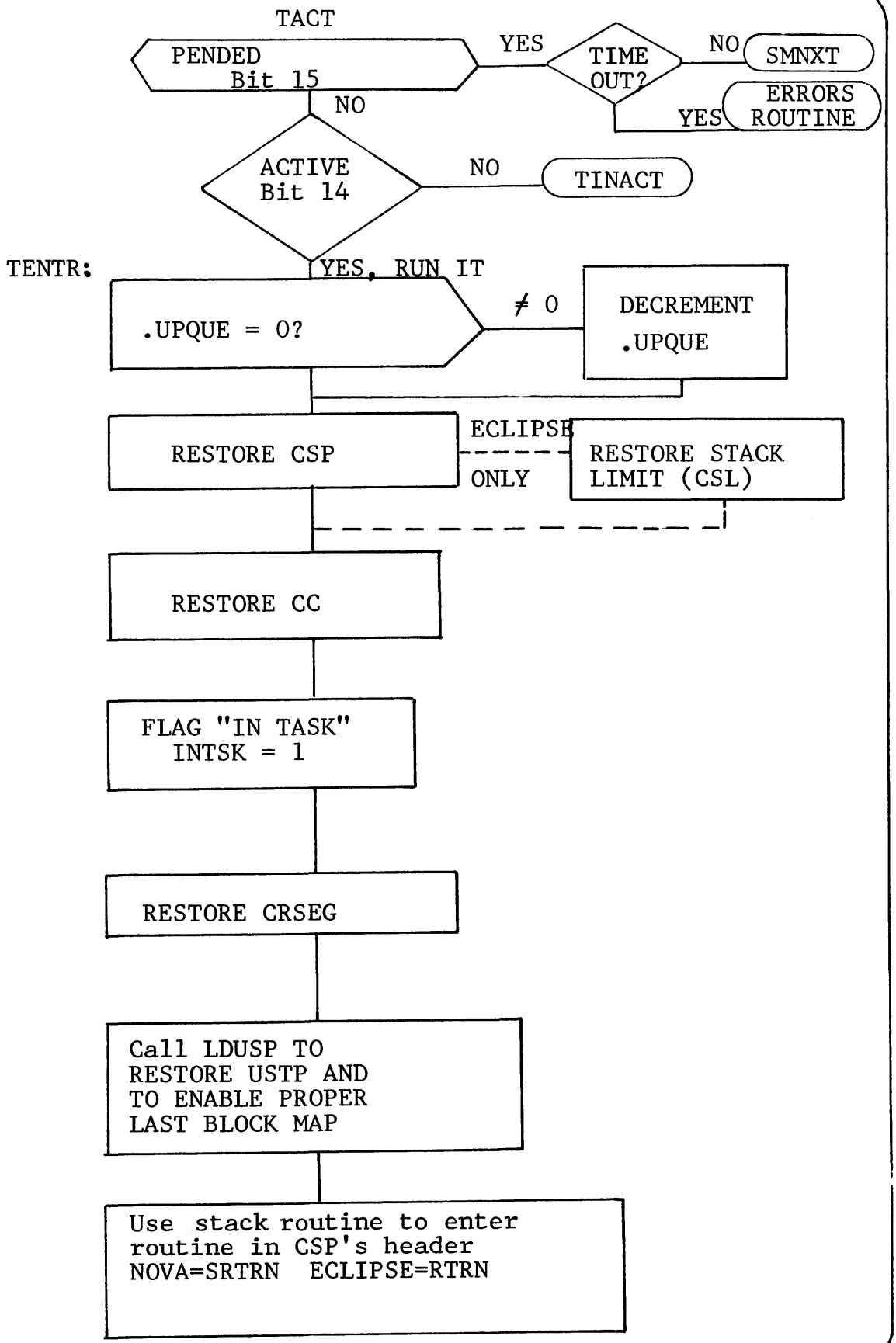
SYSTEM PAGE ZERO STRUCTURE *NOVA* REV 6

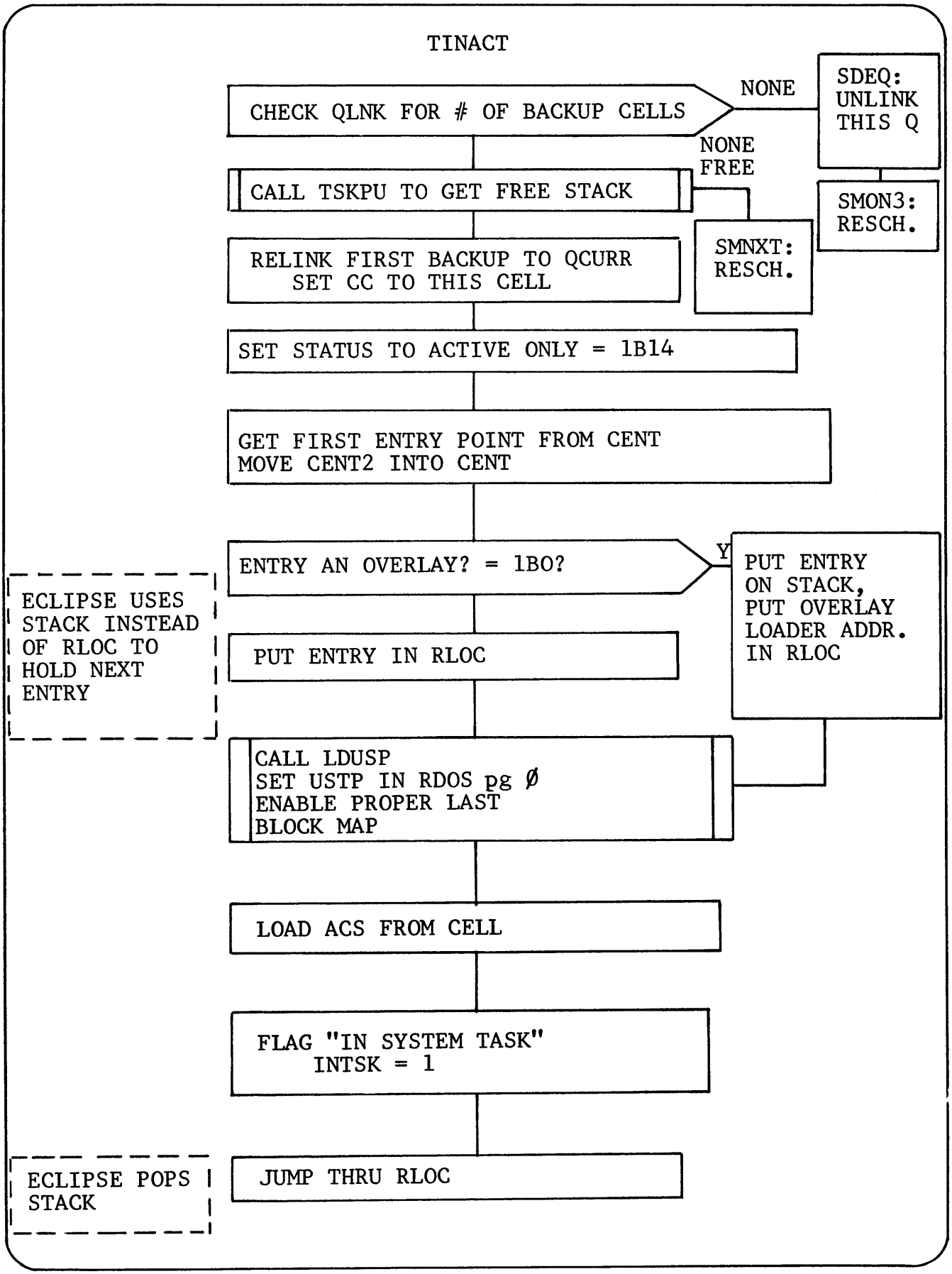
LOCATION	USE
*0	(*Not Defined in Paru/Pars) Location Ø is loaded with the current PC after an interrupt. The hardware then simulates a JMP @ 1.
1 INTS	Contains the address of the interrupt service routine entry point
2 SYST	Contains the entry point of the system from the user world. Used by the user task schedulers.
3 USTKO	Nova 3 Stack overflow address
*4 RTN	Contains the entry point of the routine to return a frame to the stack and restore the state variables.
5 CC	Contains the address of the current cell.
6 RLOC	Temporary used by the system. Preserved through interrupt handling.
7 SAV	Contains the entry point of the state save routine. Called when executing JSR and requiring a stack frame.
10 CSP	Pointer to the current stack frame. Could be a system stack or the interrupt stack.
11 .PNIC	Entry point to the system panic handler routine.
12 USTP	Contains the address of the current user status table.
13 CQ	Contains the address of the current system task queue control block or program table.
14 CRSEG	Contains the address of the overlay table entry for the currently executing request.
15 CMSK	If in core code = Ø Current interrupt mask word.

Ø		Same as NOVA
1	INTS	Same as NOVA
2		Same usage as NOVA, referenced by "SCL" type instruction
3		Map protection fault routine address
4	ISP	Interrupt stack-stack pointer
5	CMSK	Current interrupt priority mask
6	ISL	Interrupt stack-stack limit
7	ISO	Interrupt stack-stack overflow routine address
1Ø	CC	Current cell
11	PNIC	Panic Routine Pointer
12	USTP	User status table pointer
13	CQ	Current task queue
14	CRSEG	Current overlay or Ø
15		Unused
4Ø	SP	Current stack pointer
41	CSP	Current Frame Pointer
42	CSL	Current Stack Limit
43	CSO	Current Stack Overflow Routine Address
44	XOPA	XOP Table Starting Address
45	FPFA	Floating Point Fault Routine Address
46	CFA	Commercial Fault Routine Address (C/300, INFOS only)
47		Reserved for Future Use

STATE RESTORE ROUTINE







MULTITASK CONTROL

NO PRIORITIES

= NO SYSTEM TASK CAN

PRE-EMPT

ANY OTHER

ONCE A TASK IS

EXECUTING,

IT WILL RETAIN CONTROL;

INTERRUPTS WILL BE PROCESSED,

BUT

EXECUTION WILL BE

RETURNED TO THAT TASK'S CODE

SYSTEM TASKS

CAN

CHANGE THEIR OWN CONTROL BLOCKS

BECAUSE

THE SCHEDULER IS NEVER RUN

BEHIND THEIR BACKS

PRIME DIRECTIVE

OF MULTITASK SCHEDULING

DO NOT USE

CONTROL TABLES

FOR

RESCHEDULING

WHILE

THOSE TABLES ARE

BEING CHANGED

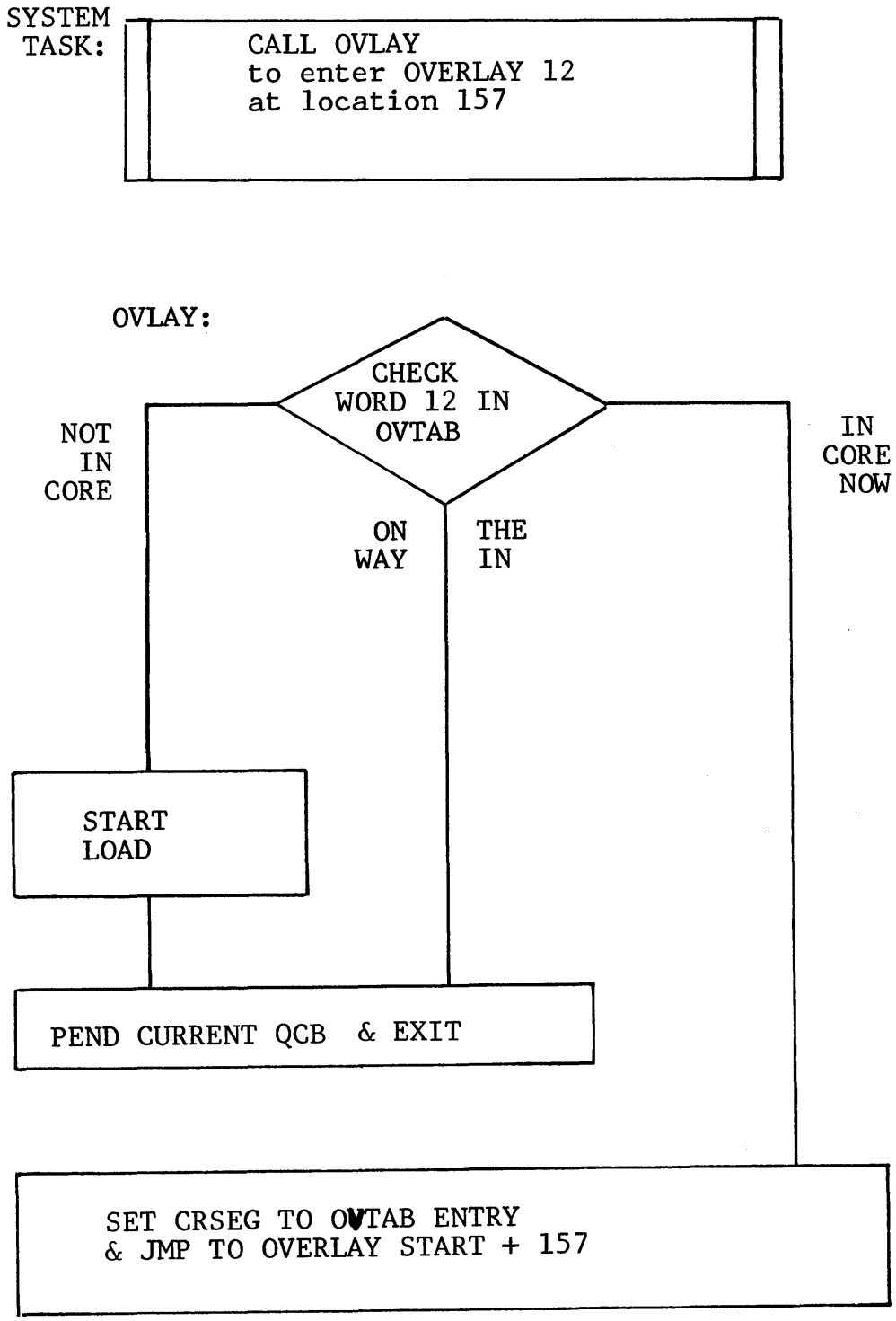
notes

notes

Section 6

OVERLAYS

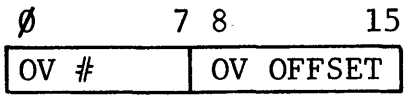
SYSTEM OVERLAYS



SYSTEM OVERLAYS

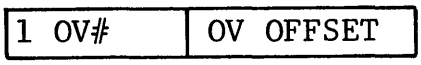
A system overlay is a portion of the operating system code that usually resides on the disk. When it is required that an overlay be executed it is brought into core into any available system buffer. The only restriction this places on the writing of system overlays is that no absolute addressing can be done where the location used as an address points to the overlay itself. Indirect addressing can occur if the address is an external which references the root portion of RDOS. When a return address must be used to an overlay (i.e., one overlay uses a subroutine in another overlay) this is created as a virtual overlay address. This must be done since the calling overlay may not reside in the same memory locations when the return is made.

Overlay virtual return:



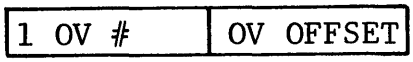
*Return to any point within overlay

Overlay virtual entry:



*Entry only to first half ($Ø$-177_8) of overlay

Virtual Entry:



0 1 7 8 15

*Only up to 128 overlays entry in either half

OVERLAY TABLE

To control the loading of system overlays into system buffers, and to make available to the system the locations of overlays currently in core, there is an overlay pointer table (OVTAB). This table has one word entries for each system overlay. This entry specifies whether or not the overlay is in core, being loaded into core, or non-resident.

i.e.	If ØBØ	Overlay is either resident or being loaded. The low order 15 bits contain a buffer address if overlay is resident. The low order 15 bits contain +1 if the overlay is being loaded.
	If 1BØ	Overlay is not resident. The word contains either the overlay table address (i.e., its own address) or Ø. If it contains an address then the overlay has been loaded at some time. If it contains Ø the overlay has never been loaded.

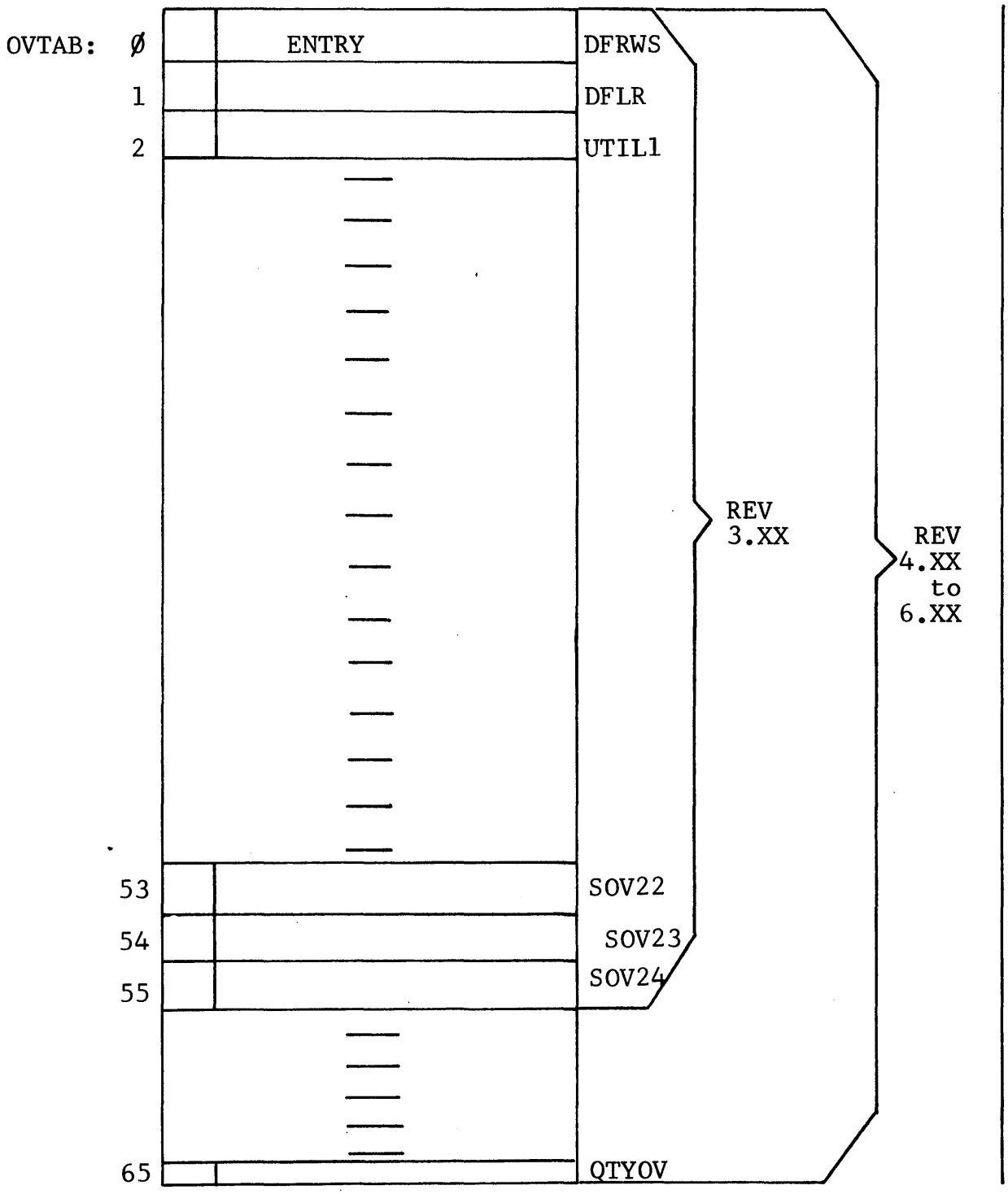
A one-word database OVTAB is the first entry in the overlay table.

A page zero one-word database CRSEG contains the overlay table address of the currently executing segment. If CRSEG contains Ø then resident code is executing. CRSEG is used by the overlay loader to make a virtual return and release current OV. Before loading next OV.

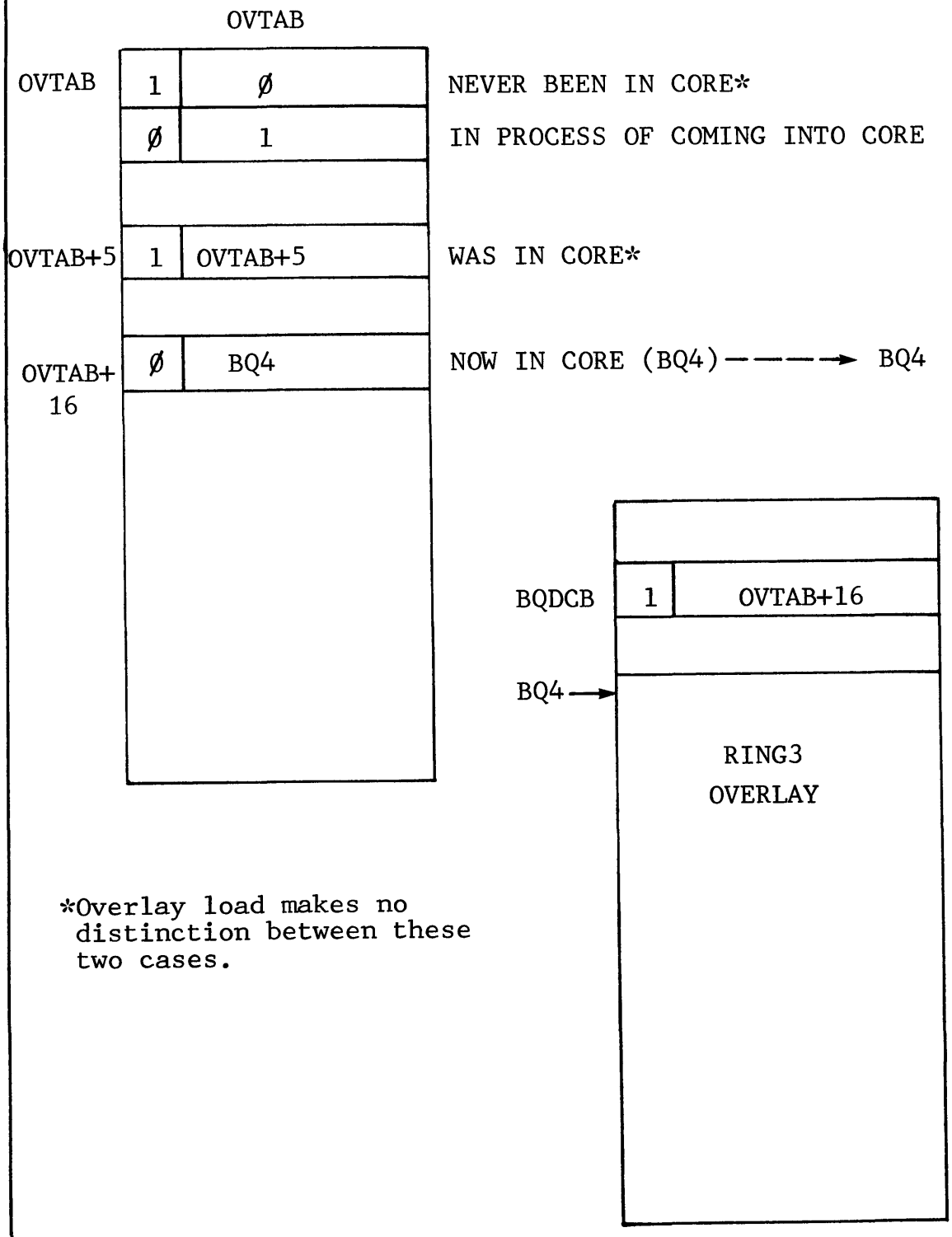
A database OVBAS contains the logical base address of the disk overlay file. Rev 3.XX is one word, REV 4.XX to 6.XX 2 words. The database is located in the OVLAY.SR module.

OVERLAY TABLE

located in TABLE.SR



OVERLAY TABLE

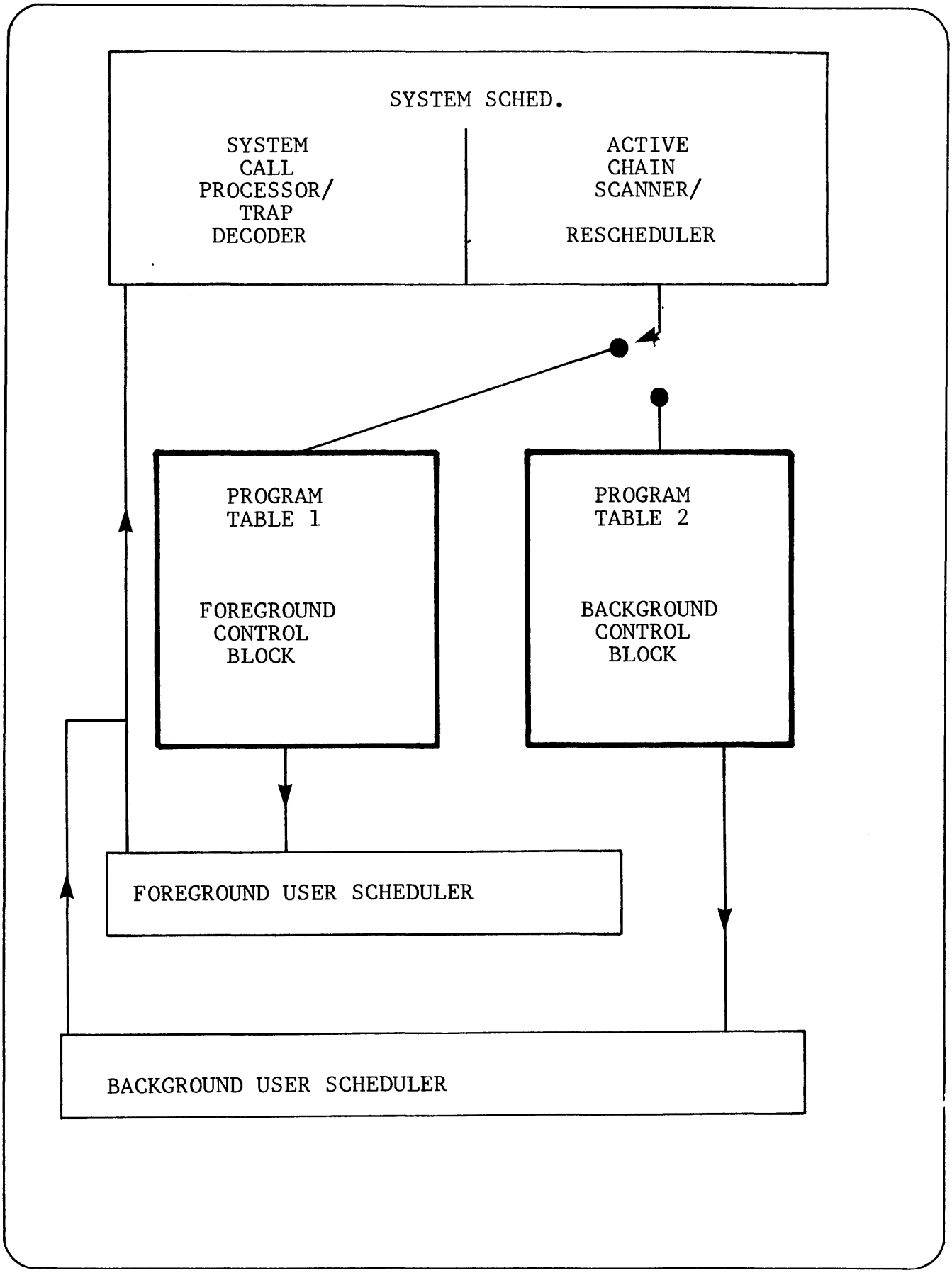


*Overlay load makes no distinction between these two cases.

notes

Section 7

PROGRAM TABLES



PROGRAM TABLES

For the system to have available full information on a current program level it has to maintain a user program database. This database is called a Program Table (PT). There is one program table for each user area. i.e., PT1 is the program table for the foreground and PT2 is the program table for the background.

This table contains active information on the program level, and is also used to store state variables when switching between foreground and background.

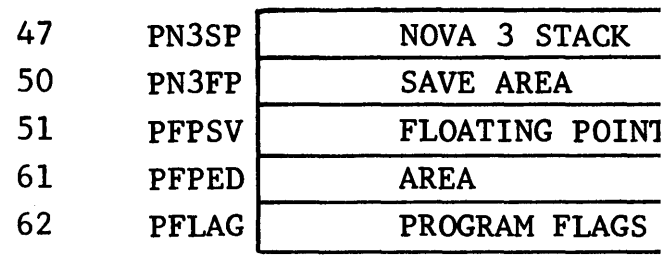
A one-word database is maintained to indicate whether or not there is a foreground program in the system. It is called SYSFG. If SYSFG contains 1 there is a foreground program. If there is not a foreground program it contains zero.

A one-word database is maintained to indicate the priority of an existing foreground program. It is called PRISW. If PRISW contains 1, then the foreground program has equal priority with the background. If PRISW contains Ø, then the foreground has higher priority than the background.

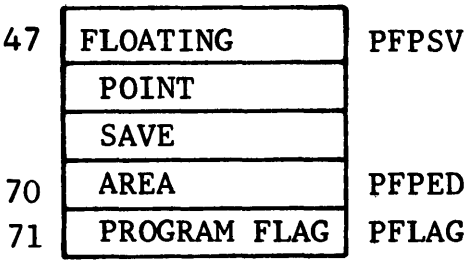
PT1 is the address of the start of the foreground program table.

PT2 is the address of the start of the background program table.

RDOS



BRDOS



PROGRAM TABLE STRUCTURE

PCLAC	User clock active count. This is being counted down to zero at each RTC interrupt.
PCLCN	User clock constant. The number of RTC interrupts between user clock interrupts. Used to initialize PCLAC.
PCLAD	User clock processing address. i.e., Start address of user clocks service routine.
PLNK	Link to other program table. (i.e., link to PT2 if in PT1 or -1 if in PT2).
PMST	Map Status

NOTE

Words 4 7 are common entries for all system task control blocks. The program tables form part of the active chain when operating.

PSTAT	Status Word. If any of the flags are set then system will not return to the user scheduler. See later description of the flag settings.
POLNK	Link to next control block in active chain (i.e., in system queue). PT1→PT2 PT2=-1
PPRI	Program priority: Foreground = 1; Background =2
PPC	Program counter to return to. Always set to PENTR in PTS

NOTE

The next three words are used to control. .DELAY CALLS. A description of delay chains follows.

PDFR	Delay chain start address. (Delay time is a TAC1 of user's TCB)
PDEN	Delay chain end.

PDTOT	Total time delay on the queue.
PUSTP	Pointer to User Status Table.
PSWD	Temporary. Used by system to back-up TCB's when no free cells. When a cell becomes free, the TCB can be picked up from this location.
PDDCB	Current default directory DCB address.
PTPB1 PTPBA	Base logical block address of the current partition. (2 word address)
PTCLF	Current level flag $\emptyset \rightarrow 4$
PTSPN	System push number for the parent program level $1 \rightarrow 4$ Background $5 \rightarrow 10_8$ Foreground
PINTU	Program interrupt word. If 1B15 = 1, wake up program after an interrupt.
PUFPT	Address of User File Pointer Table.

NOTE

The next two locations refer to the foreground/background communications area.

PCMST	Start address of FG/BG Communication Area
PCMSZ	Size in words of FG/BG Communication Area
PDCNT	Delay active count. The number of delay requests on the queue. (MRDOS & ARDOS)
PFLNK	File System link (system use)
PFLN1	File System link (system use)

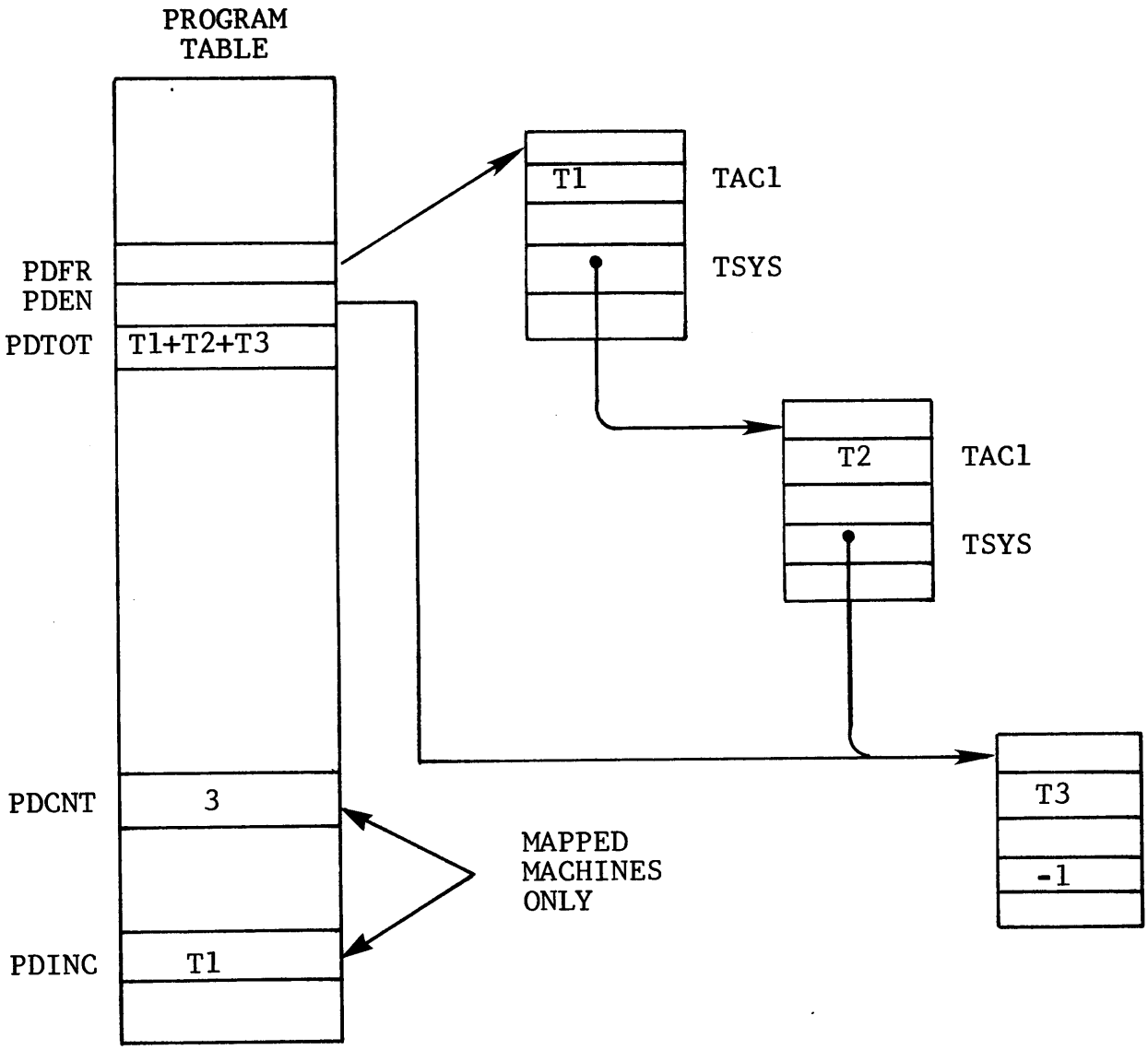
DELAY CHAINS

All user program tasks which have caused themselves to be delayed for a period of time have their TCB's chained together. The address of the first TCB in the chain is pointed to by PDFR. This TCB refers to the task with the shortest outstanding delay. (The outstanding delay is stored in TAC1 of the TCB). The TCB's are chained in order of the completion of their delays. The delay time stored in TAC1 of each TCB is the delay outstanding after the previous TCB's delay is over. The TCB's are linked by TSYS. The address of the last TCB in the chain is stored in PDEN. PDTOT is the total delay time outstanding before all the TCB's on the chain have been processed.

DELAY TIME INCREMENT (MRDOS & ARDOS)

All user tasks involved in a time delay (DELAY) have their TCB's in a chain off the Program Table (PDFR). In the unmapped system the first delay time (stored in the first TCB's TAC1) is decremented at each real time interrupt. In the mapped system it would be inefficient to have to go to into user address space and decrement this count at each clock interrupt. Instead the delay time for the first TCB in the chain is copied into a word in the Program Table, PDINC. It is this word which is decremented at each real time clock interrupt.

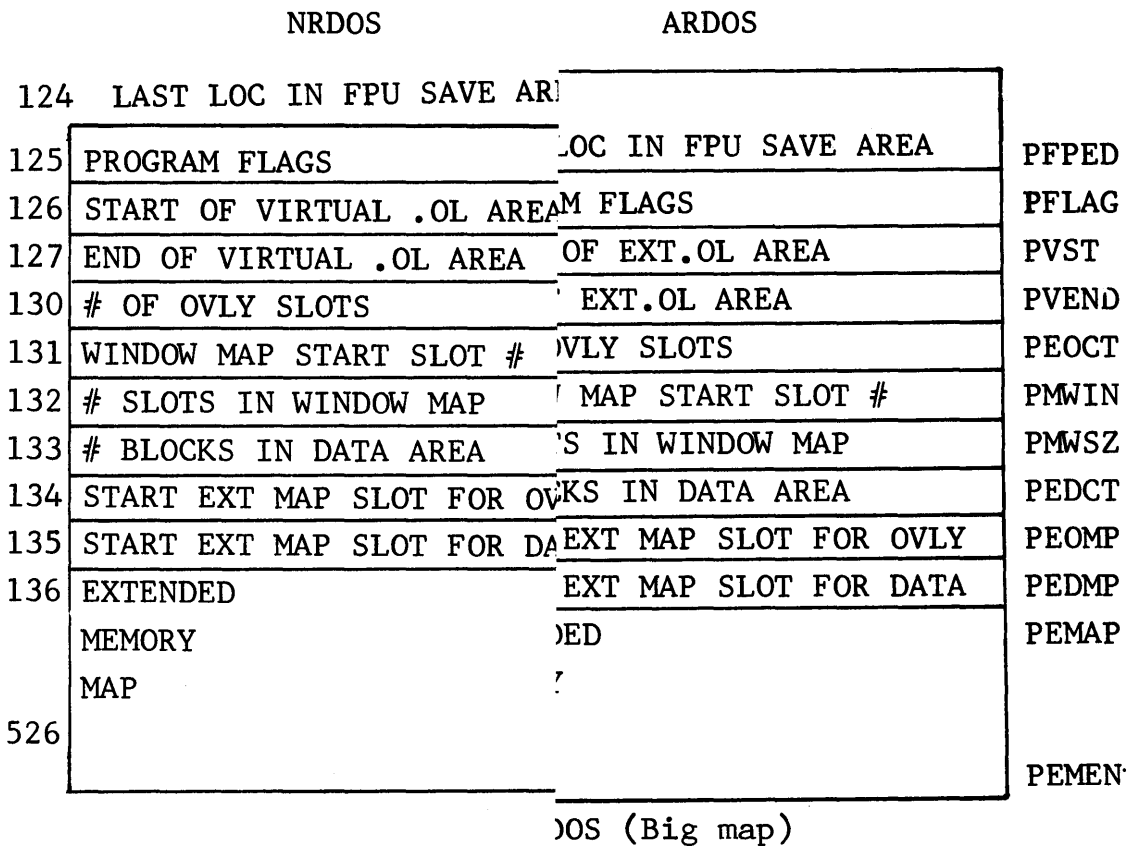
DELAY CHAIN



NRDOS
MRDOS
ARDOS
ZRDOS

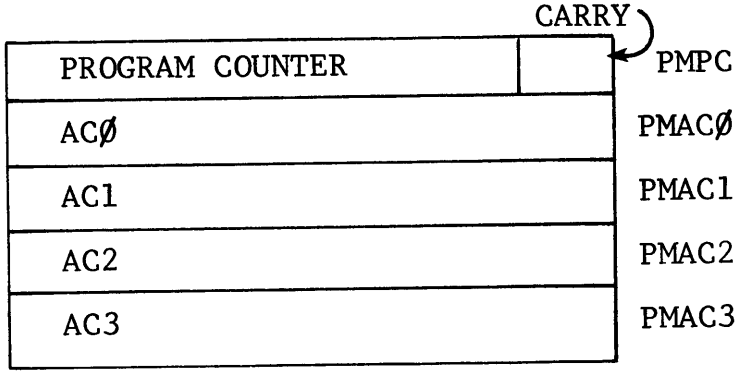
Ø	NOVA 3 & ECLIPSE MAP STATUS WORD	PMST	
1	ACTIVE USER CLOCK COUNT	PCLAC	
2	USER CLOCK CONSTANT	PCLCN	
3	USER CLOCK PROCESSING ADDRESS	PCLAD	
4	STATUS WORD	PSTAT	*COMMON ENTRIES FOR ACTIVE CHAIN
5	ACTIVE CHAIN LINK	POLNK	
6	PRIORITY OF TASK/PROGRAM	PPRI	
7	PC OF STATE RESTORE ROUTINE	PPC	
10	DELAY CHAIN START	PDFR	
11	DELAY CHAIN END	PDEN	
12	TOTAL DELAY ON QUEUE	PDTOT	
13	USER STATUS TABLE POINTER	PUSTP	
14	HEAD OF BACKUP TCB'S	PSWD	
15	DEFAULT DIRECTORY DCB ADDRESS	PDDCB	
16/17	DISK PARTITION BASE ADDRESS	PTPB1/PTPBA	
2Ø	CURRENT LEVEL FLAGS	PTCLF	
21	SYSTEM PUSH NUMBER	PTSPN	
22	PROGRAM INTERRUPT WORD	PINTU	
23	USER FILE POINTER TABLE	PUFPT	
24	COMMON START	PCMST	
25	COMMON SIZE	PCMSZ	
26	DELAY ACTIVE COUNT	PDCNT	
27	SYSTEM LINK WORD	PFLNK	
3Ø	SYSTEM LINK WORD	PFLN1	
31	STATIC LINK	PLNK	

notes



STATE SAVE AREA (MRDOS ONLY)

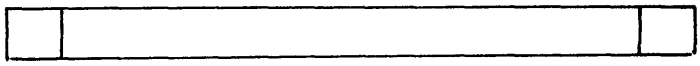
At interrupt time if the user program proper is executing, a TCB is available to carry out a state save of PC and accumulators. If the user scheduler is executing then no such area is available. To provide such an area (in system address space) the Program Table Save Area includes 5 words of storage.



FLOATING POINT SAVE AREA

When switching between foreground and background programs it is necessary to save the state of all machine registers. If the configuration includes a Floating Point Unit (FPU), or Eclipse floating point processor, then the values in the registers of the unit should also be saved. If the FPU was not being used by both the foreground and the background this would obviously be unnecessary. The operating systems allow users to specify whether the FPU is being used in their program areas. Only if both programs are using the FPU simulteneously will the system save the state of the FPU registers. The system call used is the "Enable User Access" command (.DEBL). In the unmapped world this is the only device for which the system call is not treated as a no-op.

A one word database is used to indicate which program area(s) is using the FPU. This word is called FPUSV. The format is:



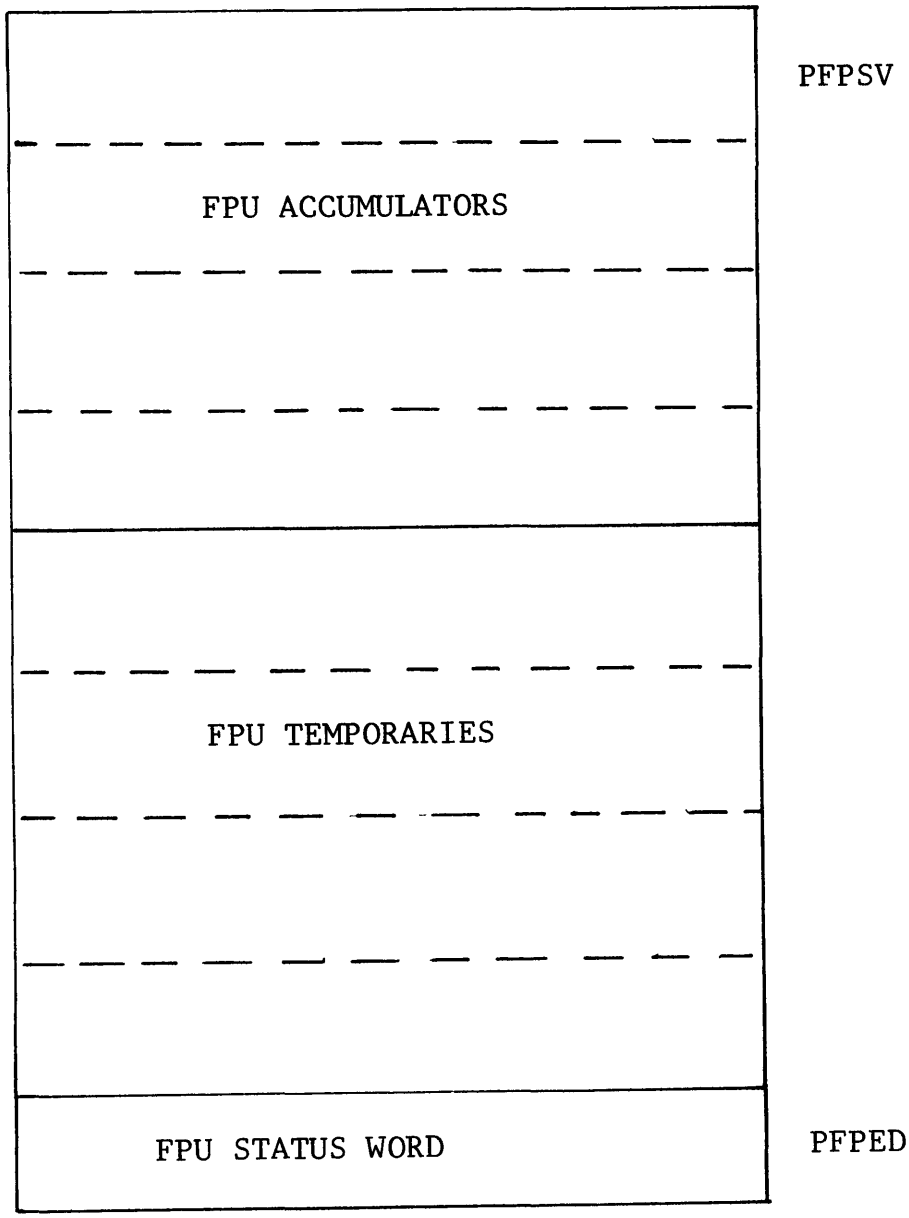
1BØ Indicates foreground using FPU
1B15 Indicates background using FPU

Thus only if both 1BØ and 1B15 are set will the system save the state of the FPU.

Another one word database is used as a flag to indicate which program area the FPU registers are currently applicable to. This is called CFPU. This can be used to determine whether in fact the FPU registers have to be re-loaded or whether they are still applicable.

FLOATING POINT SAVE AREA (RDOS AND MRDOS)

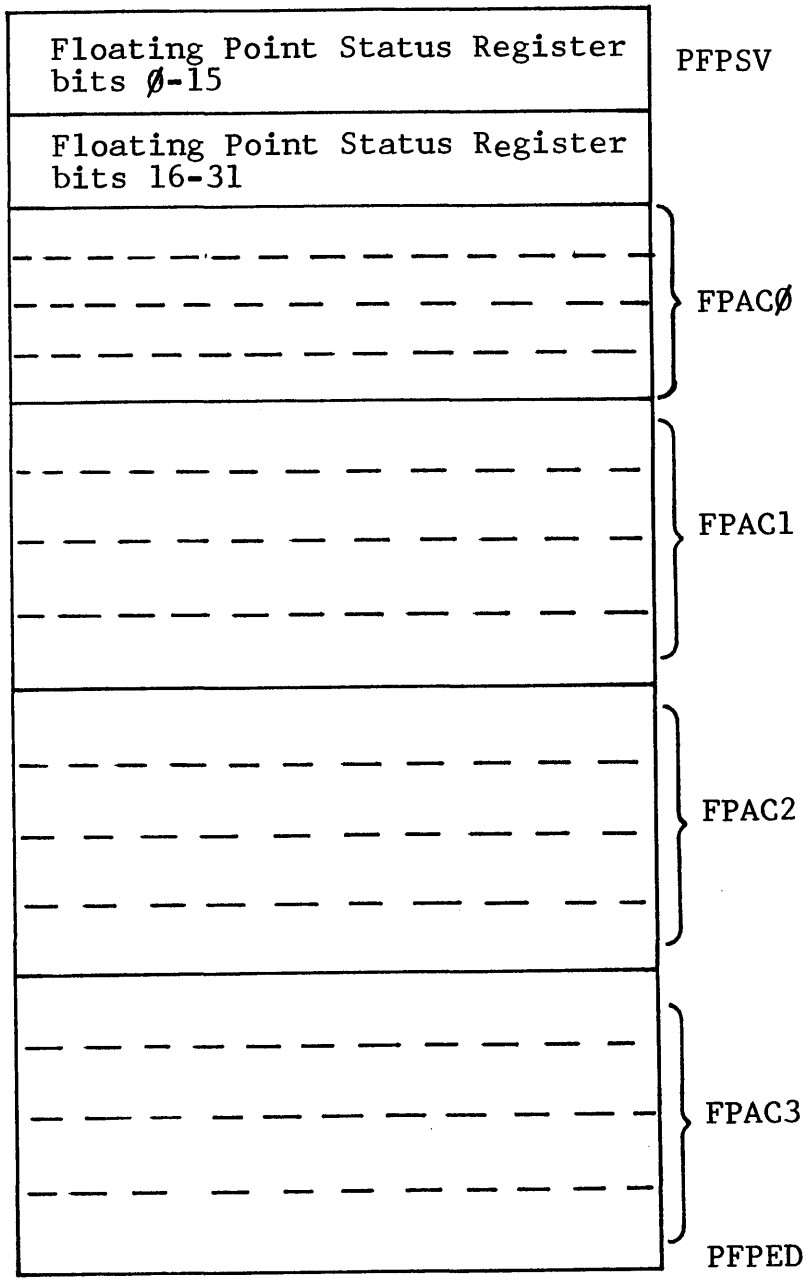
NOVA



FLOATING POINT STATE SAVE AREA

ECLIPSE

(Accessed by ECLIPSE BRDOS/ARDOS by the "FPSH" and "FPOP" Floating Point stack instructions.)



PROGRAM FLAGS (MRDOS AND RDOS)

PFLAG contains two possible flag settings:

1BØ Program not interruptable. The system call (.ODIS) to disable operator interrupts has been made.

1B15 Program not checkpointable. A background program is not checkpointable if it uses QTY I/O or uses any of the following system calls:

.DELAY
.RDOP
.IDEF/.IRMV
.DUCLK/.RUCLK

Nested checkpoints are not allowed.

* CHECKPOINTED PROGRAM DATABASES

CPFLG: 1BØ : old BG using FPU
1B14: old PRISW=1
1B15: BG already checkpointed

CPUSH: Push level reached before checkpoint

CPFPS: Save area for checkpointed programs FPU/FPP

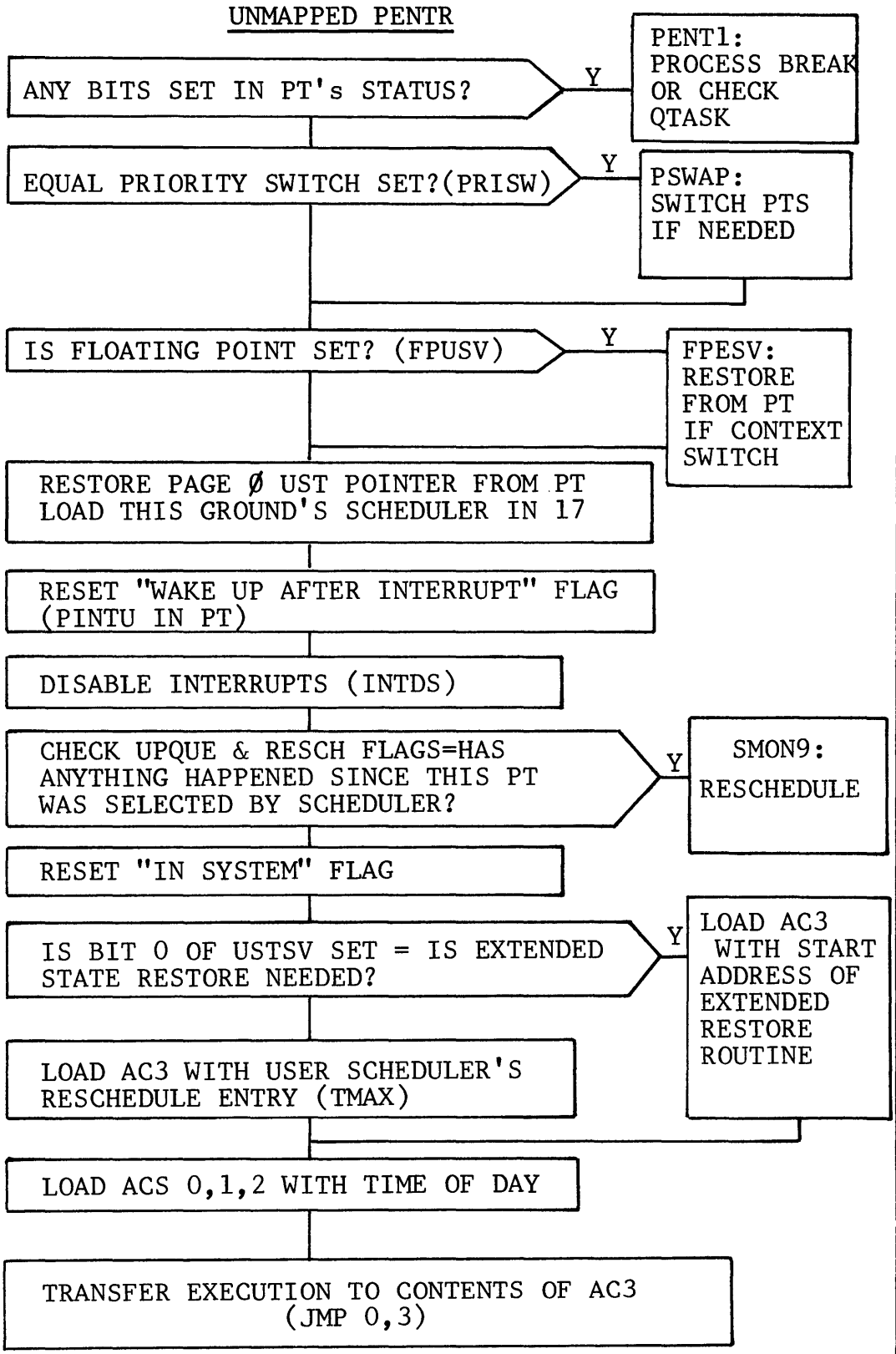
CPDSV: Save area for checkpointed programs device protection registers

PROGRAM STATUS WORD FLAGS

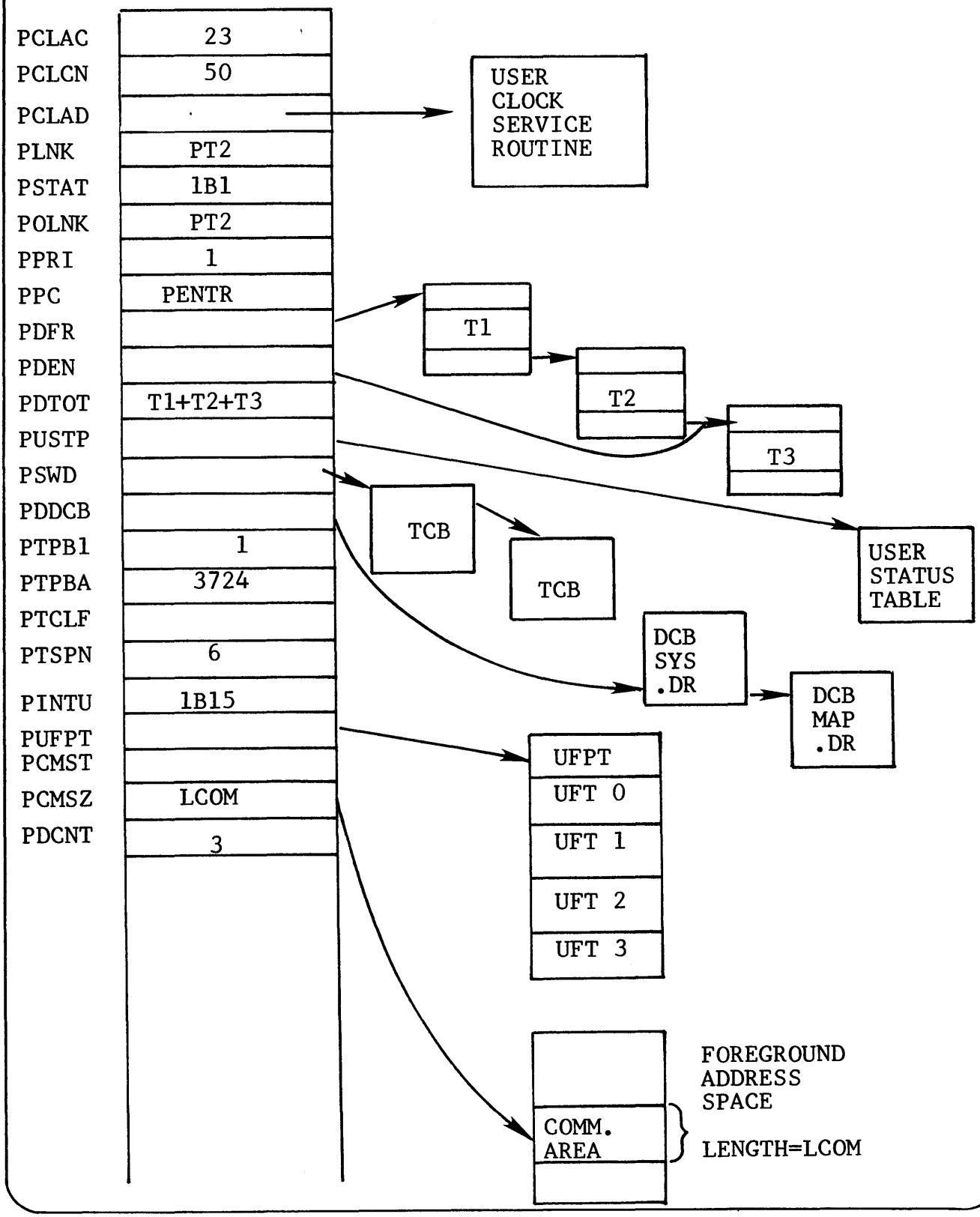
These flags are in the status word, offset +4 (PSTAT). If any of the flags are set, then the system will not return to the user scheduler. The following list is in order of relative priority.

BIT	MNEMONIC	INDICATION IF SET
1BØØ	PSRDY	If set, not ready to run. Common to all control blocks on the active chain.
1B11	PSCP	Checkpoint request received for this program. (ARDOS, MRDOS, NRDOS background only)
1B13	PSEW	Waiting for action. i.e. program level being loaded, pushed, etc.
1B12	PSBRK	Operator break requested. (or .BREAK)
1BØ1	PSQW	QWAIT active flag (QTASK action) system will check every second if any action is required.

PSRDY is the only bit inspected by the system scheduler (SMON). All other bits are analyzed by PENTR after SMON has decided that bit Ø indicates that the user program is ready to run. This division of power allows SMON to be simple and generalized.



MAPPED NOVA FOREGROUND PT



notes

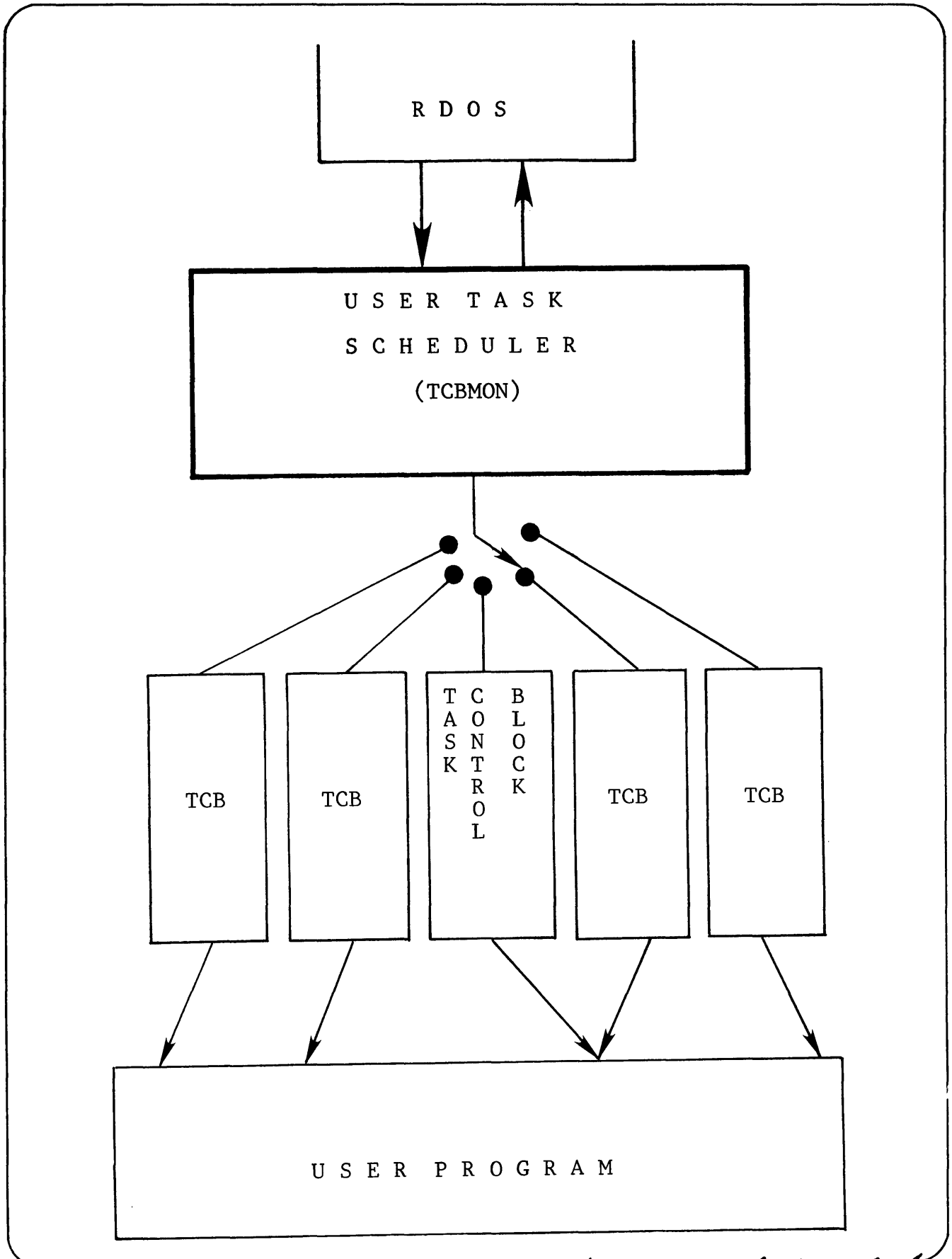
notes

Section 8

USER

TASK SCHEDULAR

(TCBMON)



In mapped systems all TCB's kept IK block core limit to 44 tasks

USER TASK SCHEDULER

MULTITASKING PRIME DIRECTIVE

DO NOT USE CONTROL TABLES FOR RESCHEDULING WHILE THOSE TABLES ARE BEING CHANGED

The user task scheduler is the entry point JMPed to by PENTR. This scheduler (and its user tasks) is the only system task that can be pre-empted by another system task as soon as an interrupt readies it. Whenever the readied task finishes the user world is re-entered at the user scheduler. This arbitrary rescheduling after each significant interrupt makes it difficult to obey the prime directive.

Whenever TCBs are being changed (task calls and .SYST calls) the user scheduler sets a "don't reschedule" flag. If a significant interrupt occurs while the flag is set, the pre-interrupt state is copied from the interrupt stack to the Executive TCB (ETCB). ETCB is the user scheduler's own state save table and is automatically loaded regardless of the number of user tasks. When the user scheduler is entered, it always checks the flag. If set, it restores from ETCB without even looking at user TCBs.

USER TASK SCHEDULER

Every user program loaded under RDOS will contain a user task scheduler by default. The scheduler which acts as the interface between the user and RDOS, is a relocatable binary module extracted from the system library. For a single task program a minimal version of the scheduler, TMIN, is loaded. TCBMON is the more powerful version that supports multitasking. Location 17 within the user program will always contain the address of the user scheduler's system call routine. Every system call results in a transfer of execution to the contents of location 17.

```
.SYST = JSR @ 17
```

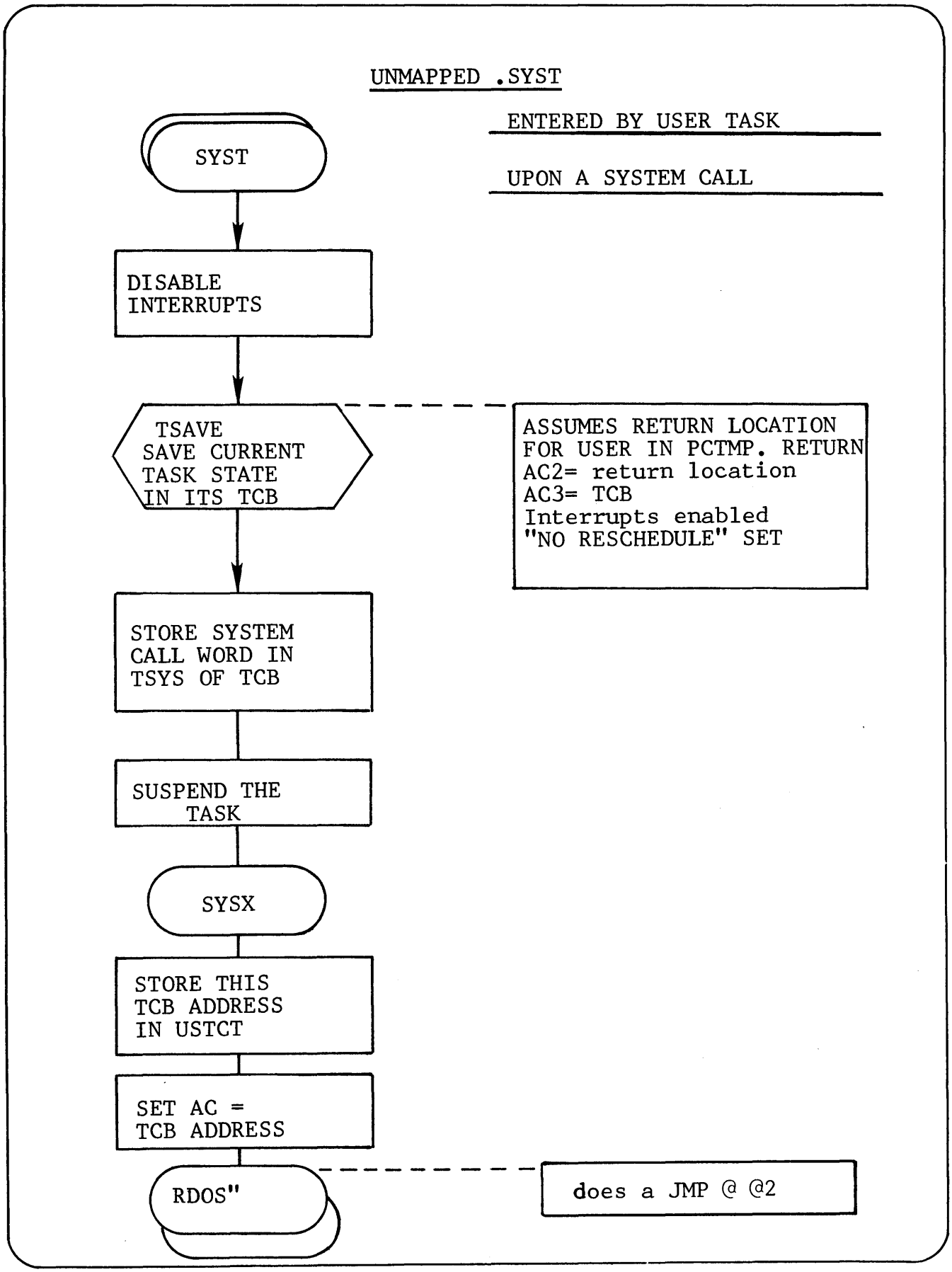
The scheduler will flag the fact that it is executing and will store the state of the CPU in the calling task's TCB. This state saving routine is also available for use by the code envoked through task calls. In unmapped environments, in which controller level I/O is not regulated, the scheduler turns interrupts off (INTDS) immediately, saves the CPU state, flags it's running and then reenables interrupts (INTEN). In mapped machines, the scheduler immediately flags "in scheduler", then saves the state. In a mapped system, all interrupt control is done in RDOS.

In either case, the scheduler will store a pointer to the newly filled TCB in an accumulator and transfer control to RDOS's system call processor. In an unmapped machine this is done by jumping through the contents of location 2. In a mapped environment, system addresses are not available to the user and a form of map trap must be used instead. Disabling the map unit in this manner causes execution to resume in a portion of memory in which RDOS has loaded a trap decoder.

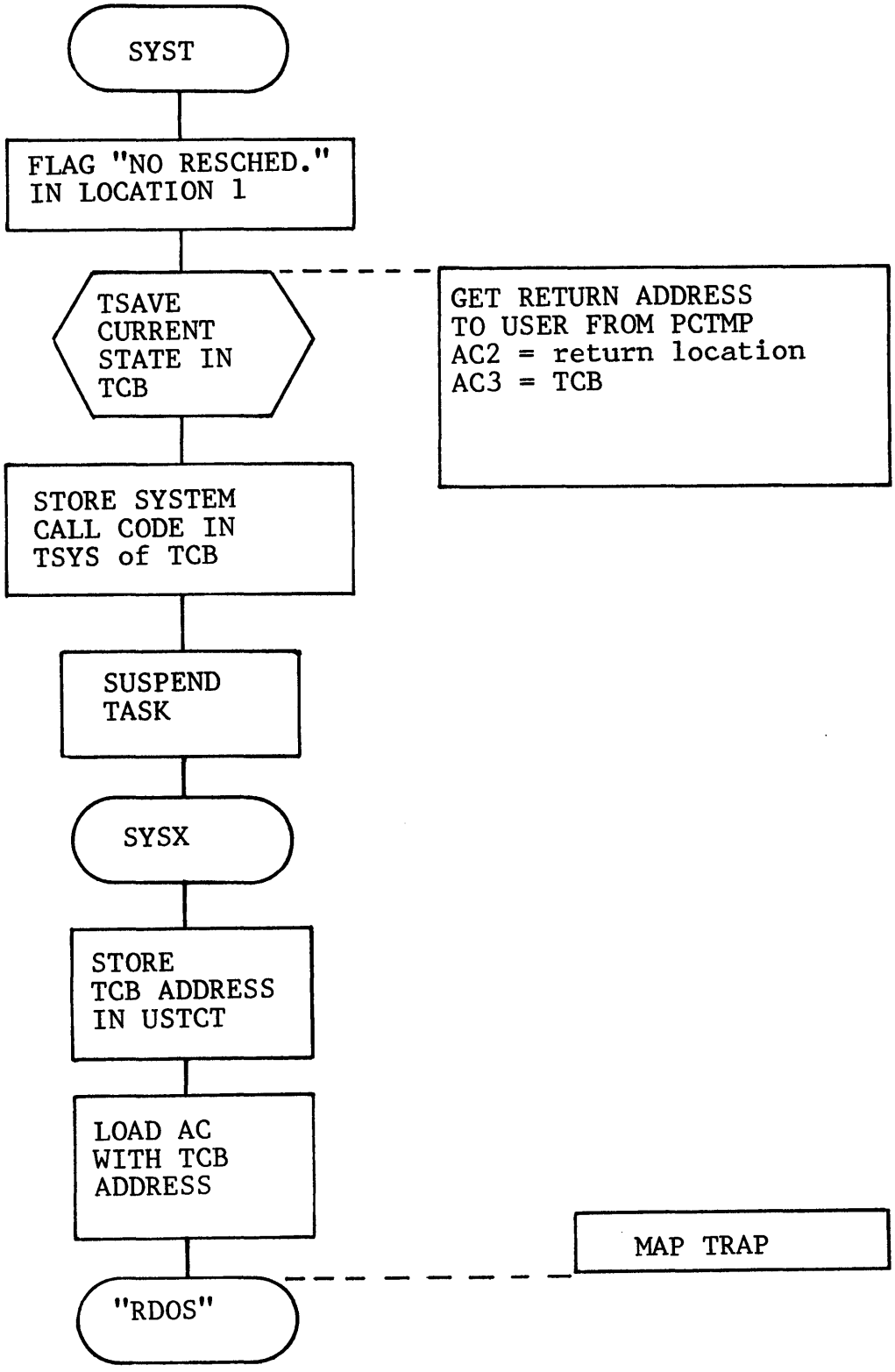
Both TMIN and TCBMON process system calls in the preceeding way. TCBMON has the added responsibility of choosing among a number of possible tasks to run whenever RDOS allows the user program to execute. This rescheduling is done every time there is a change in the status or priority of any user task.

POINTS OF RETURN FROM RDOS

FROM WHERE?	HOW WAS RDOS ENTERED?		
	SYSTEM CALL	INSIGNIFICANT INTERRUPT	SIGNIFICANT INTERRUPT
USER SCHEDULER	RESCHEDULE	CONTINUE	CONTINUE
USER TASK		CONTINUE	RESCHEDULE



MAPPED .SYST



USER/RDOS INTERFACE

USER TASK CODE

- SYSTEM CALL REQUEST
- TCB ADDRESS → AC
- USED TO FILL IN A CELL
- GROUND IS IDLE
- SETS PT STATUS TO PENDED

MAPPED SYSTEMS ONLY

USER INTERRUPT CODE

- RETURN FROM ".IDEF" CODE
- RDOS ADDRESS SPACE PROTECTED
- SWITCHING INTERRUPTS ON & OFF
- MAP COULD PROTECT DEVICE 77

BOTH

- CONTROL HARDWARE DIRECTLY
- (NOT THROUGH SYSTEM TASK)
- MAP CHANGES
- CONTROL LFE

MESSAGES TO RDOS FROM USER SCHEDULER

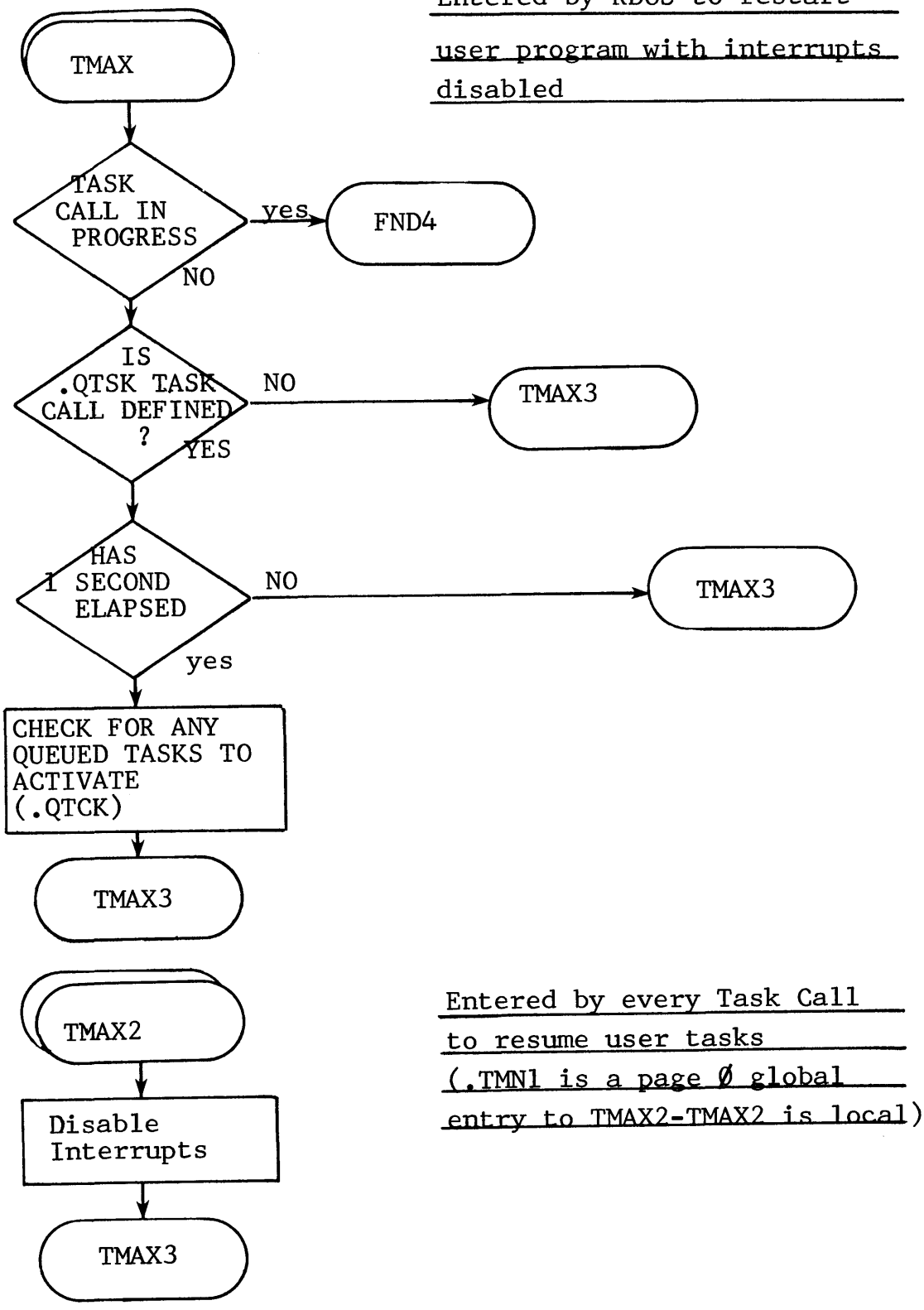
	UNMAPPED	MAPPED
"IN SCHEDULER" FLAG	BIT 0 SET TO 1 IN CURRENT TCB POINTER OF USER STATUS TABLE (1BO of USTCT)	CONTENTS OF USER MEMORY LOCATION 1 SET TO NON ZERO
CODES PASSED IN A JUMP INTO RDOS	PASSED IN ACCUMULATOR 2	PASSED IN ACCUMULATOR 0

RDOS JUMP CODES

M A P P E D	Unmapped AC >14 ⇒	Contains TCB address for system call
	Unmapped AC = 0 ⇒	No ready tasks, thus ground idle
	Unmapped AC = 1 ⇒	No ready tasks, but QTASK outstanding Return to scheduler once every second
	AC = 2 ⇒	Returning from user clock routine
	AC = 3 ⇒	Returning from user interrupt routine
	AC = 4 ⇒	Returning from user power fail routine
	AC = 5 ⇒	Request to disable interrupts
	AC = 6 ⇒	Request to re-enable interrupts and execute a user task defined by USTCT
	AC = 7 ⇒	Request to re-enable interrupts
	AC = 10 ⇒	Set map request
	AC = 11 ⇒	Remap call

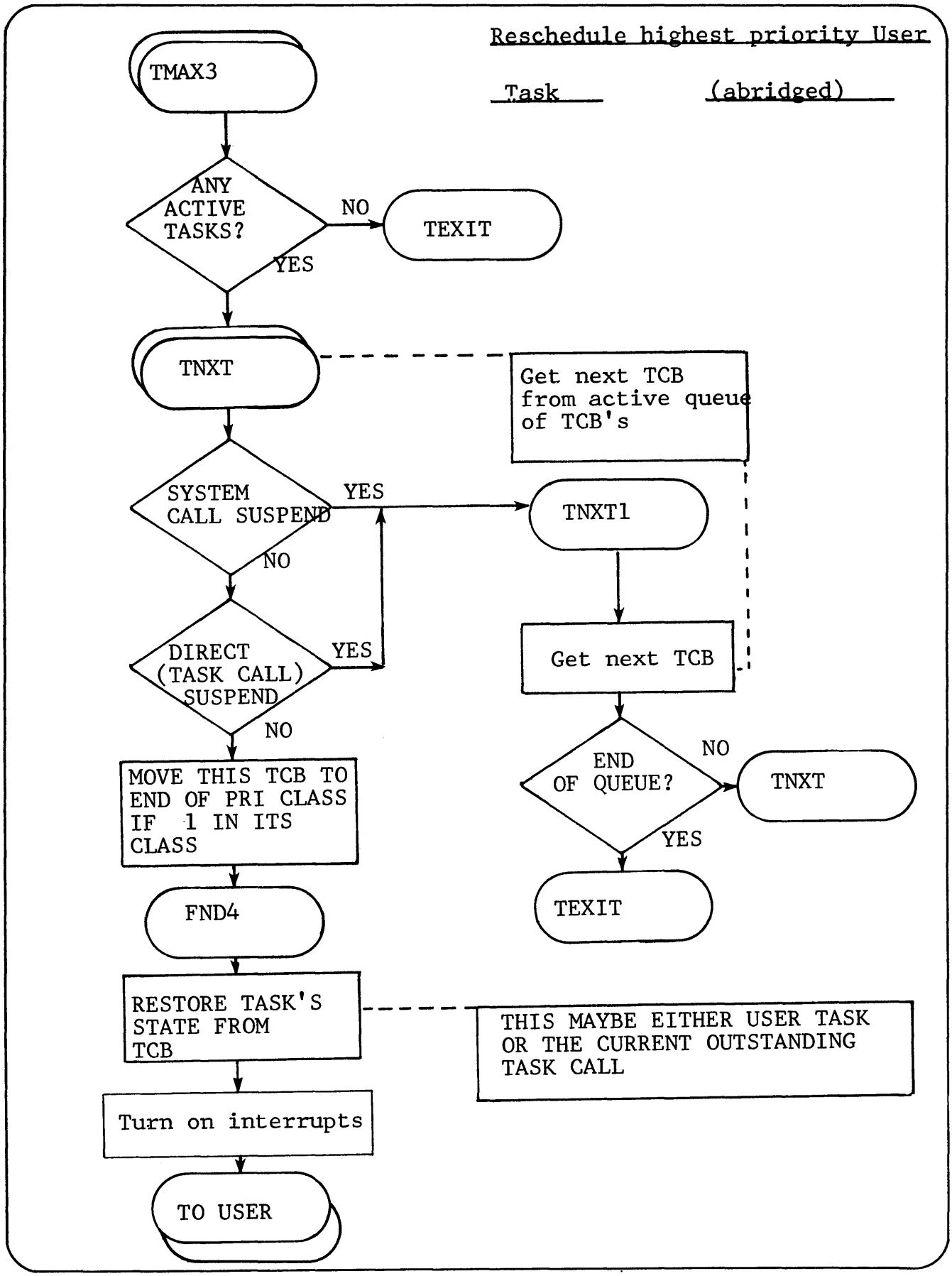
ARDOS	AC = 12, 13, 14 =	LEF	ENABLE, DISABLE, STATE

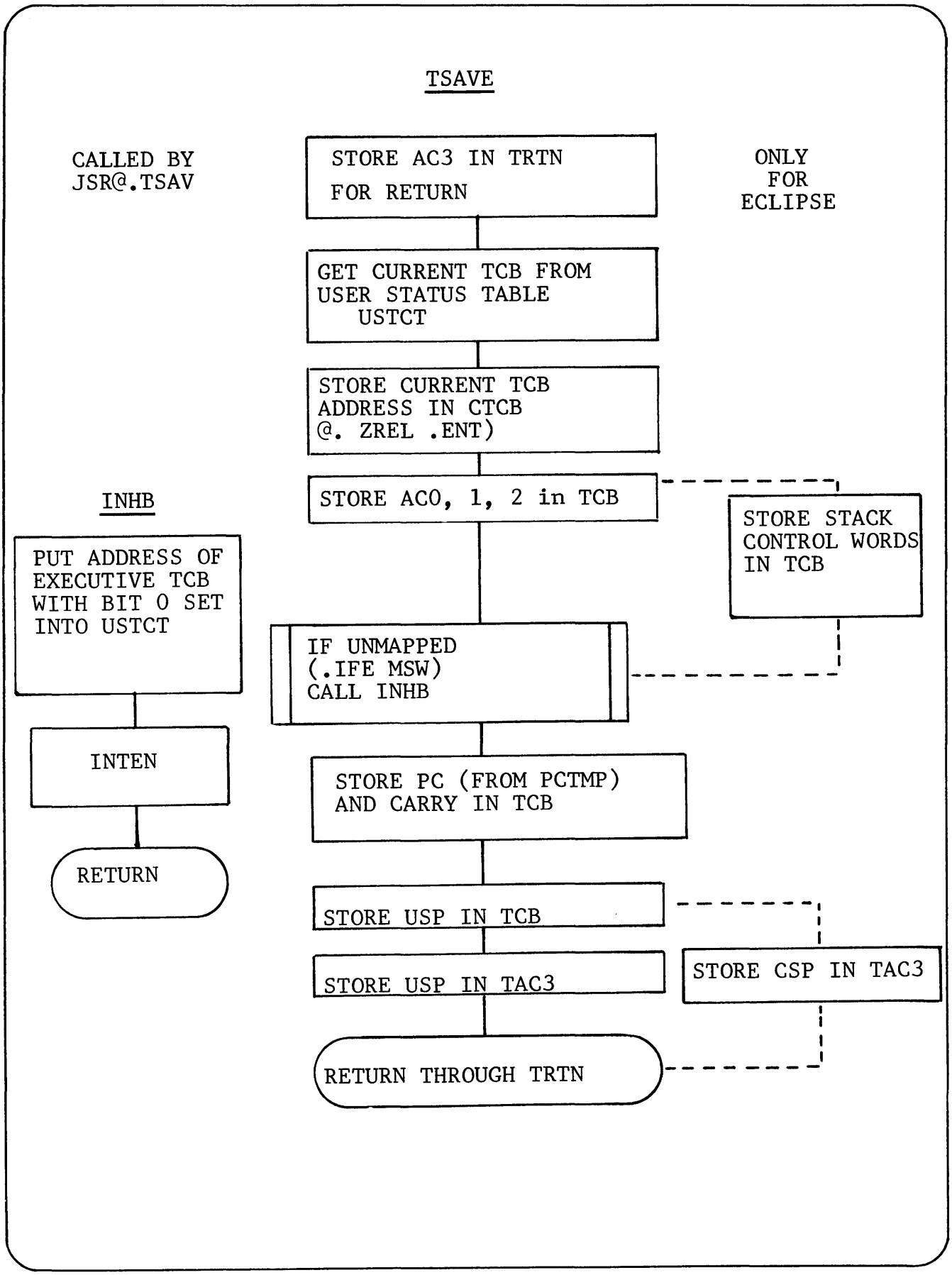
USER TASK SCHEDULER

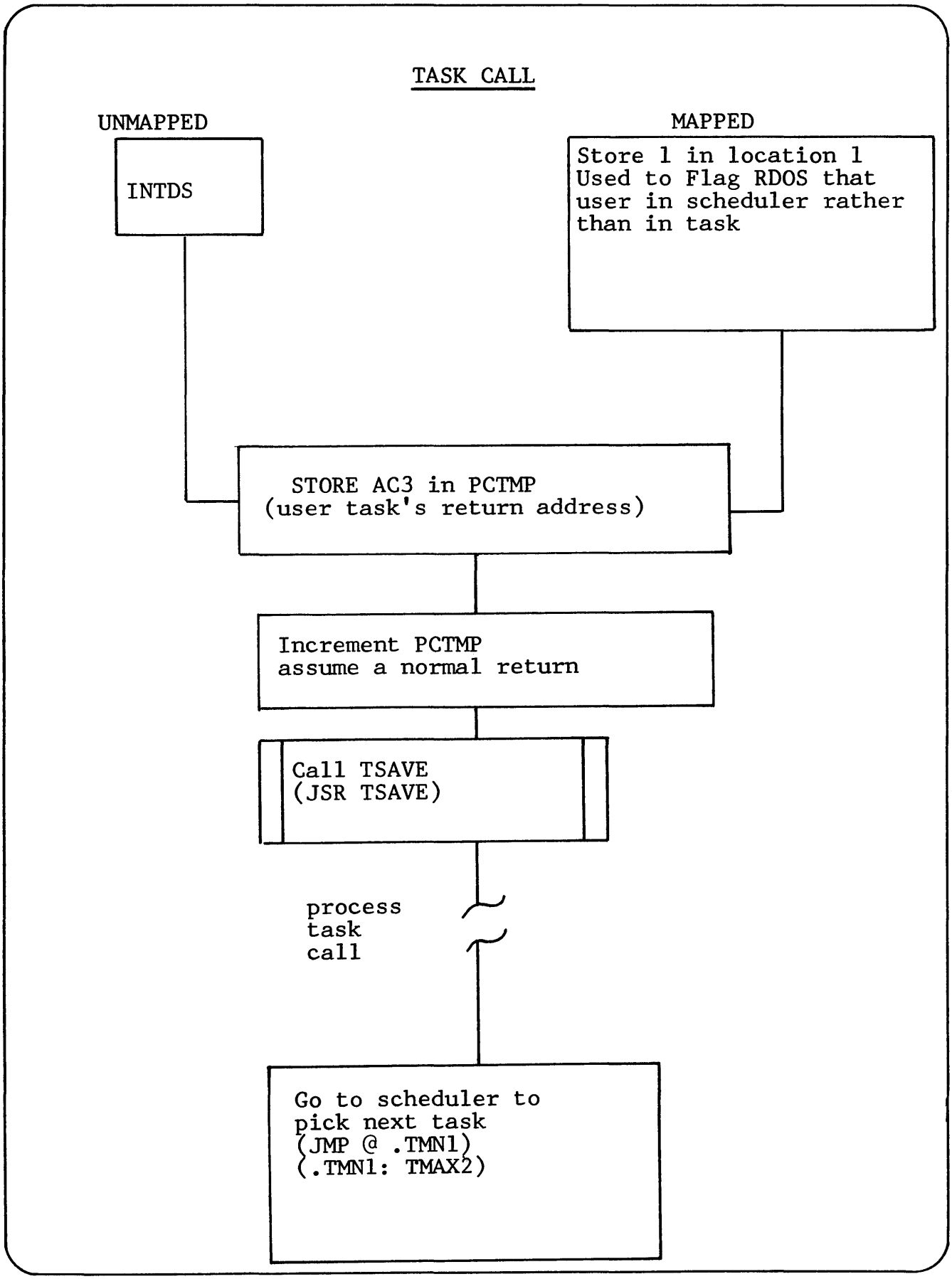


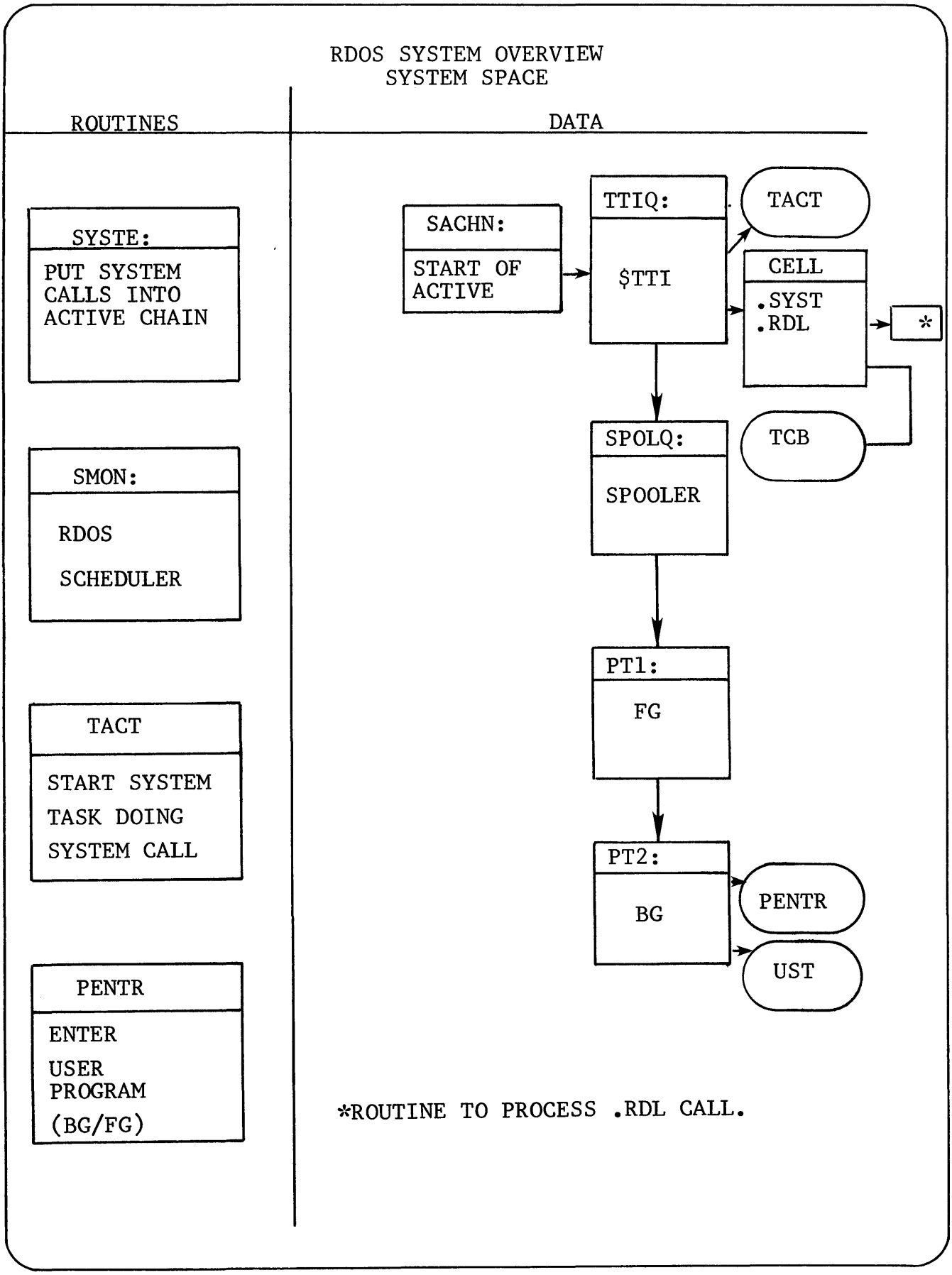
Entered by RDOS to restart user program with interrupts disabled

Entered by every Task Call to resume user tasks (.TMN1 is a page Ø global entry to TMAX2-TMAX2 is local)









RDOS SYSTEM OVERVIEW

USER SPACE

ROUTINES

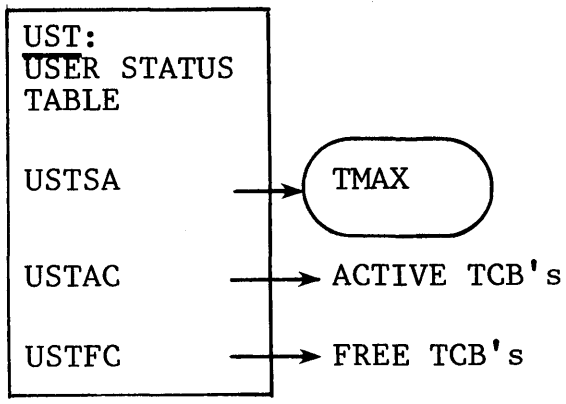
TMAX:

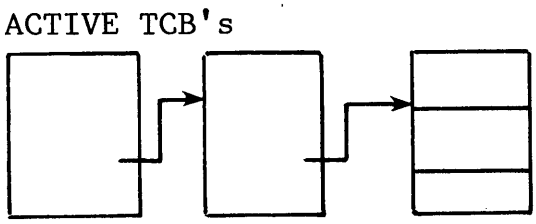
FIND HIGHEST PRIORITY USER TASK AND RUN IT.

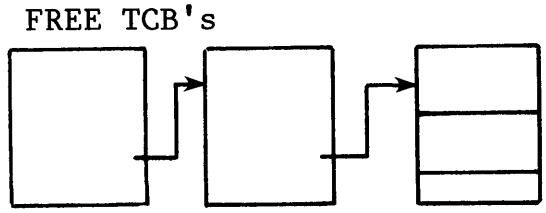
SYST:

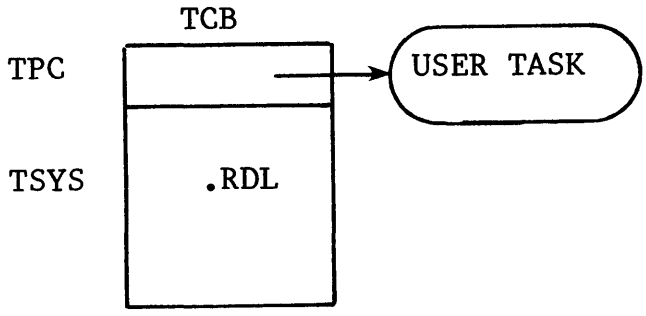
PUT SYSTEM CALL PARA-METERS IN TCB AND PASS TO RDOS (AT SYSTE

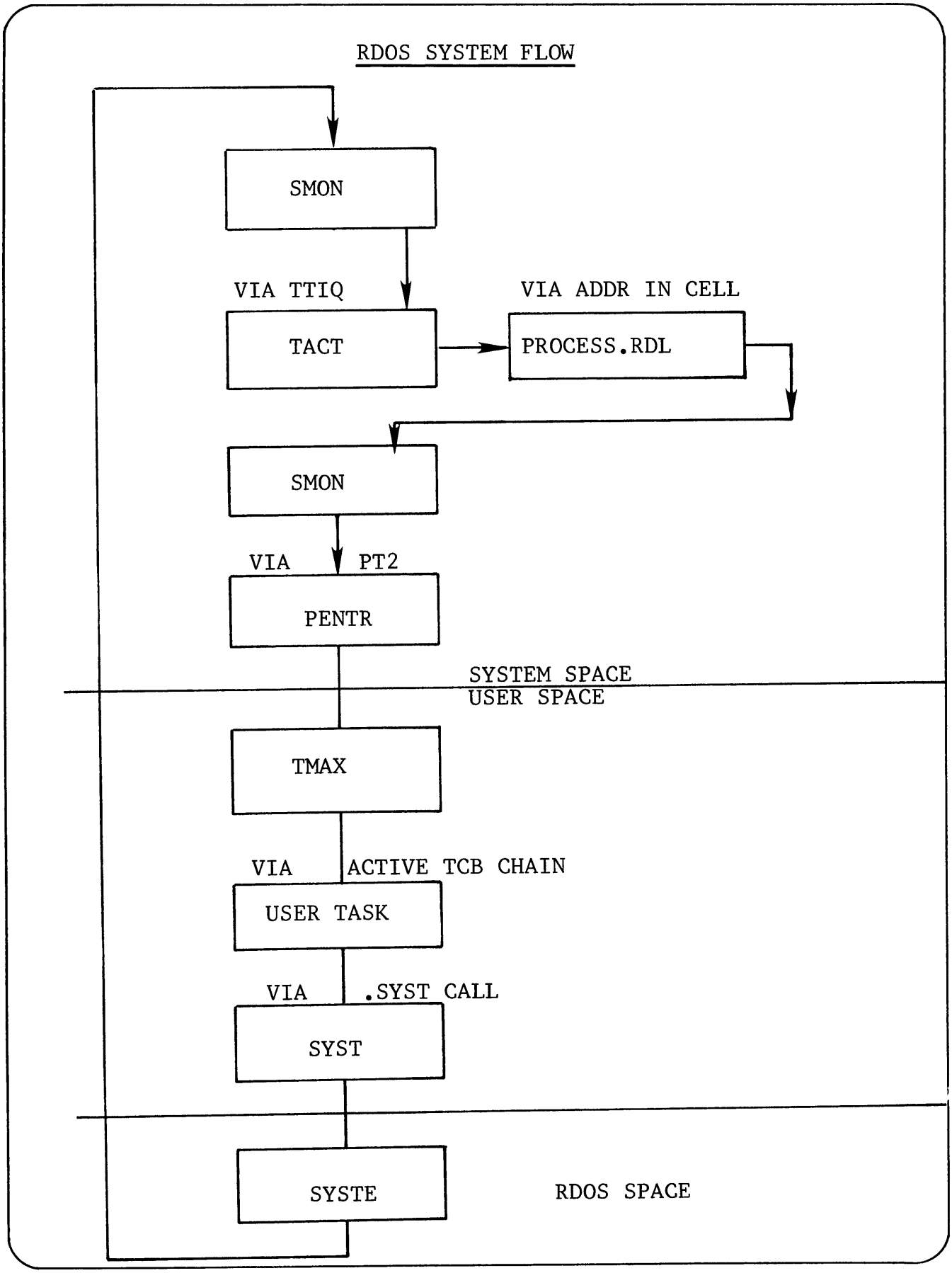
DATA











EFFECT OF INTERRUPTS ON SYSTEM FLOW

1. A SIGNIFICANT INTERRUPT SUCH AS I/O COMPLETION (AS OPPOSED TO ANOTHER TTY CHARACTER OUTPUT OR RTC TICK) WILL CAUSE RESCHEDULE FLAG TO BE SET (RESCH).

2. IF A SYSTEM TASK HAS BEEN FREED (AS FROM AWAITING A RESOURCE OR I/O COMPLETION) THE SYSTEM WORK COUNT WILL BE INCREMENTED (UPQUE).

IF NEITHER FLAG (RESCH/UPQUE) IS SET THE INTERRUPT WILL BE TRANSPARENT TO WHOMEVER IS INTERRUPTED.

IF EITHER FLAG IS SET AND THE USER IS INTERRUPTED THEN THE USER WILL BE PRE-EMPTED AND CONTROL PASSED TO SMON.

IF THE INTERRUPTED ROUTINE IS IN THE SYSTEM OR THE INTERRUPT SERVICE ROUTINES, THE INTERRUPT IS TRANSPARENT.

BEFORE THE USER IS RUN BY PENTR, THE RESCH AND UPQUE FLAGS ARE CHECKED WHICH SET CAUSES ANOTHER PASS THRU SMON.

notes

notes

Section 9

DCB'S

&

SYSTEM FILE TABLE

RECOGNIZING FILENAMES

ALL USER I/O

THROUGH CHANNELS

ASSOCIATED WITH FILENAMES

BY OPEN

OPEN A CHANNEL

MEANS

FILL IN A USER FILE TABLE

D C T

CONTAINS THE INFORMATION

TO PUT INTO UFT

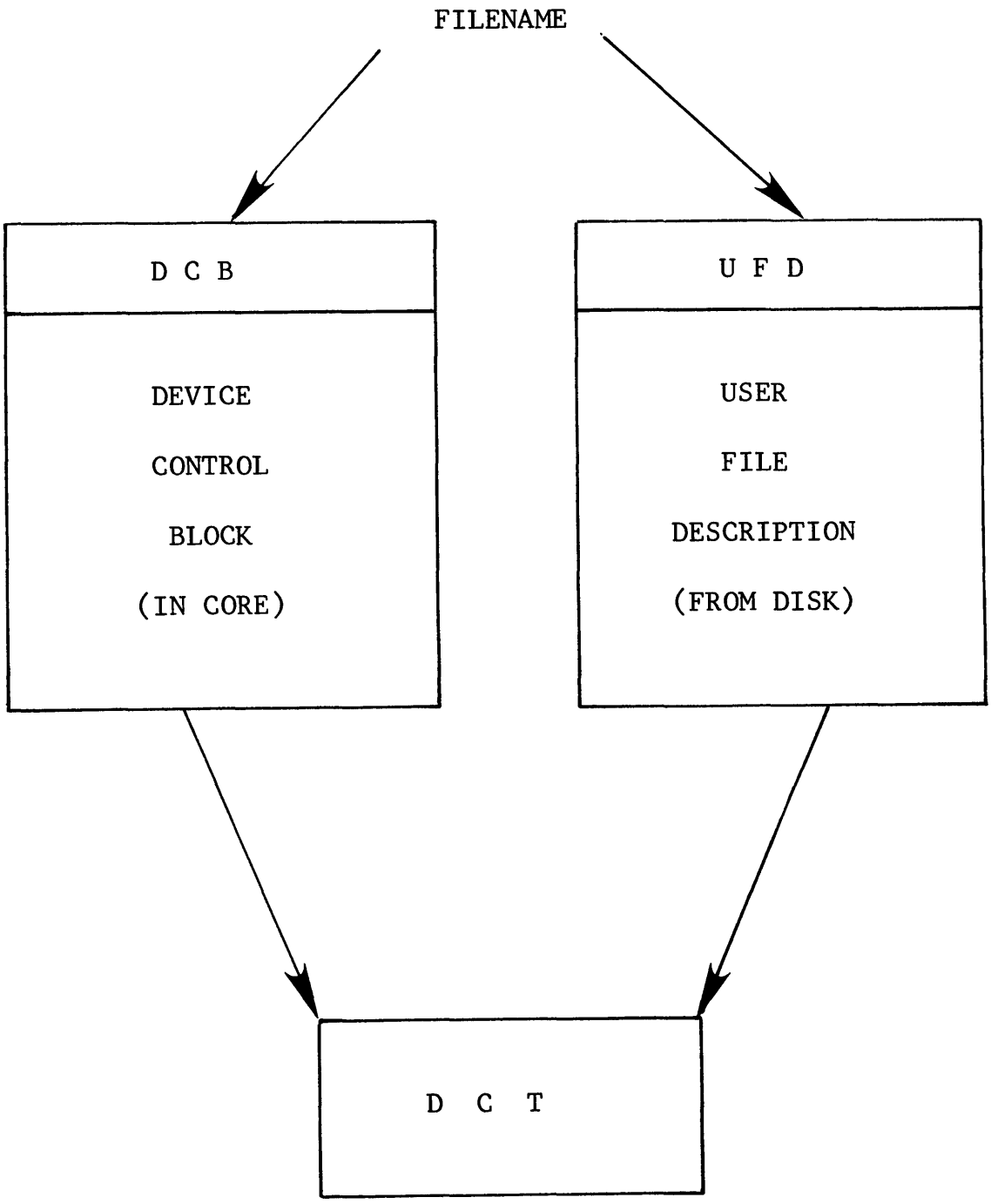
THEREFORE

SYSTEM TASK CODE MUST

LINK FILENAME

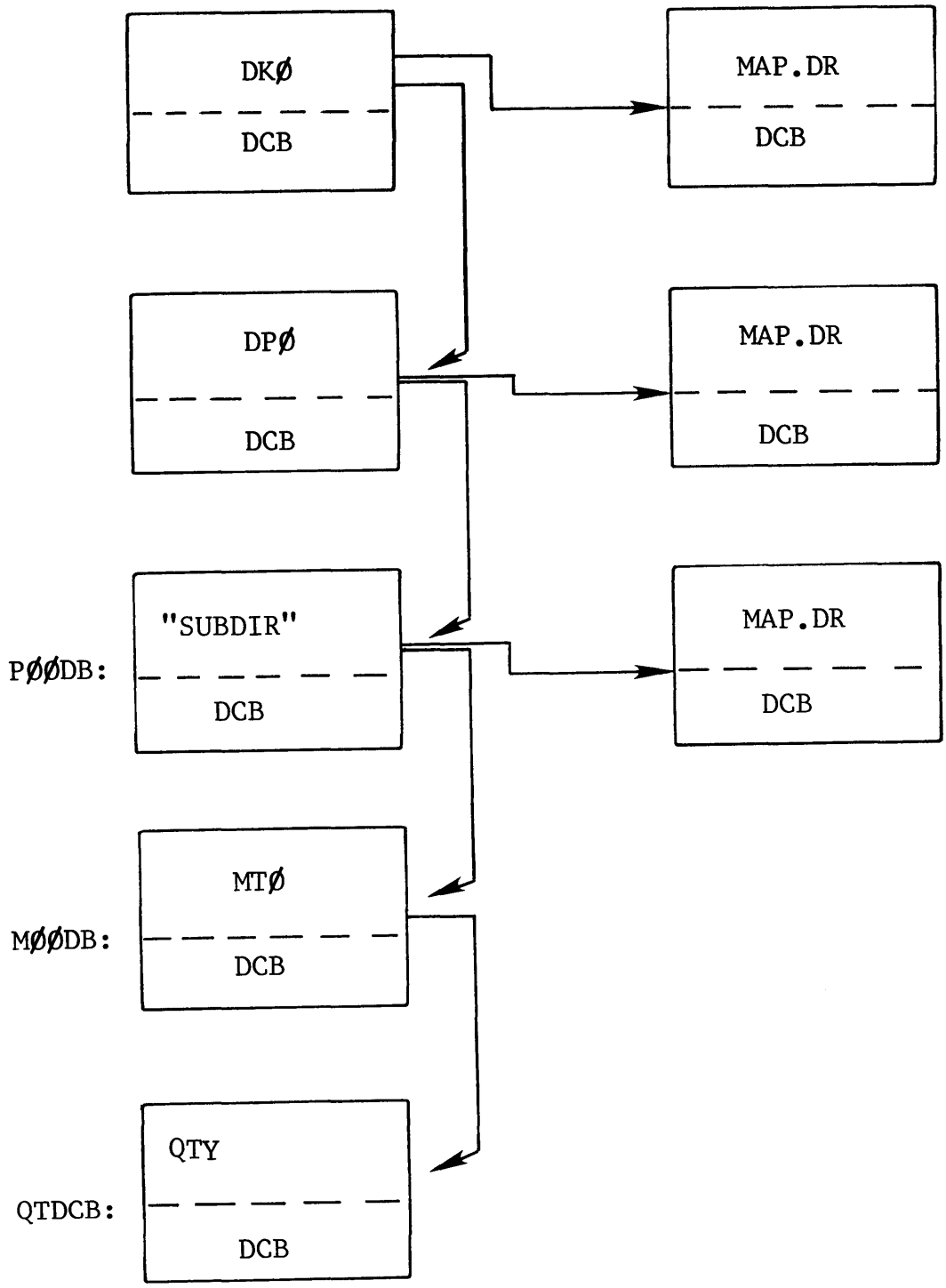
TO DCT

PATHS TO A DCT



DEVICE CONTROL BLOCK CHAIN

(AN EXAMPLE)

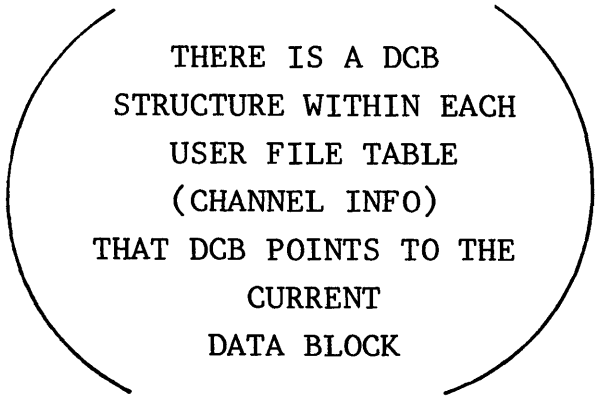


SYSTEM FILE DCBs

D C B =

1 BLOCK POINTER TO FIRST BLOCK
OF SYS.DR & MAP.DR
2 UNIT INDICATOR FOR OTHER
MULTIPLE FILE DEVICES

FOR A NAME FOLLOWED BY A COLON (:)
TO BE USED IN A FILENAME,
THE NAME MUST BE ASSOCIATED WITH A
SYSTEM FILE DCB



located in TABLE.SR	DIRR contains the start of the chain of system DCB's
	MDCB contains the address of the DCB of the master directory. This is the directory specified at bootstrap time.

DEVICE CONTROL BLOCK

15/16	FIRST LOGICAL BLOCK	DBFA1/DCBFA
14	SYS.DR DCB POINTER	DCBDR
12/13	LAST ADDRESS (SEQUENTIAL)	DBLA1/DCBLA
1Ø/11	NEXT ADDRESS (SEQUENTIAL)	DBNA1/DCBNA
7	PAD SPACE	DCBPD
6	USER COUNT	DCBUC
5	STATUS WORD	DCBST
4	CURRENT BLOCK NUMBER	DCBCB
2/3	CURRENT LOGICAL BLOCK	DBCA1/DCBCA
1	UNIT NUMBER	DCBUN
0	ADDRESS OF DCT	DCBDC

DEVICE CONTROL BLOCK

DEVICE CONTROL BLOCK STRUCTURE

Word Ø,	DCBDC	Address of the DCT of the device associated with the DCB.
Word 1,	DCBUN	Specific unit number of the device (as there is only one DCT per controller)
Word 2,	DBCA1 (high order)	Current block address (logical block address of that portion of the file which was most recently accessed).
3,	DCBCA (low order)	

NOTE The above four words form a unique identifier for any disk block.

Word 4,	DCBCB	Current block number (i.e., relative logical block number of the block within the file).
Word 5,	DCBST	Status word. The status indications are described later.
Word 6,	DCBUC	If the DCB refers to a system DCB or a directory, this word indicates whether the foreground or background initialized the directory, and how many files within the directory are open.
Word 7,	DCBPD	Not used here
Word 1Ø,	DBNA1 (high order)	The next logical block address of the file (applicable to sequential and random files).
11,	DCBNA (low order)	
Word 12,	DBLA1 (high order)	The previous logical block address of the file (applicable to sequential and random files).
13,	DCBLA (low order)	
Word 14,	DCBDR	Address of the parent directory (SYS.DR) DCB's address. If the file is the primary partition's SYS.DR DCB, then it points to itself.
Word 15,	DBFA1 (high order)	The first logical block address of the file or the directory. If the file is organized randomly it is the first random index block.
16,	DCBFA (low order)	

SYSTEM FILE DCB OFFSETS

0	DCB ENTRY	SFDCB
-1	BYTES IN LAST BLOCK	SFBC
-2	# LAST LOGICAL BLOCK	SFBK
-3	NEXT ENTRY IN CHAIN	SFNX
-4	MAP.DR LINK (-1=NOT DISK)	SFLK
-5	DEVICE KEY (SECOND WORD)	SFKY1
-6	DEVICE KEY (FIRST WORD)	SFKEY
-7 -10 -11 -12 -13	LOGICAL NAME	SFLNA
-14	DIRECTORY FRAME SIZE	SFMSZ

SYSTEM FILE DCB OFFSETS

	SFDCB	DCB ENTRY
Word -1,	SFBC	The number of bytes in the last block of the file.
Word -2,	SFBK	The number of the last logical block in the file.
Word -3,	SFNX	A link to the next entry in the system DCB chain. (If this DCB refers to a MAP.DR then it is used to interlock MAP.DR when withdrawals of block(s) are being made).
Word -4,	SFLK	If not disk = -1 If disk DCB, points to MAP.DR DCB. If a MAP.DR DCB used to indicate SYS.DR type. 0 - Primary partition 1 - Sub-partition -1 - Sub-directory
Word -5, -6,	SFKY1 SFKEY	Device key (DK, DP, MT, etc.). Used to rebuild device name after an equivalence (.EQUIV).
Word -13,	SFLNA	Logical name of system device or directory.
Word -14,	SFMSZ	Directory frame size

*NOTE on SFNX = MPLCK (Map lock for IPB systems)

Ø:	unlocked
-1:	locked by this side
1BØ:	locked by other side
Address:	pointer to cell while this side is in process of locking both sides

STATUS WORD FLAG

The bit settings in the flag word indicate the following if the bit is set.

BIT	MNEMONIC	INDICATION
1B15	STER	Error detected
1B14	STIOP	I/O in progress. Has been queued for I/O action.
1B13	STFWR	First write to file flag (this block has to go to disk)
1B12 1B11	STDIU (no name)	Directory is in use.
1BØ9	STRWD	Opened for MTA/CTA direct I/O (.MTOPD).
1BØ8	ST7T	7-track MTA flag
1BØ7	STMOD	File has been modified (written to) UFD must go to disk)
1BØ6	STTOPN	File has been transparently open
1BØ5	STEOT	Physical end of mag tape seen
1BØ3	STIOI	File initialized by IOCS
1BØ2	STOPN	DCB opened (MTA only)
1BØ1	STINI	Initialization and release not possible. Refers to system files which are not directories. Only exception is master directory which cannot be initialized.
1BØØ	STCMK	System DCB has been released (.RLSE)

FILE STATUS FLAGS

REV 6.XX

BIT	INDICATION IF SET	
15	ERROR DETECTED	STER
14	I/O IN PROGRESS	STIOP
13	FIRST WRITE FLAG	STFWR
12	DIRECTORY IN USE	STDIU
11	DIRECTORY IN USE	(NOT PARAMETERED)
10	NO CHATR, CHLAT or SPOS (QTY & MCA)	STUTP
9	OPENED FOR MTA DIRECT I/O	STRWD
8	7 TRACK MTA	ST7T
7	FILE HAS BEEN MODIFIED (WRITTEN TO)	STMOD
6	FILE TRANSPARENTLY OPENED	STTOPN
5	PHYSICAL END OF MAG TAPE	STEOT
4		
3	FILE INITIALIZED by IOCS	STIOI
2	DCB OPENED (MTA)	STOPN
1	NO INIT OR RELEASE POSSIBLE	STINI
0	DEVICE RELEASED	STCMK

SYSTEM FILE TABLE

When a directory is initialized, entries have to be made in it for all peripheral devices. All peripheral devices relating to the previous operating system are first deleted*. To carry out this deletion and insertion ** phase, reference has to be made to a table which contains entries for all known peripheral devices.

Each peripheral device entry in the table consists of three data locations.*** The contents of these locations are:

WORD	CONTENTS
1	Relative pointer to name string.
2	Attributes associated with file.
3	Device link (logical device code).

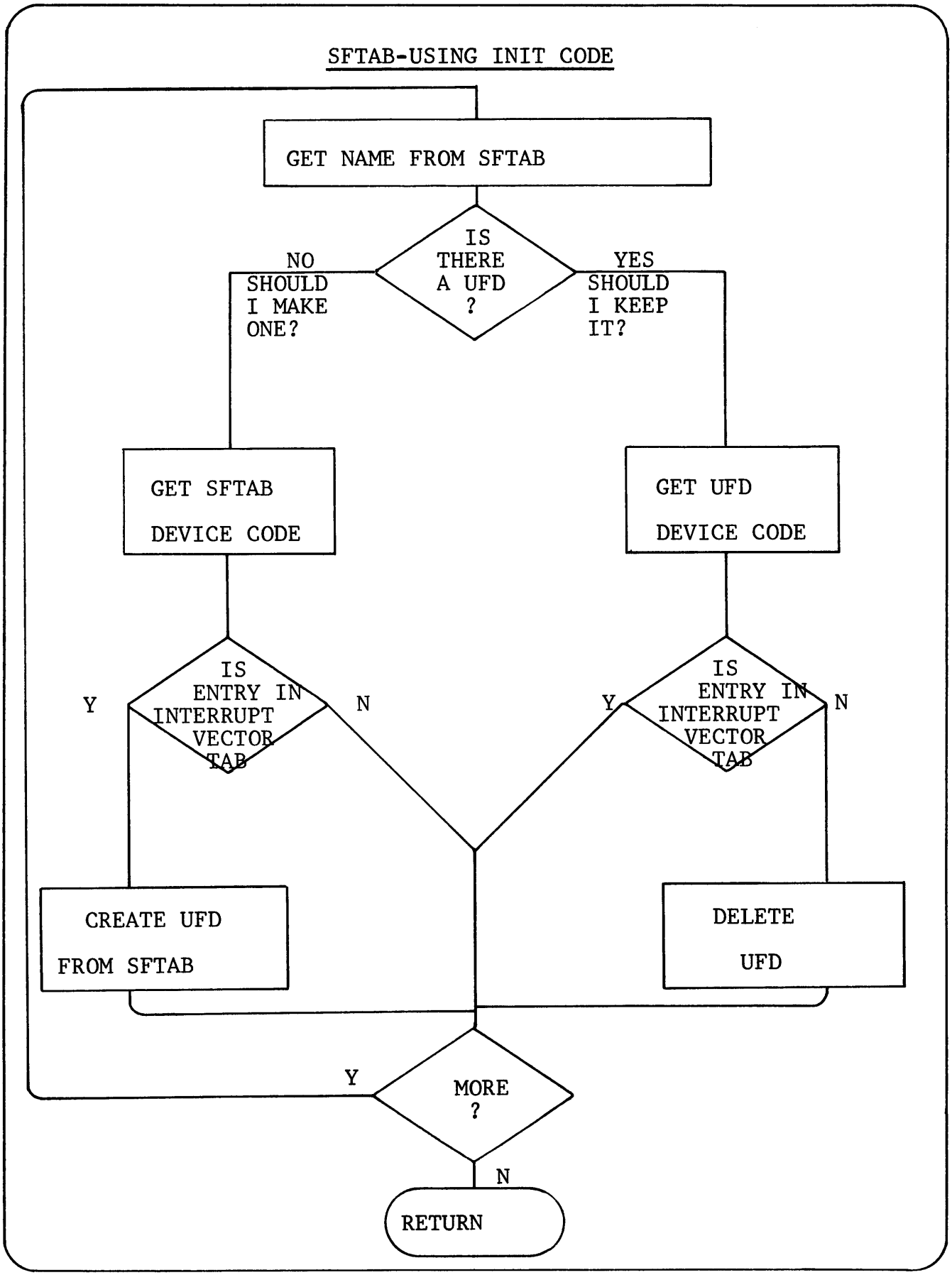
*** Must be maintained as triplets if any changes

The following is the entry for the first paper tape punch.

```
                    ---

                    ---

                    PTPP        ;*Deletions based on Filename

                    ATPER + ATCHA + ATRP

                    PTP         ;** insertions based on devices
                                   sysgened per ITBL (int vector TBL)

                    ---

                    ---

    PTPP:               .TXT/$PTP/
```

Note that the system file attributes are defined in the parameter tapes (PARU.SR). The settings are described later. The logical device code for the paper tape punch is defined in the Macro Assembler definitions.

The first entry in the table is addressed by SFTAB. The last entry in the table is followed by Ø. The whole table is located in a system overlay called SFTAB.



SYSTEM FILE TABLE

```
SFTAB:          SYSP
                ATPER+ATCHA+ATCON+ATWP
                Ø
                TTIP
                —
                —
                —
                —
                —
                PLT1P
                ATPER+ATCHA+ATRP
                PLT1
                Ø

SYSP:           .TXT/BOOTSYS.OL/
TTIP:           .TXT/$TTI/
PLT1P:          .TXT/$PLT1/
                —
                —
                —
                —
```

FILE ATTRIBUTES & CHARACTERISTICS

BIT	INDICATION IF SET	
15	WRITE PROTECTED	ATWP
14	PERMANENT FILE	ATPER
13	RANDOM FILE	ATRAN
12	CONTIGUOUS FILE	ATCON
11		
10	USER ATTRIBUTE #2	ATUS2
9	USER ATTRIBUTE #1	ATUS1
8	DIRECT BLOCK I/O ONLY	ATDIO
7	NO LINK ALLOWED	ATNRS
6	RESOLUTION ENTRY (Temporary)	ATRES
5	DIRECTORY	ATDIR
4	PARTITION	ATPAR
3	LINK ENTRY	ATLMK
2	SAVED FILE	ATSAV
1	ATTRIBUTE PROTECTED	ATCHA
Ø	READ PROTECTED	ATRP

notes

notes

Section 10

USER

FILE TABLES

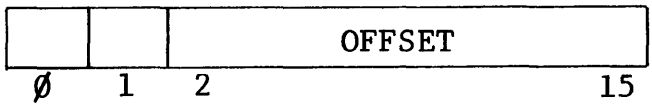
(UFT'S)

USER FILE TABLES

User file tables (UFT's) are directly equivalenced to the channels used within a RDOS user program. There is one UFT for each channel used within the program. The UFT's are used to hold current information on the channels in use by the user program.

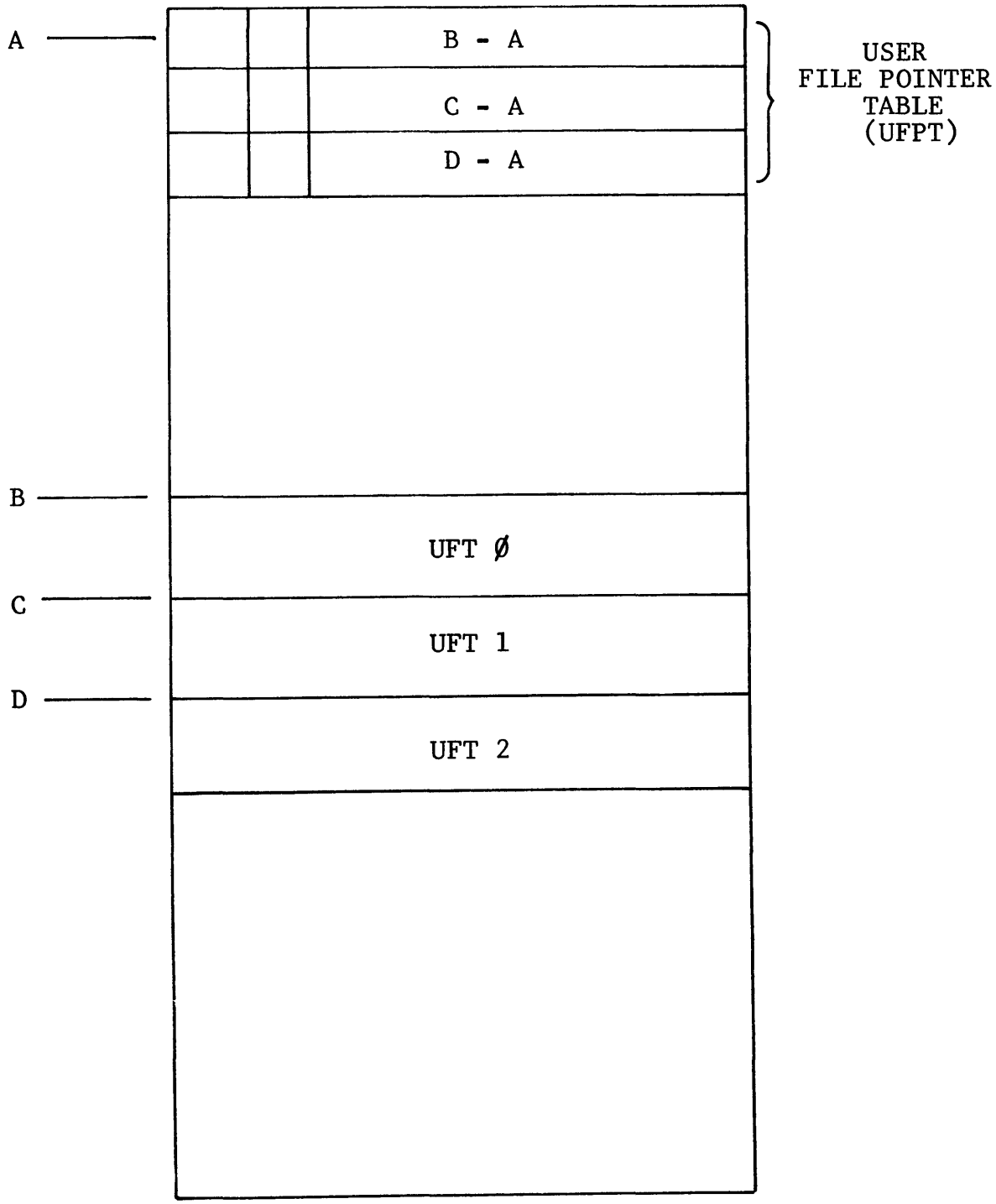
As the UFT's are directly related to the user program, the UFT area of memory is constructed when the user program is loaded. The UFT area is adacent to the memory resident portion of the operating system. For a mapped system the UFT area is reserved at INITialization area for all the UFT's and the related User File Pointer Table (UFPT) This table has an entry corresponding to each UFT. The entry contains the offset of the UFT from the base of the UFPT. It also contains flags to indicate whether that particular UFT (i.e., RDOS channel number) is in use or not.

USER FILE POINTER TABLE ENTRY



1BØ = Ø FILE OPEN
1BØ = 1 FILE CLOSED
1B1 = 1 FILE BEING OPENED

USER FILE POINTER TABLE



USER FILE TABLE STRUCTURE

A UFT can be divided into three sections:

1) User File Description (UFD)
2) Device Control Block (DCB)
3) Device Control Block Extension

USER FILE DESCRIPTION OF UFT

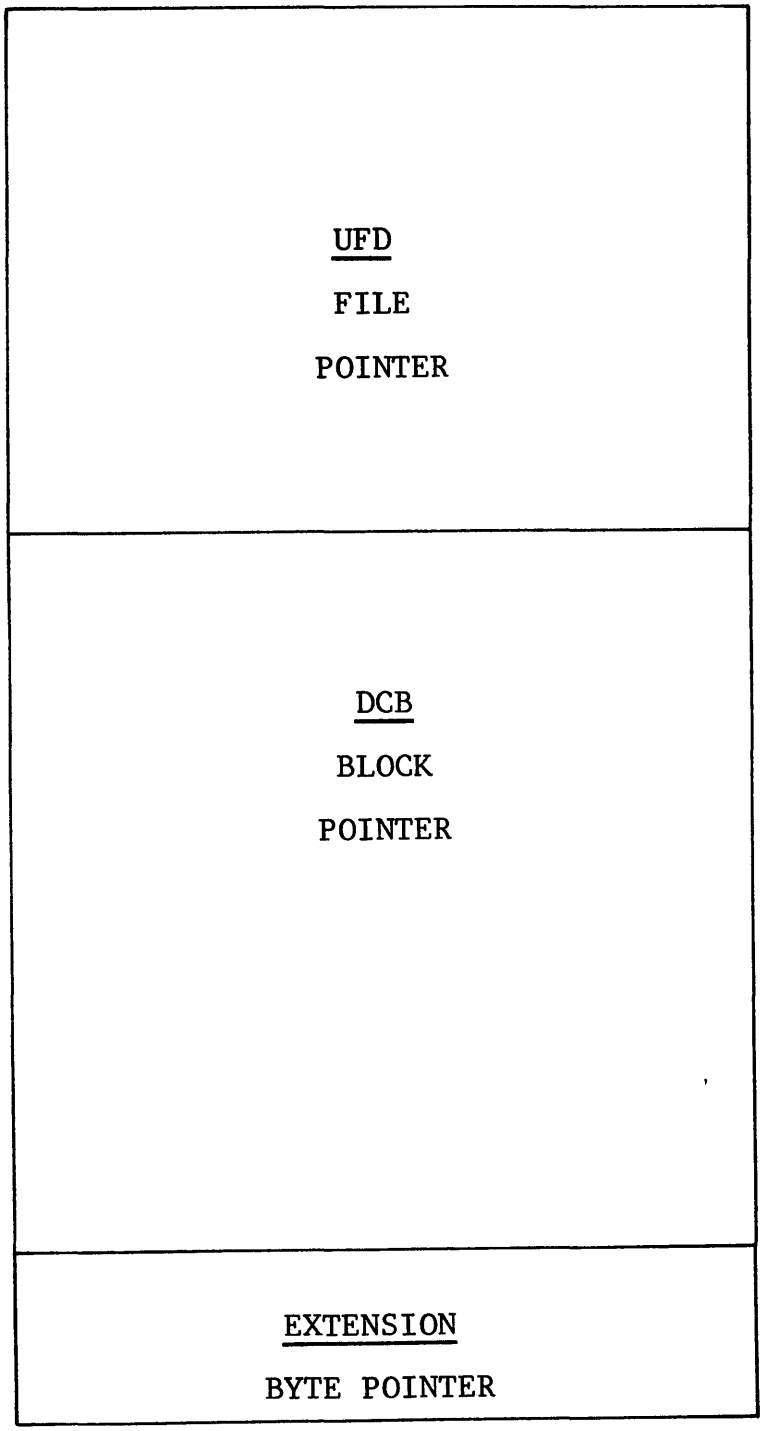
This is essentially a copy of the files UFD in the system directory (SYS.DR) on the disk. The main difference is that the UFD in the UFT is dynamic, so that some of the entries will change during the use of the UFT.

(i.e., UFTP1, UFTP2, UFTUC)

DCB

This copy of the DCB is fully used (unlike the System File DCBs). This DCB is dynamic and constantly points to the current block of data.

UFT - USER FILE TABLE



USER FILE DESCRIPTION STRUCTURE

Each User File Description (UFD) consists of 22_8 words. The structure of the UFD depends on whether it refers to a resolution file or a link file. For both, words Ø - 6 are identical.

Word Ø - 4, UFTEN Filename, ASCII characters packed from left to right, Ø through 4.

i.e. A file called DATA1 would be stored as

0	<D>	<A>
1	<T>	<A>
2	<1>	<null>
3	<null>	<null>
4	<null>	<null>

Word 5, UFTEX File Extension. Two ASCII character extension packed left to right.

i.e., A file called DATA1.SV would have as its extension

5	S	V

Word 6, UFTAT Characteristics/Attributes

UFD PORTION OF UFT

RESOLUTION FILE

0 1 2 3 4	FILENAME	UFTFN
5	EXTENSION	UFTEX
6	FILE ATTRIBUTES	UFTAT
7	LINK ATTRIBUTES	UFTLK
10	LOGICAL # OF LAST BLOCK	UFTBK
11	# BYTES IN LAST BLOCK	UFTBC
12	DEVICE ADDRESS FIRST BLOCK	UFTAD
13	YEAR/DAY LAST ACCESSED	UFTAC
14	YEAR/DAY CREATED	UFTYD
15	HOUR/MINUTE CREATED	UFTHM
16	UFD TEMPORARY	UFTP1
17	UFD TEMPORARY	UFTP2
20	USER COUNT	UFTUC
21	DCT LINK	UFTDL

COPY OF SYS.DR ENTRY FOR FILE.

USER FILE DESCRIPTION FOR RESOLUTION FILE (WORDS 7-21)

Word 7, UFTLK — Link Access Attributes. These are the attributes and characteristics which govern any user linking (.LINK) to the resolution file. The actual attributes the user will see are those formed by the logical OR of the resolution file attributes.

Word 10, UFTBK — Number of logical blocks in the file.

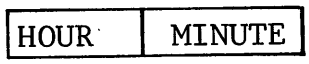
Word 11, UFTBC — Number of bytes in the last block of the file.

Word 12, UFTAD — Logical address of the first block in the file. For sequential or contiquous files this would be a data block. For random files this would be the first random index block. This is lower order of 2 word address in REV 4 & 5. High order in 21

Word 13, UFTAC — Year/Day last accessed. Number of days since January 1, 1968.

Word 14, UFTYD — Year/Day created. Number of days since January 1, 1968.

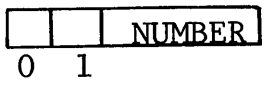
Word 15, UFTHM — Hour/Minute created



Word 16, UFTP1 — System temporary storage word

Word 17, UFTP2 — System temporary storage word

Word 20, UFTUC — For disk files used to record the number of all types of "opens" on the file.



1BØ : .EOPEN in operation
1B1 : .OPEN in operation
ØBØ AND ØB1 : .ROPEN

Word 21, UFTDL — DCT link. The last seven bits contain the logical device code of the device to which the file belongs. Left 3 bits is high order of 2 word first block address for REV 4 to 6.

USER FILE ATTRIBUTES/CHARACTERISTICS

* = For REV 4.XX, if these 3 bits are all set, they indicate that this disk is a CDC (3330 type).

ATTRIBUTES	INDICATION IF BIT SET	CHARACTERISTICS
0	READ PROTECTED	
1	CHANGE ATTRIBUTE PROTECTED	
2	SAVED FILE	
	LINK ENTRY	3
	PARTITION ENTRY	4
	DIRECTORY ENTRY *	5
	LINK RESOLUTION *	6
7	NO RESOLUTION *	
	Direct I/O ONLY	8
9	USER ATTRIBUTE #1	
10	USER ATTRIBUTE #2	
	CONTIGUOUS FILE	12
	RANDOM FILE	13
14	PERMANENT FILE	
15	WRITE PROTECTED	

DEVICE CONTROL BLOCK

DEVICE CONTROL BLOCK STRUCTURE

Word 22	UFTDC		Address of the DCT of the device associated with the DCB.
Word 23	UFTUN		Specific unit number of the device (as there is only one DCT per controller).
Word 24	UFCA1	(high order)	Current block address (logical block address of that portion of the file which was most recently accessed).
Word 25	UFTCA	(low order)	

NOTE The above four words form a unique identifier for any disk block.

Word 26	UFTCB		Current block number (i.e., relative logical block number of the block within the file). For index blocks in random files. Same as data block for others.
Word 27	UFTST		Status word. The status indications are described later.
Word 30	UFEA1	(high order)	If this is a DCB in a User File table then it is the logical address of the file's entry block in the system directory.
Word 31	UFTEA	(low order)	
Word 32	UFNA1	(high order)	The next logical block address of the file (applicable to sequential and random files).
Word 33	UFTNA	(low order)	
Word 34	UFLA1	(high order)	The last logical block address of the file (applicable to sequential and random files).
Word 35	UFTLA	(low order)	
Word 36	UFTDR		Address of the parent directory (SYS.DR) DCB's address. If the file is the primary partition's SYS.DR DCB, then it points to itself.
Word 37	UFFA1	(high)	The first logical block address of the file or the directory. If the file is organized sequentially or contiguously then this is the first data block. If the file is organized randomly it is the first random index block. (All directories are randomly organized).
Word 40	UFTFA	(low)	

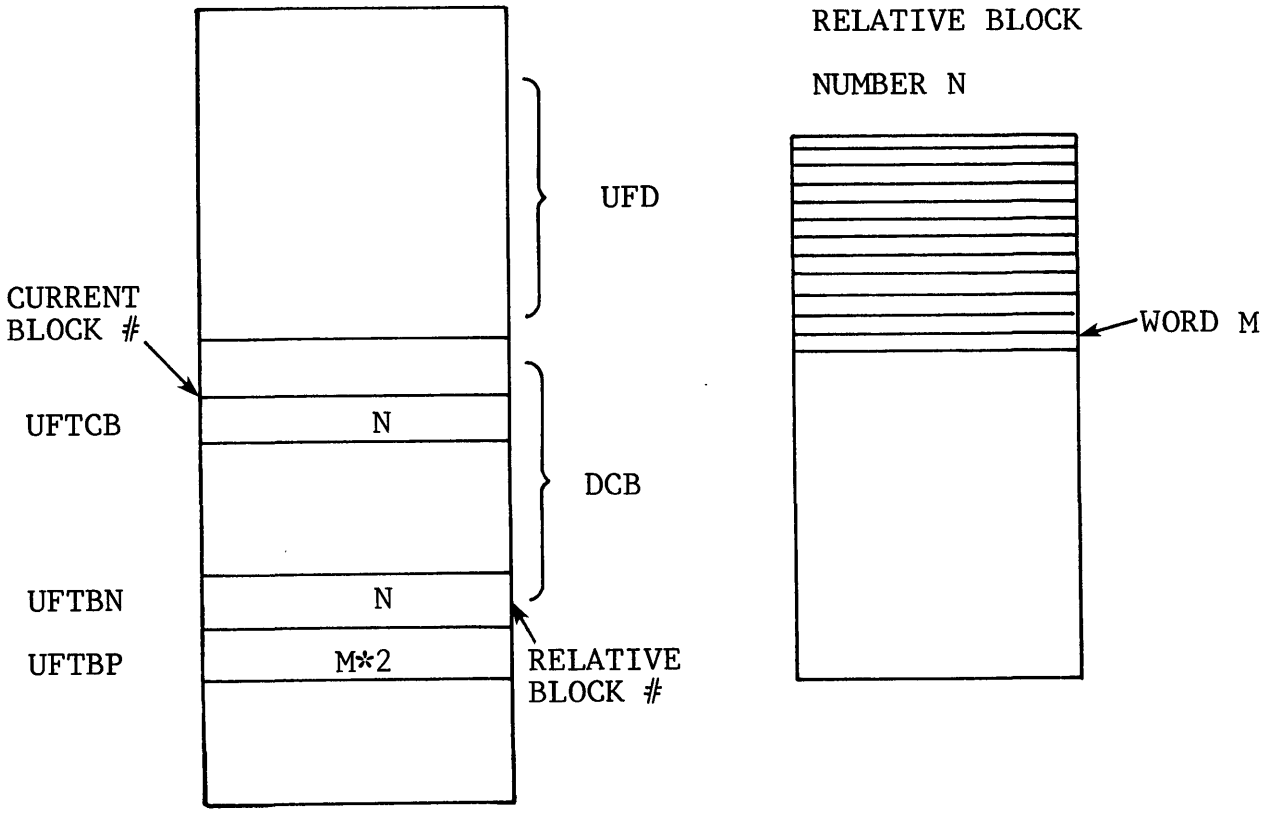
DCB PORTION OF UFT

22	DCT ADDRESS	UFTDC
23	UNIT NUMBER	UFTUN
24 25	CURRENT BLOCK ADDRESS	UFCA1 (high order) UFTCA (low order)
26	CURRENT BLOCK # OF BOOKKEEPING	UFTCB
27	FILE STATUS	UFTST
3Ø 31	ENTRY BLOCK ADDRESS	UFEA1 (high order) UFTEA (low order)
32 33	NEXT BLOCK ADDRESS	UFNA1 (high order) UFTNA (low order)
34 35	LAST BLOCK ADDRESS	UFLA1 (high order) UFTLA (low order)
36	SYS.DR DCB ADDRESS	UFTDR
37 4Ø	FIRST ADDRESS	UFFA1 (high order) UFTFA (low order)

CORRESPONDS TO SYSTEM DCB EXCEPT FOR UFTEA & UFEA1

UFT DATA HANDLING

(1) SEQUENTIAL OR CONTIGUOUS FILES

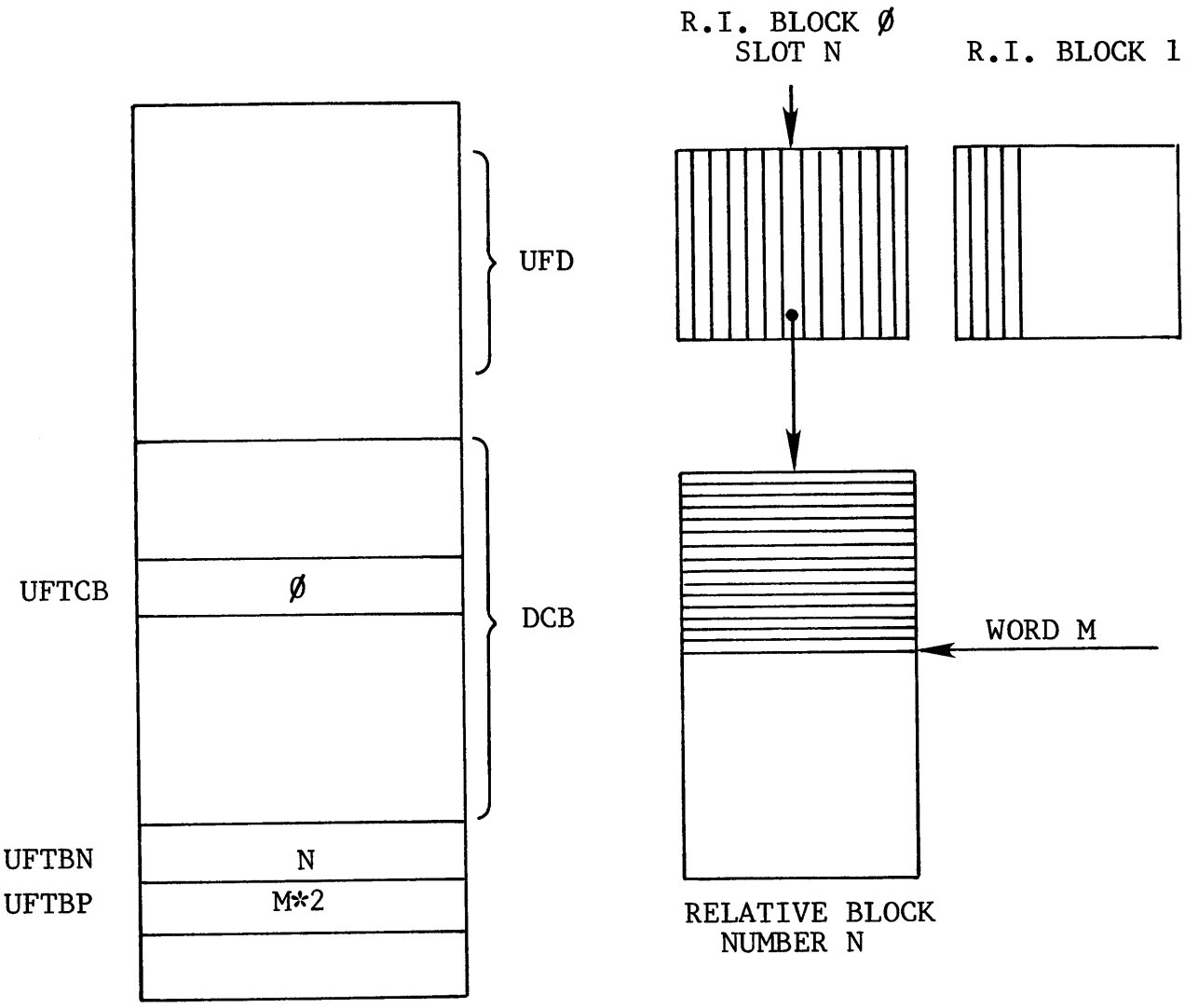


NOTE THAT THE NEXT BYTE TO BE READ FROM OR WRITTEN TO IS THE LEFT BYTE OF WORD M OF THE DATA BLOCK N. FOR SEQUENTIAL OR CONTIGUOUS FILES IT IS SEEN THAT

UFTCB = UFTBN

(2)

RANDOM FILES



NOTE THAT AS WITH CONTIGUOUS AND SEQUENTIAL FILES, THE NEXT BYTE TO BE ACCESSED IS THE LEFT BYTE OF WORD M OF THE DATA BLOCK N. BUT IN THIS CASE UFTCB IS THE RANDOM INDEX BLOCK NUMBER, AND UFTBN IS THE LOGICAL DATA BLOCK NUMBER.

FILE STATUS FLAGS

BIT	INDICATION IF SET	
15	ERROR DETECTED	STER
14	I/O IN PROGRESS	STIOP
13	FIRST WRITE FLAG	STFWR
12	BLOCK IN USE	STDIU
11	BLOCK IN USE	(NOT PARAMETERIZED)
10	NO CHATR, CHLAT or SPOS (QTY & MCA)	STUTP
9	OPENED FOR MTA DIRECT I/O	STRWD
8	7 TRACK MTA	ST7T
7	FILE HAS BEEN MODIFIED (WRITTEN TO)	STMOD
6	FILE TRANSPARENTLY OPENED	STTOPN
5	PHYSICAL END OF MAG. TAPE	STEOT
4		
3	FILE INITIALIZED BY IOCS	STIOI
2	DCB OPENED (MTA)	STOPN
1	NO INIT OR RELEASE POSSIBLE	STINI
0	DEVICE RELEASE	STCMK

STATUS WORD FLAGS

The bit settings in the flag word indicate the following if the bit is set.

BIT	MNEMONIC	INDICATION
1B15	STER	Error detected
1B14	STIOP	I/O in progress. Has been queued for I/O action.
1B13	STFWR	First write to file flag
1B12 1B11	STDIU (no name)	Block is in use. Used to lock blocks when an IPB exists in the system.
1BØ9	STRWD	Opened for MTA/CTA direct I/O (.MTOPD).
1BØ8	ST7T	7-track MTA flag
1BØ7	STMOD	File has been modified (written to)
1BØ6	STTOPN	File has been transparently opened
1BØ5	STEOT	Physical end of Mag tape seen
1BØ3	STIOI	File initialized by IOCS
1BØ2	STOPN	DCB opened (MTA only)
1BØ1	STINI	Initialization and release not possible. Refers to system files which are not directories. Only exception is master directory which cannot be initialized.
1BØØ	STCMK	System DCB has been released (.RLSE)

DEVICE CONTROL BLOCK EXTENSION OF UFT

The extension is used during program execution for handling the user file at the data level.

41 Word, UFTBN — Current relative block number of the data block presently being accessed (manipulated by .GPOS and .SPOS.)

42 Word, UFTBP — Current byte pointer to the data byte currently being accessed in the block (manipulated by .GPOS and .SPOS, and associated I/O routines).

Note: UFTBN and UFTBP can be compared against UFTBK and UFTBC in the UFD portion to determine end of file (EOF)

43 Word, UFTCH — For I/O devices, the device characteristics (refer to DCT). For disks or magnetic tape it gives the number of words per data block.

400 = random/contiguous
377 = MTA/sequential

44 Word, UFTCN — Active request count.
Indicates which device queue the UFT is on in 1BØ:

1BØ DSQ2
ØBØ DSQ1

UFT DCB EXTENSION

41	CURRENT FILE BLOCK #	UFTBN
42	CURRENT FILE BLOCK BYTE POINTER	UFTBP
43	DEVICE CHARACTERISTICS	UFTCH
44	ACTIVE REQUEST COUNT	UFTCN

notes

notes

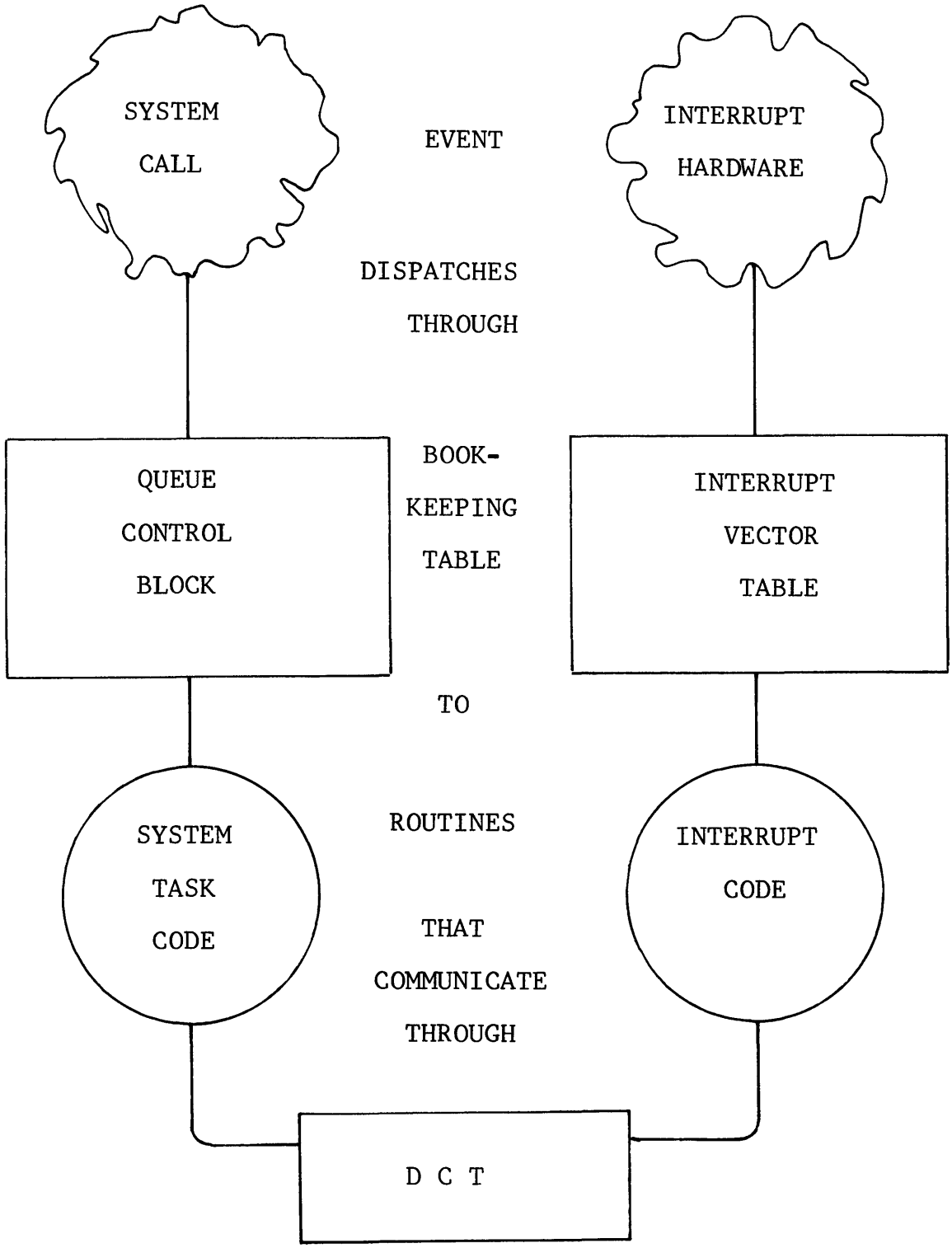
Section 11

DEVICE

CONTROL TABLES

(DCT'S)

DEVICE CONTROL TABLE



1.6 DEVICE CONTROL TABLE

A Device Control Table (DCT) contains all the information necessary to translate logical file manipulation into physical device action.

DEVICE CONTROL TABLE (DCT) STRUCTURE

Words 0-6 are common for all devices. Words 7-26 differ for Byte and Block oriented devices.

Word 0,	DCTBS:	ØBØ if device is a byte device. 1BØ if device uses data channel. (block device)
Word 1,	DCTMS:	Mask Word. The devices corresponding to the priority bits that are left cleared will be permitted to interrupt the current device.
Word 2,	DCTIS:	Address of the device interrupt service routine.
Word 3,	DCTCH:	Device characteristic word.
Word 4,	DCTCD:	Device code.
Word 5,	DCTEX:	Address of execute I/O instruction area.
Word 6,	DCTDT:	Address of the device command enable dispatch table. 1BØ for disk devices.

DEVICE CHARACTERISTICS

All I/O devices have certain characteristics in common. RDOS uses one word to define the characteristics of any device. The bit indications of the characteristics word are as follows:

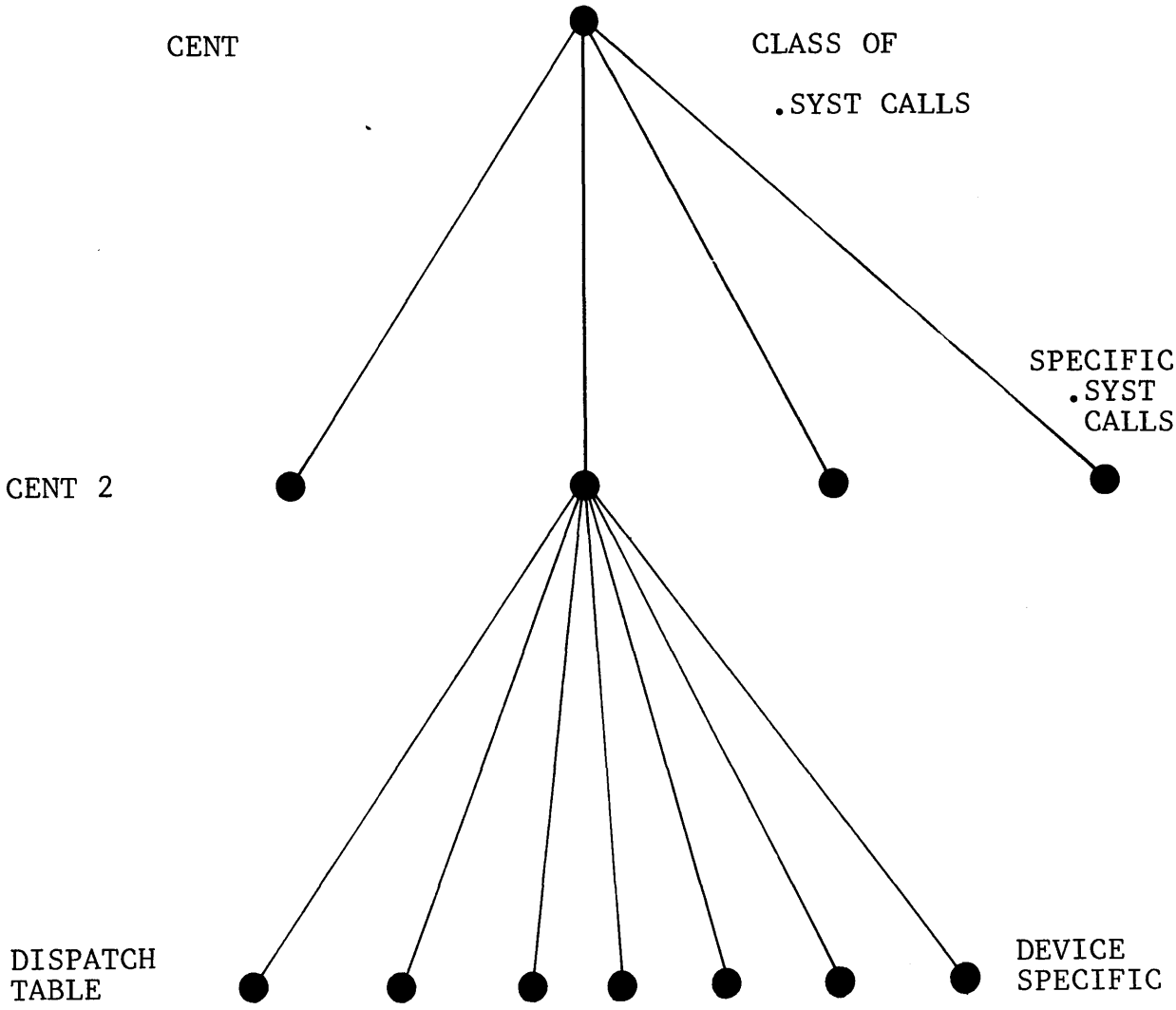
MNEMONIC	BIT	MEANING
DCCPO	15	Device requiring leader/trailer
DCSTO	15	User specified timeout constant (used with MCA)
DCCGN	14	Device requiring tab simulation
DCIDI	13	Device requiring operator intervention
DCCNF	12	Device requiring form feed simulation
DCTO	11	Teletype output device
DCKEY	10	Keyboard input device (uncontrollable)
DCNAF	9	Device requiring nulls after form feeds
DCRAT	8	Device requiring rubouts after tabs
DCPCK	7	Device requiring even parity check on input, even parity computation on output
DCLAC	6	Device requiring line feeds after carriage returns
DCSPO	5	Spoolable device (LPT, PTP, PLT, TTO)
DCFWD	4	Full word device (any size greater than a byte).
DCFFO	3	Form feed sent on .OPEN
DCLTU	2	Convert lower to upper case ASCII
DCC80	1	Read 80 columns on input if set, 72 if reset. Send 80 characters on output, 72 if reset.
DCSPC	0	Spooling enabled if set to 1, disabled if reset
DCDIO	0	Suspend protocol on transmit (used with MCA)
DCBDK	0	Disk characteristic (set indicates 333Ø type disk)

DEVICE CHARACTERISTICS

BIT	CHARACTERISTIC REQUIRED IF BIT SET	
15	LEADER-TRAILER/MCA TIMEOUT	DCCPO/DCSTO
14	TAB SIMULATION	DCCGN
13	OPERATOR INTERVENTION	DCIDI
12	FORM FEED SIMULATION	DCCNF
11	TELETYPE OUTPUT DEVICE	DCTO
10	TELETYPE INPUT DEVICE	DCKEY
9	NULLS AFTER FORM FEEDS	DCNAF
8	RUBOUTS AFTER TABS	DCRAT
7	EVEN PARITY	DCPCK
6	LF'S AFTER CR'S	DCLAC
5	SPOOLABLE	DCSPO
4	FULL WORD DEVICE	DCFWD
3	FORM FEED ON .OPEN	DCFFO
2	LOWER UPPER ASCII	DCLTU
1	80 COLUMN TRANSFERS	DCC8Ø
0	SPOOLING ENABLED/SUSPEND MCA PROTOCOL	DCSPC/DCDIO
	333Ø TYPE DISK	DCBDK

DEVICE COMMAND DISPATCH TABLE STRUCTURE

The table has an entry word for every possible RDOS I/O function. The entry is the address of the routine module which carries out that particular function. If the device does not permit a command then the entry is -1.



DEVICE COMMAND DISPATCH TABLE

0	OPEN A FILE	OF
1	CLOSE A FILE	CF
2	READ SEQUENTIAL	RS
3	READ LINE	RL
4	READ RANDOM	RR
5	WRITE SEQUENTIAL	WS
6	WRITE LINE	WL
7	WRITE RANDOM	WR
10	OPEN FOR APPENDING	OA
11	OPEN FOR READING ONLY	RO
12	EXCLUSIVE READ/WRITE OPEN	EO
13	TRANSPARENT OPEN/MTA DIRECT I/O	TO

Bypass Date&time (Creation)

DEVICE CONTROL TABLE

BYTE ORIENTED DEVICE

0	ØBØ: byte device 1BØ: block device	DCTBS
1	INTERRUPT MASK	DCTMS
2	ADDRESS INTERRUPT SERVICE CODE	DCTIS
3	DEVICE CHARACTERISTICS	DCTCH
4	DEVICE CODE	DCTCD
5	EXECUTE I/O LOCATION	DCTEX
6	ADDRESS COMMAND DISPATCH TABLE 1BØ:disks	DCTDT
7	DEVICE START UP ROUTINE	DCTST
10	DEVICE BUFFER SIZE (BYTES)	DCTBC
11	ADDRESS DEVICE BUFFER (BYTE ADDRESS)	DCTBP
12	PROGRAM BYTE COUNT	DCTPC
13	PROGRAM BYTE POINTER	DCTPP
14	LINK TO BEAD CHAIN	DCTQL
15	DEVICE BYTE POINTER	DCTDP
16	DEVICE DATA COUNT	DCTDC
17	BEAD STATUS WORD	DCTQS
20	BEAD ADDRESS (.-4)	DCTBD
21	DEVICE QUEUE START	DCTQP
22	TEMPORARY 1	DCTT1
23	TEMPORARY 2	DCTT2
24	TIME OUT/IN CONSTANT	DCTTO
25	COL. COUNT/ECHO PAIR (TTI)	DCTCC/DCTPR
26	LINE COUNT/LINK TO TTR (TTI)	DCTLC/DCTLK
27	"ON" DCB ADDRESS OF SPOOLER	DCTON
30	"OFF" DCB ADDRESS OF SPOOLER	DCTOF
31	LINK TO SPOOL DCT'S	DCTSL
32	DEVICE QUEUE S.A. (OP MESSAGES)	DCTOP
33	TEMP FOR OP MESSAGES STATUS	DCTT3

BEAD

$DPI
$CDR
$PTR

$TTR

$TTO $TTP
$DPO $PLT
$LPT $PTP

$TTI

*SAME FOR BLOCK OR BYTE DEVICE

Words 7-26,

Byte Oriented Devices

Word 7, DCTST: Address of the device start routine. The device start routine specification is as follows:

Input device: Activate the device and return
Output device: Character is passed in AC0

Word 10, DCTBC: Size of device buffer in bytes.

Word 11, DCTBP: Byte pointer to device buffer.

Word 12, DCTPC: Base level (program) byte count. Restart constant

Word 13, DCTPP: Base level (program) byte pointer.

Word 14, DCTQL: Link to device request bead chain.

Word 15, DCTDP: Device data byte pointer.

Word 16, DCTDC: Device data count.

Word 17, DCTQS: Bead status word.

Word 20, DCTBD: Bead address (i.e., the starting address of the first bead, word 14).

Word 21, DCTQP: Device bead queue starting address (initially -1).

Word 22, DCTT1: First temporary for device control.

Word 23, DCTT2: Second temporary for device control.

Word 24, DCTTO: Time out constant for input devices.

Word 25, DCTCC: Output device column counter

(Word 25, DCTPR: Echo device DCT pair pointer, TTI only.)

Word 26, DCTLC: Output device line counter.

(Word 26, DCTLK: Link to TTR, TTI DCTs only.) *

Word 27, DCTON: "ON" DCB address for spooler. *

Word 30, DCTOF: "OFF" DCB address for spooler. *

Word 31, DCTSL: Link to next spoolable device DCT (-1 terminates the chain).

Word 32, DCTOP: Device queue starting address for operator messages.

Word 33, DCTT3: Temporary for operator message status word whose bits are defined as follows:

BIT	MEANING
0	Set to 1 if "!" has been received.
15	Set if "F" or "B" has been received.

*get by RDOS.

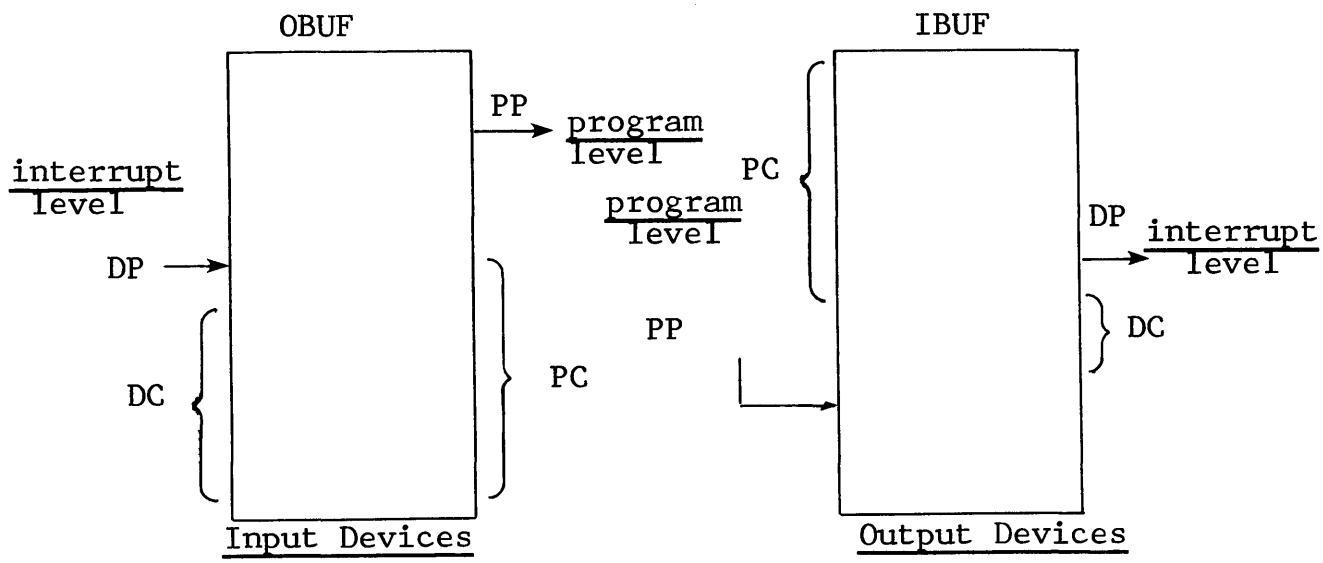
BUFFERED I/O

Byte oriented devices have in their driver modules data buffers for handling their data transfers.

The basic buffer philosophy is to maintain one or more* fixed length buffers, with pointers and counters maintained to indicate the amount of data in the buffers and the current word input or output. (A virtually unlimited number of buffers can be appended to the first buffer, with a four-word bead assigned to maintain status information and pointers for each additional buffer.)*

An input device inputs to the buffer at interrupt time and outputs from the buffer at program base level. An output device inputs to the buffer at program base level and outputs from the buffer at interrupt time.

*Only TTIDC/TTI1DC uses more than one (primarily for echoing and .RDOPR messages.)



Pointers PP and DP indicate the current slot in the buffer used for character storage or retrieval by the program and device respectively. Counters PC and DC indicate the current number of characters stored or retrieved from the buffers by the program or device halts temporarily when the last buffer has been filled. When the program or device retrieves the last buffer character, characters may once again be input to the beginning of the buffer. In the case of multiple buffers,* each buffer becomes free to receive input as soon as the last character in that buffer has been retrieved by the program or device.

When DC becomes equal to zero (after the last character in a buffer has been accepted by an output device), or when DC becomes non-zero (when a character is placed in a buffer by an input device), the task with pointer PP is readied.

One device bead consists of words 14 through 17 of the device's DCT. The structure of each bead is as follows:

Word 0	Link word
Word 1	Data Pointer (DP)
Word 2	Data Count (DC)
Word 3	Bead status/mode word

Succeeding beads, used by system tasks like echoing, are linked to by previous bead links. The first bead in the string is pointed to by DCTQP of the device's DCT. The bead status word bit settings are described on the next page.

BEAD STATUS BITS

BIT	INDICATION IF SET	
15	REQUEST COMPLETED	BSDON
14	ECHO MODE (TTI ONLY)/EOF ($CDR) Control(s) in effect on output device	BSECH/BSEOF
13	BUFFER IN LAPPED MODE	BSLDP
12	LAPPABLE BUFFER	BSLPA
11	.RDL MODE ($CDR ONLY)	BSRDL
10	IOCS CALL	BSIOC
9	COMMON SERVICE ROUTINES SHOULD NOT START THIS DEVICE ($CDR)	BSQIT
8	OPEN TO FOREGROUND	BSOFG
7	OPEN TO BACKGROUND	BSOBG
6	DEVICE OPEN/FREE BUF ON COMPLETION	BSDOP/BSFBF
5	BACKGROUND OPER MESSAGE	BSBOP
4	FOREGROUND OPER MESSAGE	BSFOP
3	BACKGROUND CONSOLE DEVICE	BSBCD
2	FOREGROUND CONSOLE DEVICE	BSFCD
1	UNPEND ON REQUEST COMPLETION	BSUPC
Ø	UNPEND AFTER ANY CHAR.	BSUPE

BUFFER SIZES FOR BYTE DEVICES

	(IN WORDS)	
TTI/TTI1	72_{10} → change?	
TTR/TTR1	5	
TTO/TTO1	32_{10}	(line)
DPI	10_{10}	
DPO	10_{10}	
PLT/PLT1	64_{10}	
CDR/CDR1	81_{10}*2	(2 cards)
PTP/PTP1	20_{10}	
LPT/LPT1	80_{10} → 132?	
PTR/PTR1	32_{10}	

TELETYPE READER / KEYBOARD

Two devices sharing same device code:

TTI/TTR	1Ø
TTI1/TTR1	5Ø

One DCT associated with each device.

Device can only be used on one device code at any one time. The ITBL entry points to the DCT currently in use.

Last entry in that DCT points to the other DCT.

i.e. TTRDCT has @TTIDC
TTIDCT has TTRDC

.OPEN on either DCT gets correct DCT and places its entry in the correct pointer in ITBC.

D C T

BLOCK DEVICE WITHOUT SEPARATE UNIT DESCRIPTORS

0	BIT Ø SET TO SHOW DATA CH.	DCTBS
1	INTERRUPT MASK	DCTMS
2	INTERRUPT SERVICE ROUTINE	DCTIS
3	DEVICE CHARACTERISTICS	DCTCH
4	DEVICE CODE	DCTCO
5	EXECUTE I/O AREA	DCTEX
6	COMMAND DISPATCH TABLE	DCTDT
7	START DEVICE ROUTINE	DCSTR
10	SET DST ROUTINE ADDRESS	DCDST
11	CURRENT BUFFER POINTER	DCCRQ
12	FIRST SLOT RESERVED IN DATA CH. MAP	DCHMP
13	NUMBER OF DATA CH. SLOTS NEEDED	DCHNM
14	POINTER TO "PARENT DCT" .-14	DCTPD
15	READ PREVIOUS BLOCK ROUTINE	DCTRL
16	ROUTINE TO TRANSFER BLOCK	DCTRD
17	READ NEXT BLOCK ROUTINE	DCTRN
20	DEVICE INIT ROUTINE	DCTIN
21	DEVICE RELEASE ROUTINE	DCTRS
22	BLOCKS PER DEVICE (HIGH ORDER)	DCNB1
23	BLOCKS PER DEVICE (LOW ORDER)	DCNBK

WORDS 7-26, BLOCK TRANSFER-DEVICES

DCTRD: Address of routine to read a block.

DCNB1: Number of blocks on device (high order)

DCNBK: Number of blocks on device. (if Ø, has not been sized. -1 for MTA/CTA)

DCSTR: Device start routine address.

DCDST: Set DST word (used for logical-to-physical disk block address computation for MHD).

DCCRQ: Current request pointer. Points to current buffer in queue, else -1.

DCTSZ: Address of routine to perform disk sizing (not used by MTA/CAS. Used when disk INITialized.

DCTRL: Address of routine to read last block.

DCTRN: Address of routine to read next block.

DCTIN: Device initialization routine address.

DCTRS: Device release routine address.

DCTNS: Number of sectors per track. (MHD)

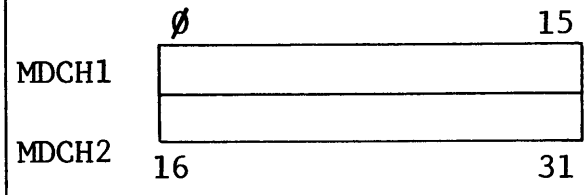
DCTNH: Number of heads per unit. (MHD)

DCTFO: Directory frame size

DCHMP: First slot in data channel map (mapped systems only).

DCHNM: Number of slots needed in data channel map (mapped systems only).

DATA CHANNEL MAP ALLOCATION TABLE (located in Table .SR)



Bit = 1 indicates slot is free and available

D C T

BLOCK DEVICE USING SEPARATE UNIT DESCRIPTORS

PARENT (CONTROLLER) DCT

0	BIT Ø SET = DATA CH. DEVICE	DCTBS
1	INTERRUPT MASK	DCTMS
2	INTERRUPT SERVICE ROUTINE	DCTIS
3		
4	DEVICE CODE	DCTCD
5	EXECUTE I/O AREA	DCTEX
6	BIT Ø SET ONLY FOR DISK	DCTDT
7	START DEVICE ROUTINE	DCSTR
10	SET DST WORD	DCDST
11	CURRENT BUFFER POINTER	DCCRQ
12	FIRST SLOT RESERVED IN DATA CH. MAP	DCHMP
13	NUMBER OF DATA CH. SLOTS NEEDED	DCHNM
14	POINTER TO "PARENT DCT" .-14	DCTPD

10 Mbyte and Floppy disks are hooked into the same controller.

For this set up, a controller level table isn't enough.

For disks only, the DCT information has been divided between a parent (controller) DCT and individual unit DCTs for each disk.

D C T

BLOCK DEVICE USING SEPARATE UNIT DESCRIPTORS

UNIT DCT

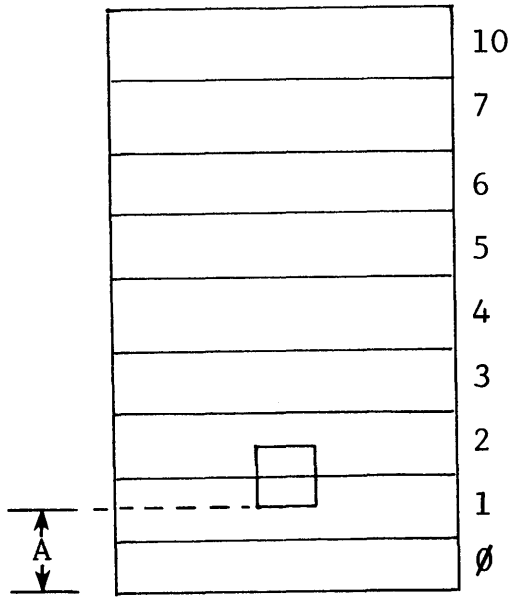
0	UNIT NUMBER		DCTUN
1	LINK TO BAD BLOCK TABLE		DCTBL
2	BAD BLOCK TABLE UPDATE FLAG		DCTBF
3	DEVICE CHARACTERISTICS		DCTCH
4	DEVICE CODE		DCTCD
5	EXECUTE I/O AREA		DCTEX
6	COMMAND DISPATCH TABLE		DCTDT
7	START DEVICE ROUTINE		DCSTR
10	SET DST WORD		DCDST
11	SECTORS PER TRACK	✓	DCTNS
12	HEADS PER UNIT	✓	DCTNH
13	DIRECTORY FRAME SIZE (HASH)	✓	DCTFD
14	POINTER TO PARENT DCT		DCTPD
15	READ PREVIOUS BLOCK ROUTINE		DCTRL
16	ROUTINE TO TRANSFER A BLOCK		DCTRD
17	READ NEXT BLOCK ROUTINE		DCTRN
20	DEVICE INIT ROUTINE		DCTIN
21	DEVICE RELEASE ROUTINE		DCTRS
22	BLOCKS PER DEVICE (HI ORDER)		DCNB1
23	BLOCKS PER DEVICE (LOW ORDER)		DCNBK

DEVICE INTERRUPT MASKS

ØØØØØ3	MKTTO	TTO
ØØØØØ3	MKTTI	TTI
ØØØØØ7	MKPTP	PTP
ØØØØ17	MKMCA	MCA
ØØØØ17	MKPLT	PLT
ØØØØ1Ø	MKLPT	LPT
ØØØØ13	MKQTY	QTY
ØØØ217	MKDPO	DPO
ØØØ217	MKDPI	DPI
ØØØ617	MKDKP	MHD
ØØØ617	MKDZP	ZEBRA MHD
ØØØ717	MKDSK	FHD
ØØ1717	MKIPB	IPB
ØØ1ØØØ	IPMKB	
ØØØ213	MKALM	LM
ØØ1737	MKPTR	PTR
ØØ1777	MKCDR	CDR
ØØ1777	MKMTA	MTA
ØØ1777	MKCAS	CAS

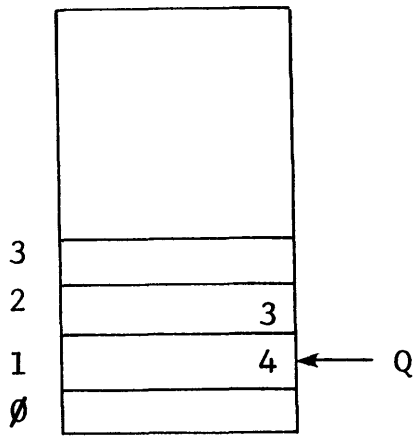
DCH TRANSFER BETWEEN DEVICE AND USER ADDRESS SPACE

LOGICAL
USER AREA



$$\frac{A}{1024_{10}} = \text{logical block} + R$$
$$Q$$

Use Q as index into user memory map (in program table) to get physical block.



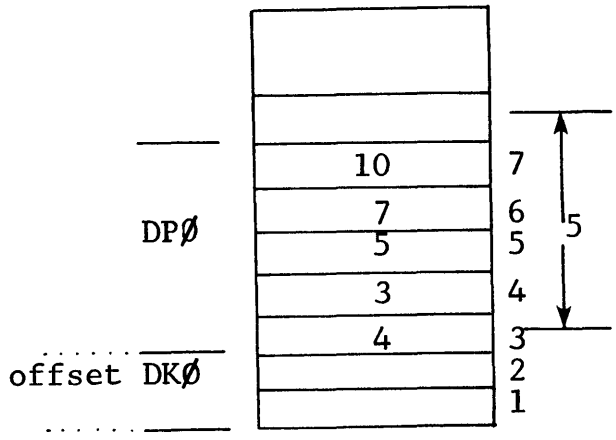
(SWAMP converts to physical)

Use offset of base of DPØ DCH slots + R

for DOB to device.

Can read data direct to users area as it knows physical blocks involved.

DCH MAP



DATA CHANNEL MAPPING

The data channel map has fixed device allocations for its 1K slots. These are decided at INIT time depending on what devices are configured into the system. The number of slots needed for system devices (dependent on maximum word transfers for the device) are as follows:

	Slots	Max words
For each FHD controller	2	256
For each MHD controller	5	4096
For each MTA controller	5	4096
For each CTA controller	5	4096
For each MCA controller transmitter	5	4096
For each MCA controller receiver	5	4096

The number of 1K slots have to allow for the occasion that the block being transferred straddles a 1K boundary.

Hence # slots = $\frac{\text{Max. word transfer}}{1024}$ +1

DATA CHANNEL TRANSFERS

There are two occasions when the DCH is used for data transfers:

(i) System requires an I/O data transfer. All data is normally transferred to/from device via a system buffer area to the user.

(ii) User requires an I/O data transfer. This only occurs with direct I/O transfer between disk or MTA/CTA and the user's area.

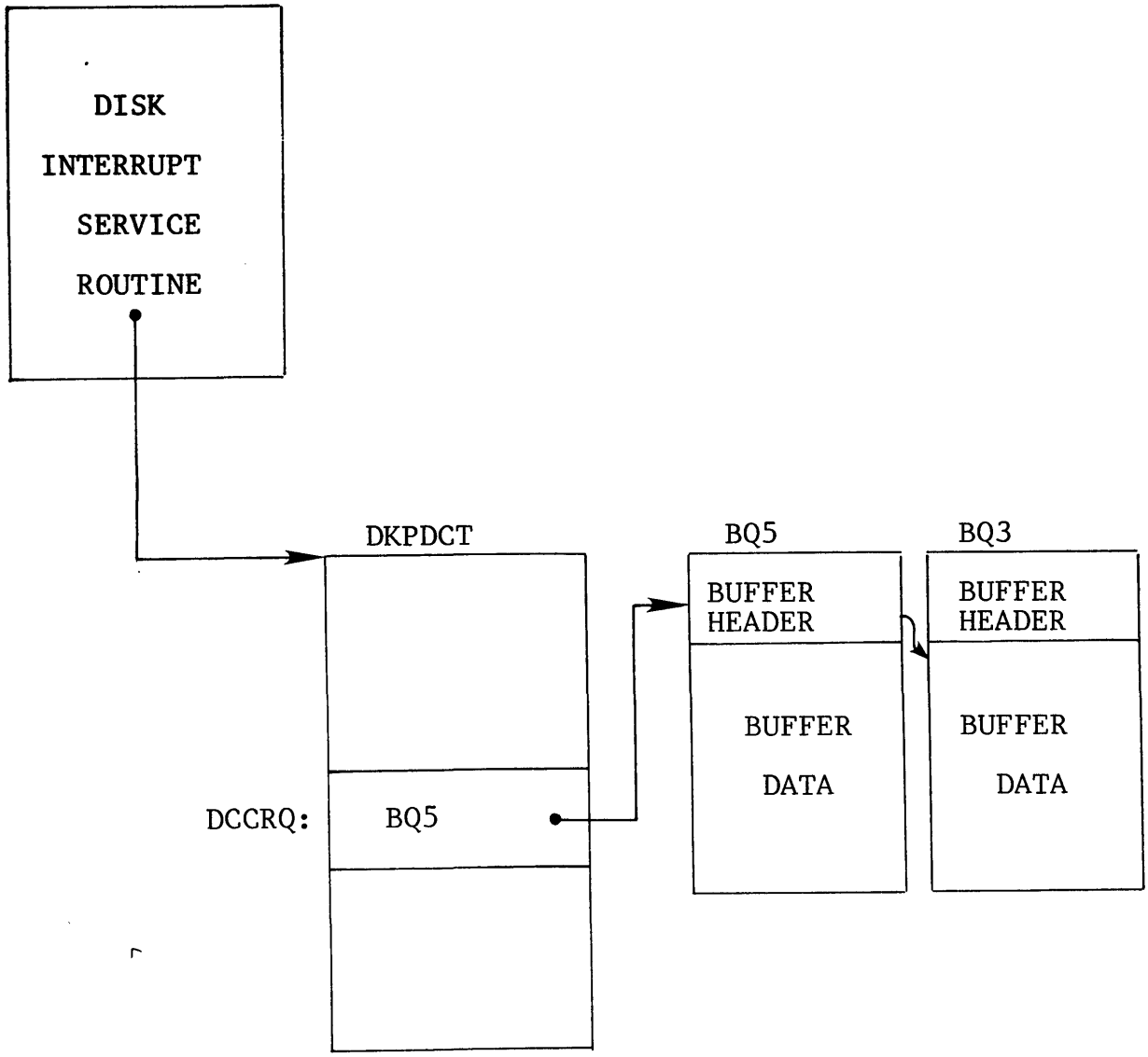
Each situation requires different considerations.

notes

Section 12

BUFFERS

DISK DCT AND BUFFERS



SYSTEM BUFFERS

System Buffers are segments of the resident operation system area which can be dynamically allocated for use either to hold system overlays or to act as a data buffer. RDOS requires a minimum of two buffers per system stack, with a lower limit of six buffers. RDOS allocates automatically two buffers per stack (≥ 6). Extra buffers can be chosen at system generation time. Extra buffers provide you with more efficient use of the system by allowing system overlays and data buffers to remain in core. This reduces the number of disk accesses required during the running of a program.

BUFFER MANAGEMENT

Various modules in RDOS manage the allocation of the system buffers A buffer can be allocated to hold a system overlay, or to act as a data buffer.

Each buffer has a 15_8 word header which indicates the status of the buffer. These buffer headers are linked together.

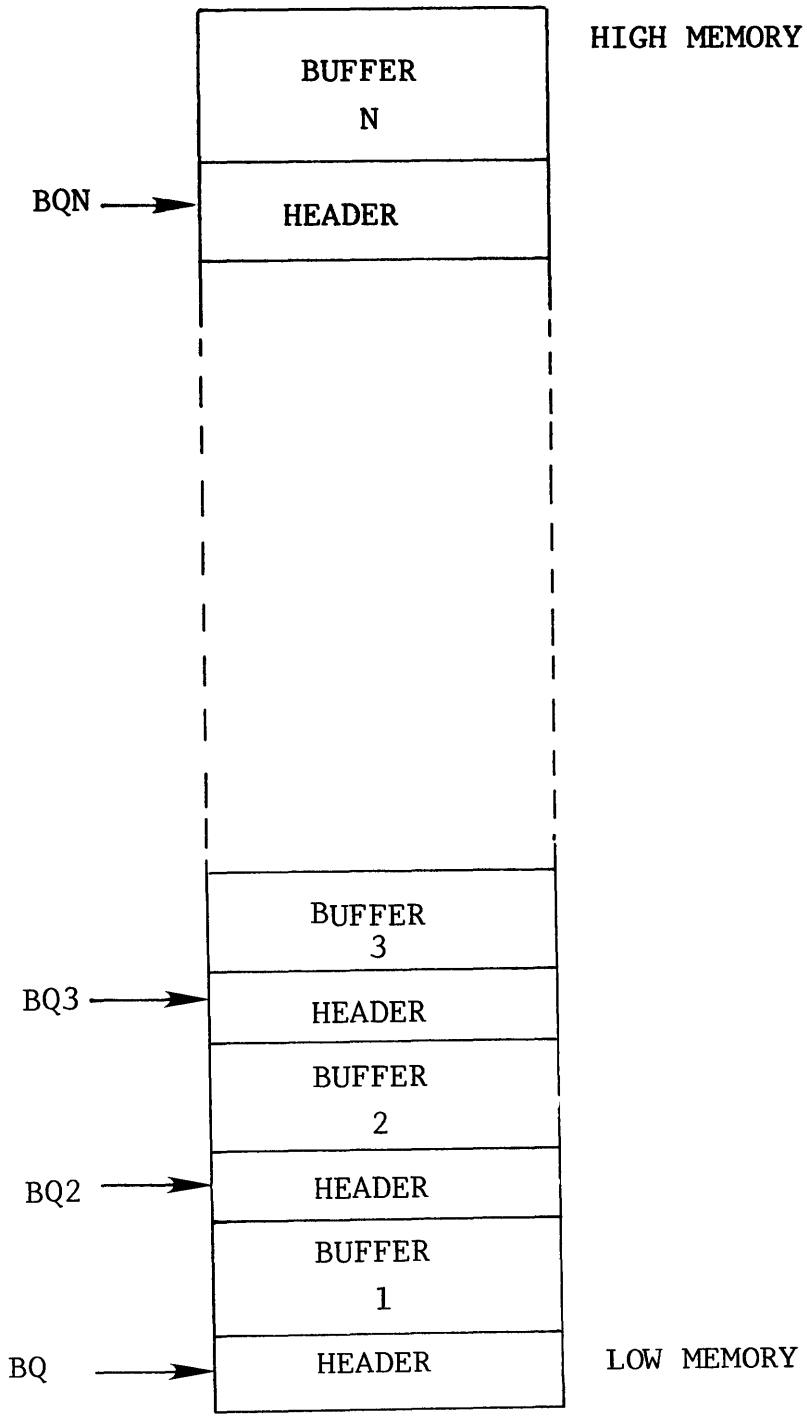
The system routines search these headers looking for an available buffer. If none is found, it searches for the oldest buffer (one that has not been used for the longest time) and uses the buffer as long as the buffer is not locked (cannot be flushed), i.e., BQUSC $> \emptyset$.

If the buffer modify bit has been set (i.e., a data buffer which has been modified) it will be written back to the disk (flushed). Otherwise it will be loaded over immediately.

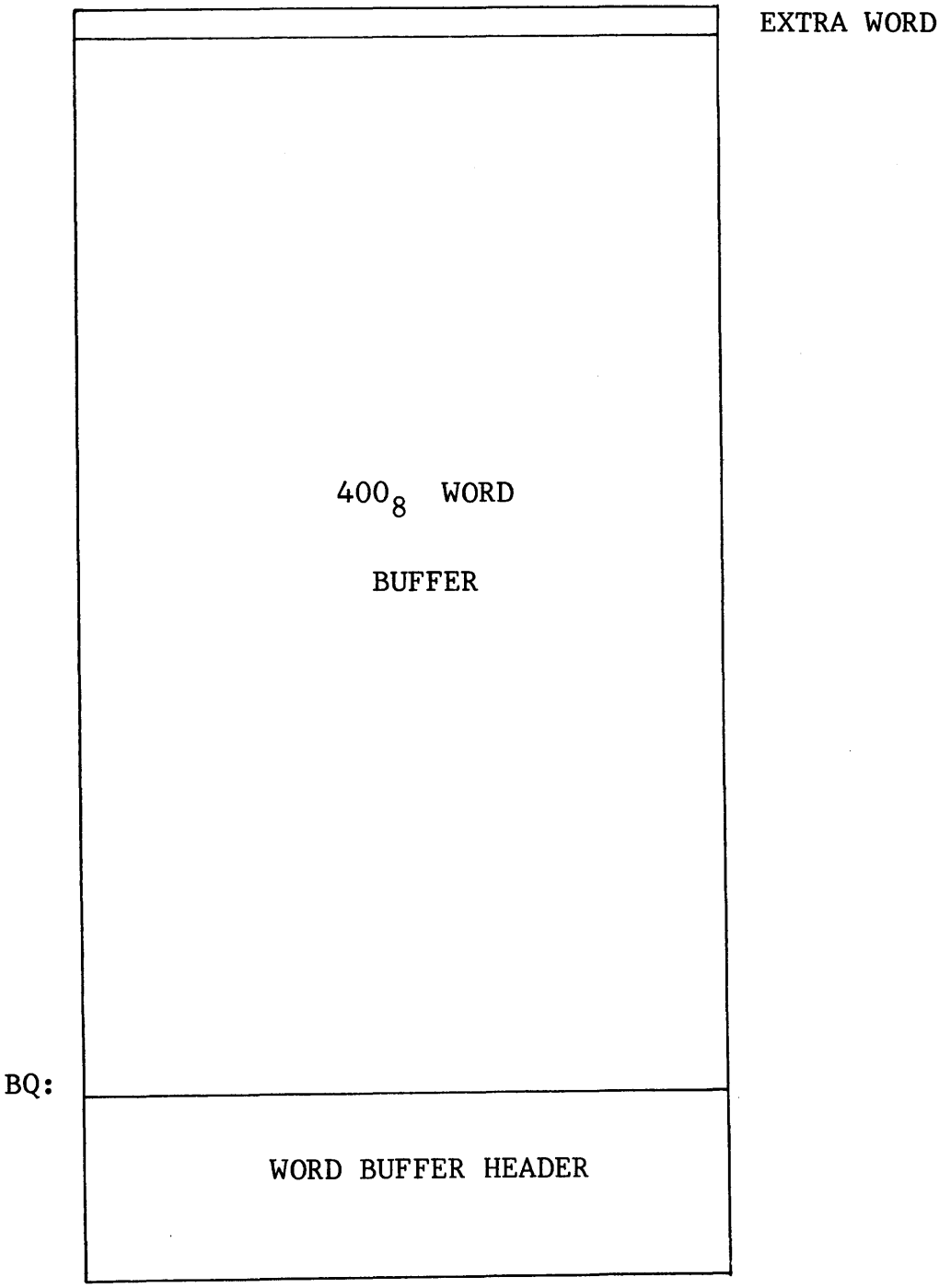
A one word database, BUFNT, contains a count of the number of process paths awaiting a buffer to become available.

BQ is the address of the first system buffer.
BCEC is the number of system buffers in the system.
A one word database, SYSTM, is incremented each time a buffer is assigned. This value is loaded into BQTLA of the buffer header when it is allocated.

SYSTEM BUFFER AREA



SYSTEM BUFFER



SYSTEM BUFFER HEADER

MOVING HEAD DISK

BQm:			
	Ø	START OF DATA	BQBF
	-1	LINK TO NEXT BUFFER	BQNXT
	-2	CURRENT BLOCK DEVICE ADDRESS (LOW ORDER)	BQCA
	-3	CURRENT BLOCK DEVICE ADDRESS (HIGH ORDER)	BQCA1
	-4	UNIT NUMBER	BQUN
	-5	DCT ADDRESS	BQDCT
	-6	STATUS WORD	BQST
	-7	ERROR COUNT	BQERC
	-10	HEAD, SECTOR, UNIT FOR DKP	BQDST
	-11	DEVICE REQUEST QUEUE LINK	BQQLK
→	-12	DCB ADDRESS	BQDCB
	-13	COUNT OF BUFFER USERS	BQUSC
	-14	TIME LAST ASSIGNED (Ø = USE IT FIRST)	BQTLA
	-15	CYLINDER NUMBER FOR DKP	BQCYL

BUFFER HEADER STRUCTURE

The following refers mainly to buffers containing disk blocks. Many are used differently, or not at all, when the buffer is being used for magnetic tape data.

WORD	400,	BQXTA	Extra word used when buffer involved in magnetic tape transfers.
WORD	377,	BQNXL	For a sequential disk file this is the link word to the adjacent blocks in the file. For a random or contiguous disk file, or for a magnetic tape file, it is a valid data word.
WORD	Ø,	BQBF	Start of the buffer data area.
WORD	-1,	BQNXT	Address of the next system buffer. Last buffer has 1BØ.
WORD	-2, -3,	BQCA BQCA1	Logical address of the block on the disk from/to which the buffer is being transferred. 2 words used.
WORD	-4,	BQUN	Device unit number from/to which the buffer is being transferred.
WORD	-5,	BQDCT	Address of the DCT which controls the device from/to which the buffer is being transferred.
WORD	-6,	BQST	Status word (Flags described later).
WORD	-7,	BQERC	Error count. Initialized to $+12_8$ every time buffer queued for I/O.
WORD	-10,	BQDST	Head, sector, unit, sector count for moving head disk. (Register C data). Valid if 1B11 of BQST set.
WORD	-11,	BQQLK	Device request queue link. Address of next buffer in queue (-1 if last). Only applicable when buffer involved in I/O transfer or queued for I/O.
WORD	-12,	BQDCB	Address of parent device's DCB in Queue Control Block (DSQ1+42). If 1BØ buffer contains system overlay and word contains overlay table address of that overlay.

WORD -13,	BQUSC	Count of the number of users of the buffer. Must be zero for the buffer to be available.
WORD -14,	BQTLA	Time buffer was last assigned. A counter (SYSTM) is incremented every time any buffer is assigned. This value stored in word every time assigned. (Ø implies buffer not in use.)
WORD -15,	BQCYL	Cylinder number for moving head disk. Valid if 1B11 of BQST set.

BUFFER STATUS FLAGS

BIT	INDICATION IF SET	
15	BUFFER MODIFIED	QTMOD
14	ERROR DETECTED	QTER
13		
12	I/O IN PROGRESS	QTIOP
11	MHD SET UP	QTDSU
10	INDIRECT MODE (ADDRESS IN BQNXT)	QTIND
9	ERROR MODE (MTA/CTA)	QTEMD
8	CONTIGUOUS FILE	QTCT
7	IPB LOCK IN PROGRESS	QTLKI
6	I/O IS TO EXTENDED MEMORY AREA	QTEXI
5	SPECIAL MODE DISK I/O FLAG	QTSIO
4	INHIBIT BAD BLOCK MAPPING AND REALLOCATION	QTNBAD
3		
2		
1		
0	IPB LOCK IS WAITING	QTLKW

BUFFER STATUS FLAGS

BIT 15 — Buffer modified. The data buffer has been modified and will eventually be ready for a transfer out to the disk. It cannot be re-assigned until this bit is reset by the completion of a write.

BIT 14 — Error detected. This is not set until a full retry has been made.

BIT 12 — I/O in progress. This buffer is in disk queue awaiting an I/O transfer.

NOTE: BIT 15, BIT 12 and BQUSC describes the state of any buffer completely.

BIT 11 — The buffer has the device set up. Applies to MHD, and implies that BQDST and BQCYL are valid.

BIT 10 — Indirect mode. Used to indicate that direct block I/O is being used (and hence this is a fake buffer header in a cell). Address in BQNXT.

BIT 9 — Error mode for magnetic tape. Used to indicate magnetic tape is being backed up for retries. (Ø = backspace needed, 1 = erase needed).

BIT 8 — Contiguous file. Direct block I/O uses this indicator.

BIT 7 — IPBlock in progress

BIT 6 — I/O is to extended memory area (mapped system only) (indirect mode only).

BIT 5 — Special Mode Disk I/O Flag

BIT 4 — Inhibit Bad Block Mapping & Reallocation

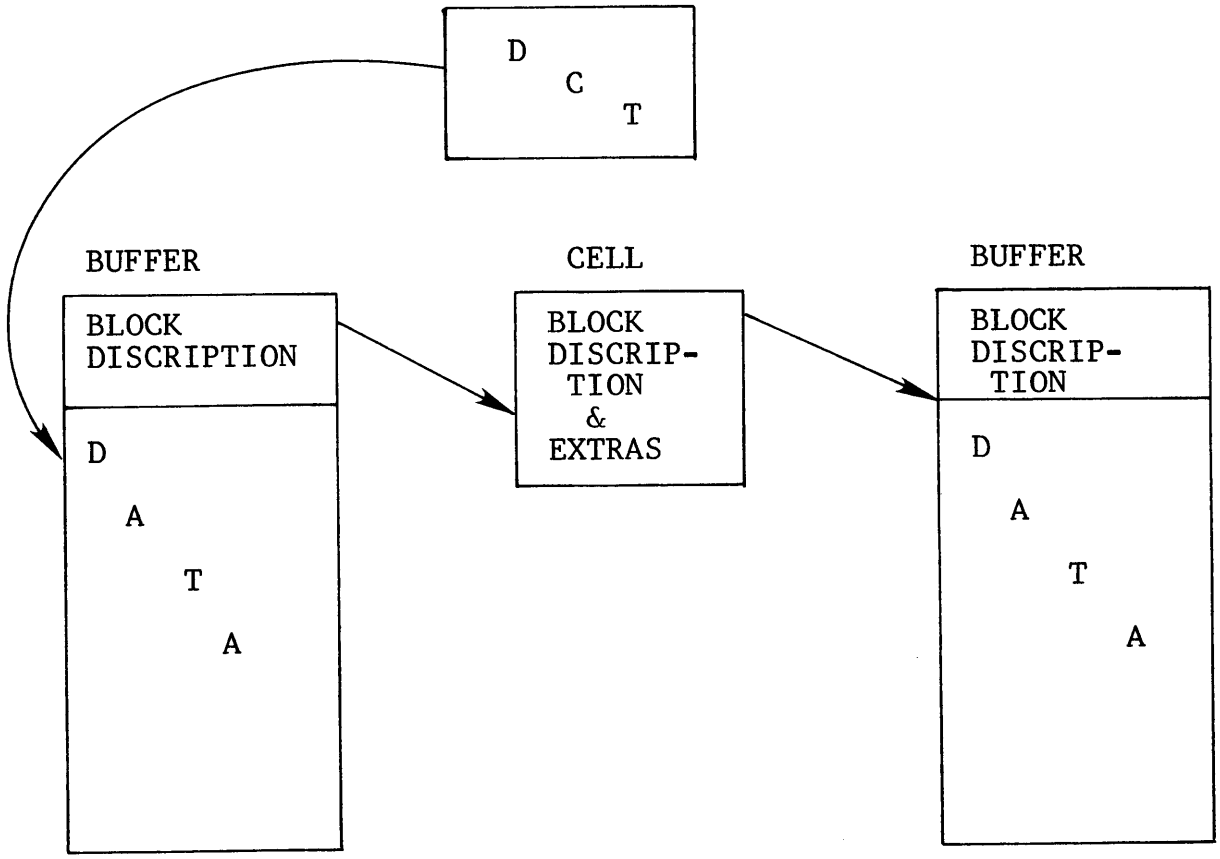
BIT Ø — IPB lock is waiting.

FAKE BUFFER HEADERS

Direct Block I/O methods do not use system buffers for their data storage. All data transfers are directly between user memory and disk. In these cases a system buffer header is not available to control the transfer. So that similar transfer methods are possible with data blocks being transferred by direct block I/O as there are with data blocks being transferred via system buffers, fake buffer headers are required.

These headers directly resemble the system buffer headers. They enable the blocks to be queued for disk I/O in the same manner as the system buffers.

The fake buffer header is built in a Task Request Cell, using the cell purely as a temporary storage area. The negative offsets to -2 have the same significance as for system buffer headers.



The additional offsets for fake Buffer Headers

Word	-1,	BQADR	User's logical core address to/from which I/O transfer is to take place. (In system buffer header this location is BQNXT).
Word	0,	BQUST	User status word. (For mapped systems)
Word	1,	BQNBK	Number of 400_8 word blocks to be moved in the transfer. Decremented at each output.
Word	2,	BQARD (Random) BQSCT (contig)	If contiguous file, # of sectors. If a random file then an addressing array is constructed in a system buffer. This array is constructed from the random file index block and refers only to the blocks involved in the transfer. This word points to the current index block entry in the array. Not used for one block random transfers.

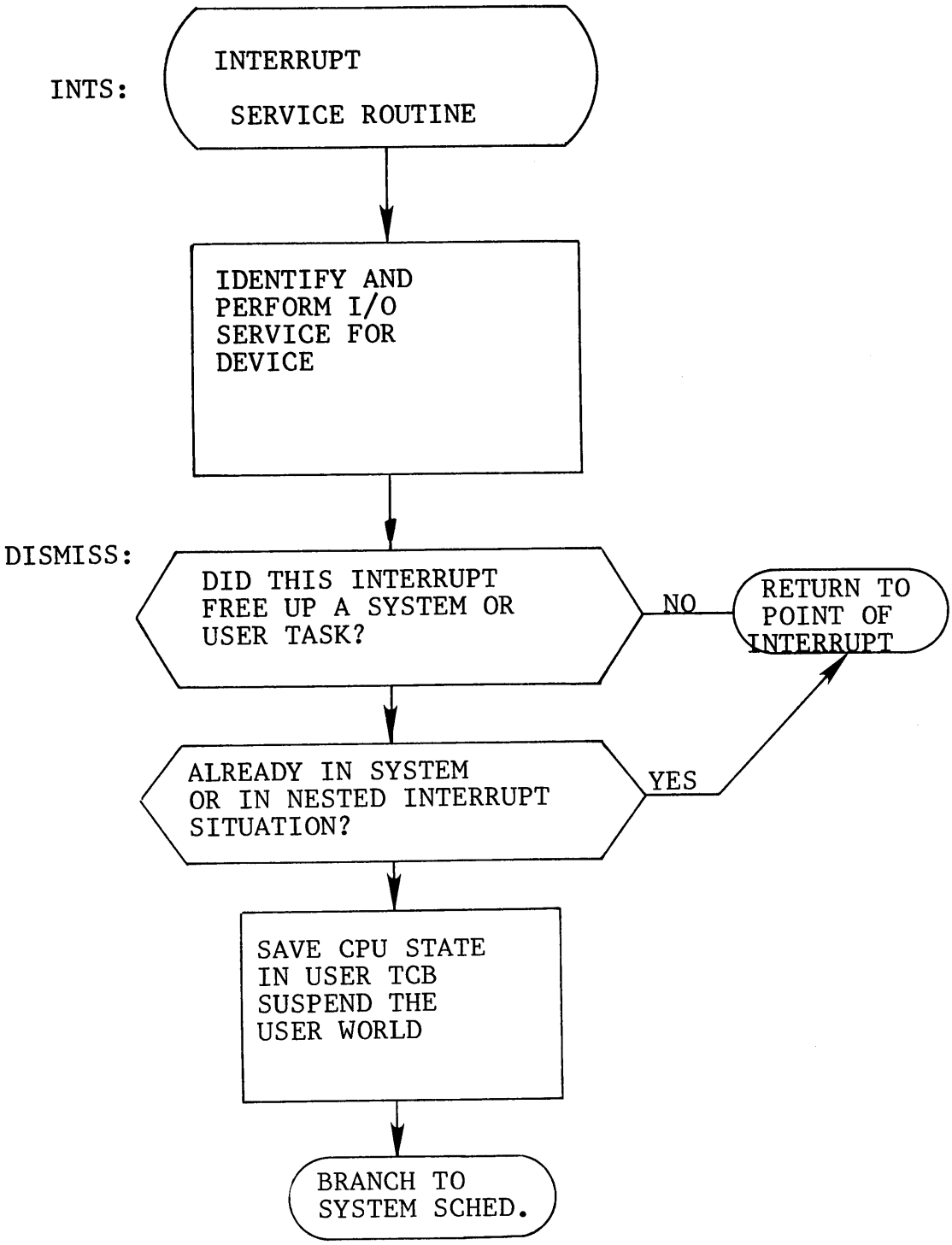
NOTE:

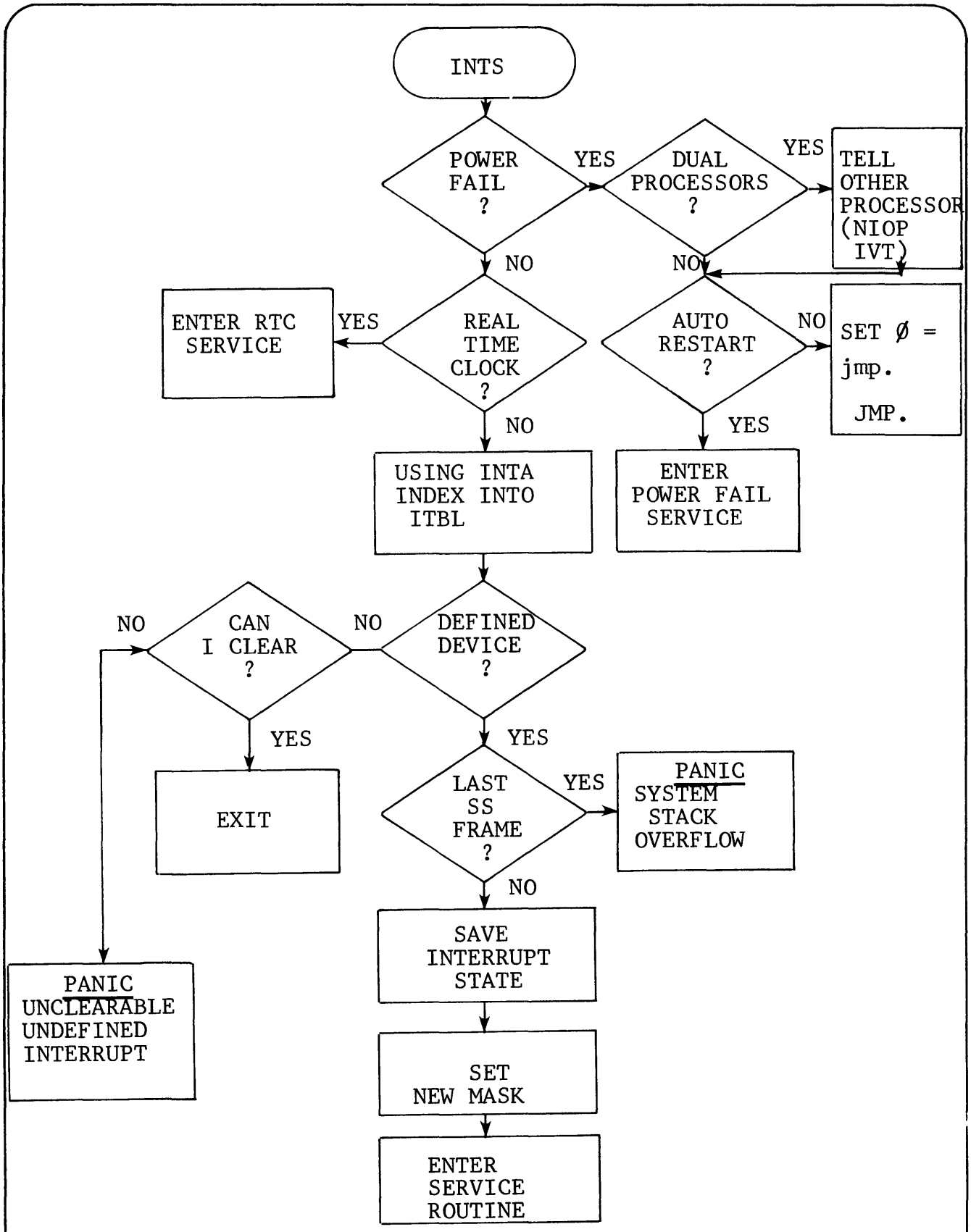
Word	-11,	BQDCB (Rev 3.XX)	When header is a fake buffer header in a mapped system this word has the address of the user map section of the program table for the ground that is getting/ sending the disk block. (Contains the address of MPT1 or MPT2).
Word	-12,	BQDCB (Rev 4.XX)	

notes

Section 13

INTERRUPTS





N O V A
INTERRUPT SERVICE ROUTINE (INTS)

INTERRUPT VECTOR TABLE (located in TABLE.SR)

All devices in the system have the address of their DCT located in an interrupt vector table. This table has one entry for each possible device code or logical device link. The entries are in ascending order from Ø. The table indicates, in addition to the DCT address, the following possibilities:

(1) RDOS sysgened device
(2) Device introduced by an interrupt definition system call (.IDEF)

If the device does not exist in any form then the entry for that device code is set to -1 for Nova to the address of the Interrupt from undefined device (IUD) routine for Eclipse.

The mapping units and the real time clock do not have associated DCT's.

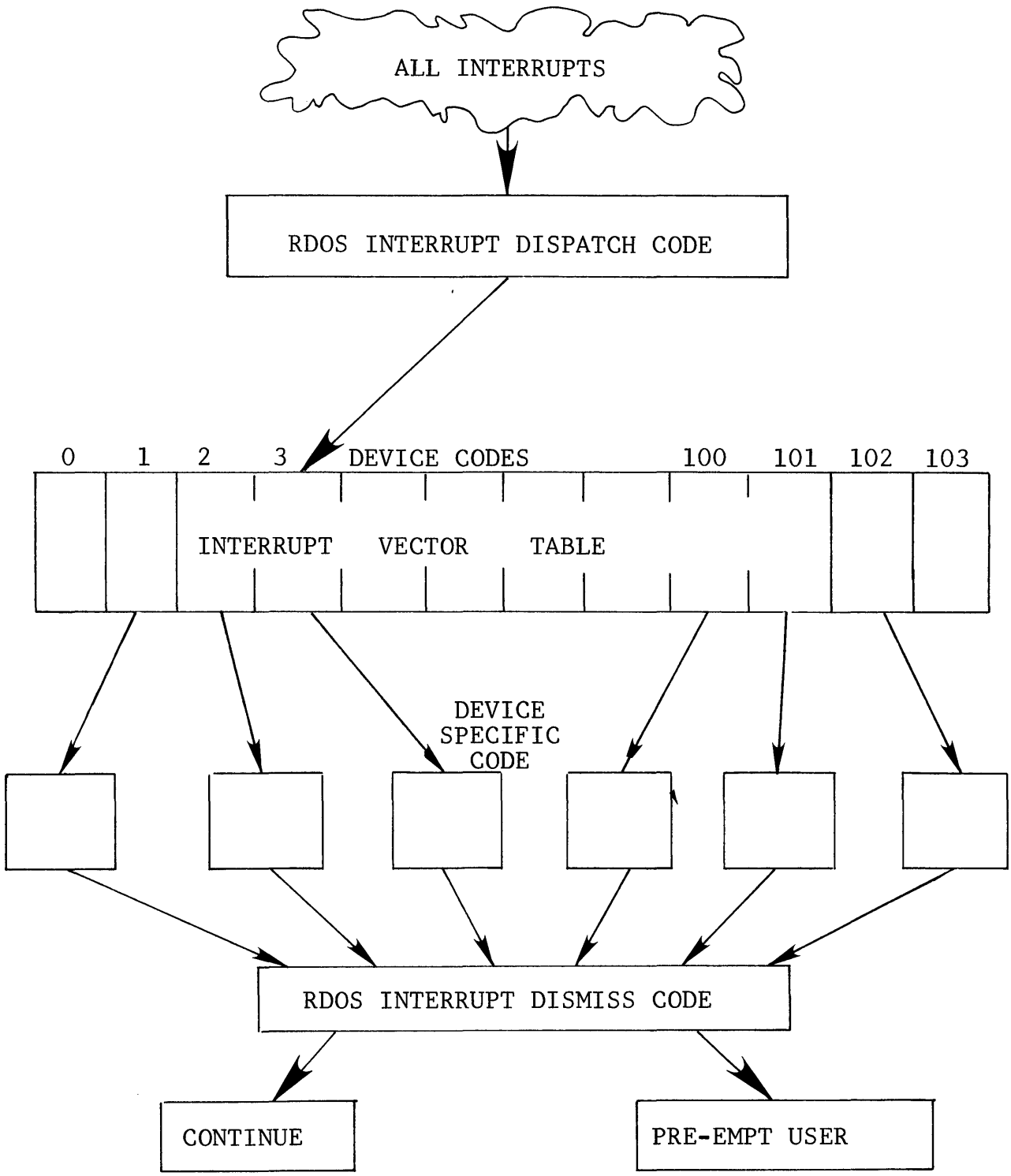
A number of pointers are associated with the Interrupt Vector Table.

IVFWA	is the address of the first word of the ITBL
IVLWA	is the address of the last word of the ITBL
ITBL	is the address of the first actual device code (i.e., 1 in NOVA RDOS)
BTBL	is the address of the first word (i.e., Ø in ECLIPSE RDOS)

There are also a number of one word databases which, although not associated with the ITBL directly, are associated with interrupt processing.

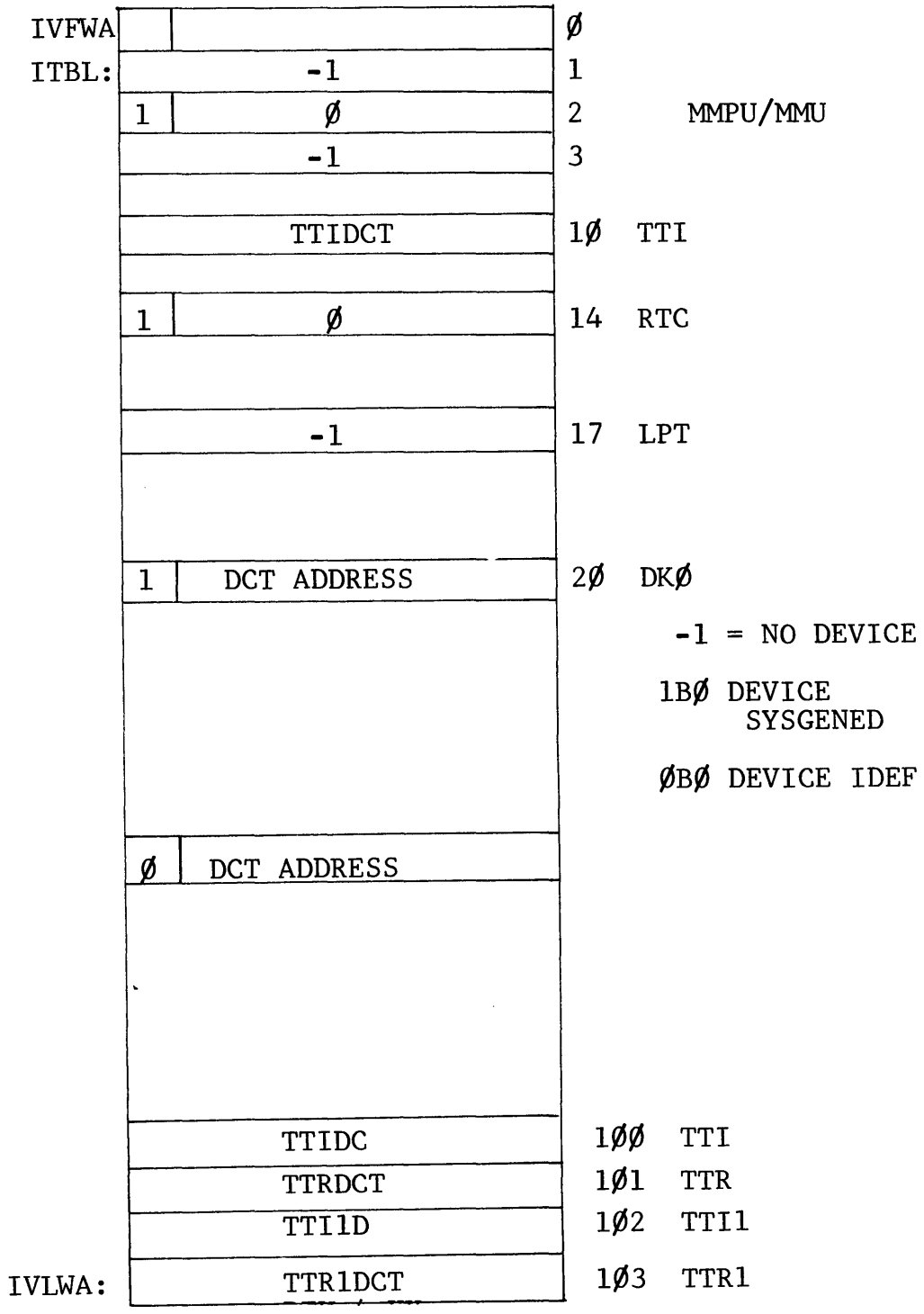
	INTLV	contains the interrupt depth. (Initially - 1 on Eclipse, Ø on Nova) (number of system interrupts being processed)
Nova only	LINT	contains the device code of the last device which interrupted. (excluding the real time clock) (located in INTD.SR)

INTERRUPT CODE FLOW



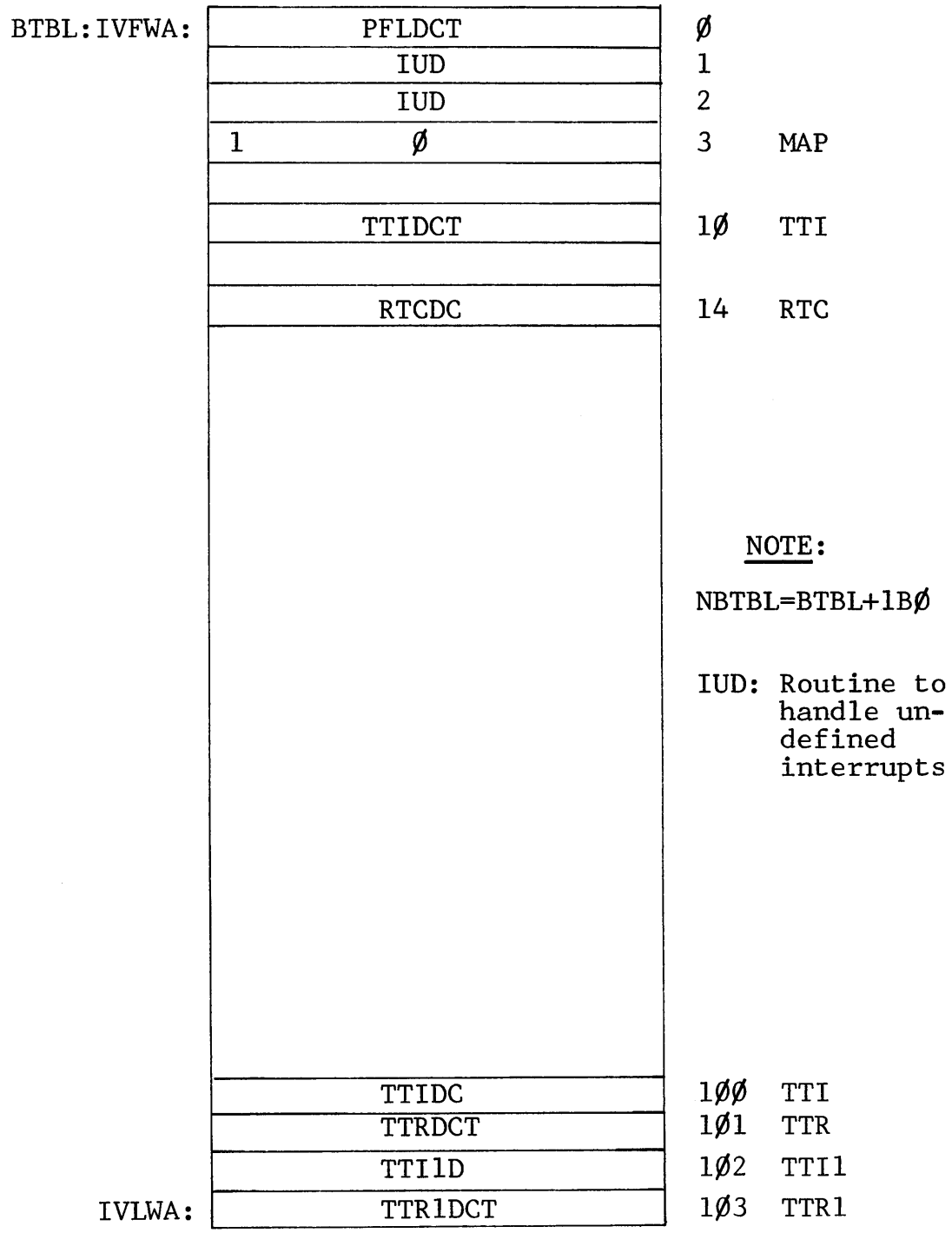
INTERRUPT VECTOR TABLE

NOVA



INTERRUPT VECTOR TABLE

ECLIPSE



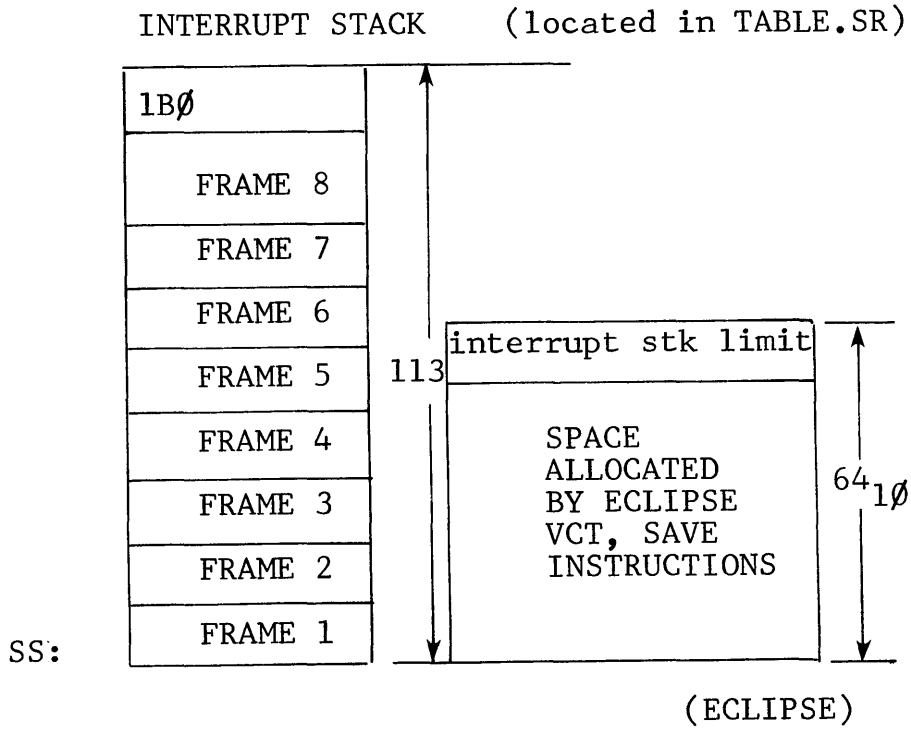
NOTE:

NBTBL=BTBL+1BØ

IUD: Routine to handle undefined interrupts

INTERRUPT STACK

This stack is used during the processing of interrupts. As nested interrupts occur, new frames on the stack are allocated to save machine state. If this stack overflows (8 unserviced interrupts) a panic will occur. This usually only occurs from a continually interrupting device that cannot be masked or cleared by the systems.



In the case of the interrupt stack, machine state is saved in the temporaries of the normal stack frame. This is possible as first level interrupt processing modules do not use stack frame temporaries. All other processing is carried out using the system stacks, if the processing path needs a stack.

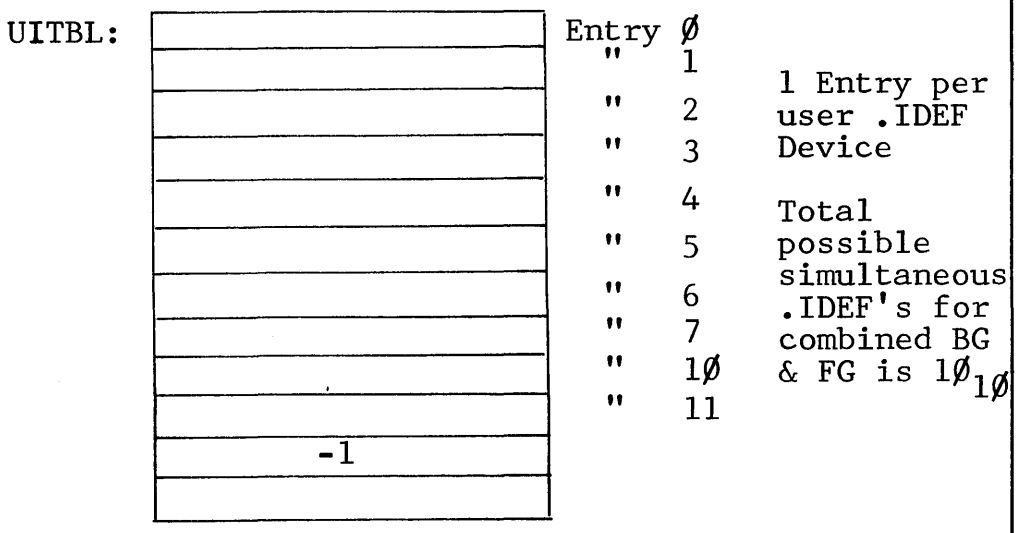
INTERRUPT STACK FRAME

14		VRTN
13	STATE OF USER MAP Ø in user, 1 in absolute	INTUS
12	RLOC	IRLOC
11	CURRENT MASK	ICMSK
10	AC3	IAC3
7	AC2	IAC2
6	AC1	IAC1
5	ACØ	IACØ
4	PC AND CARRY (in 1B15)	IPCC
3		AC2
2		AC1
1		ACØ
0		RTLOC
-1		SP

NOVA ONLY (ECLIPSE RDOS DOES NOT USE THIS TECHNIQUE)

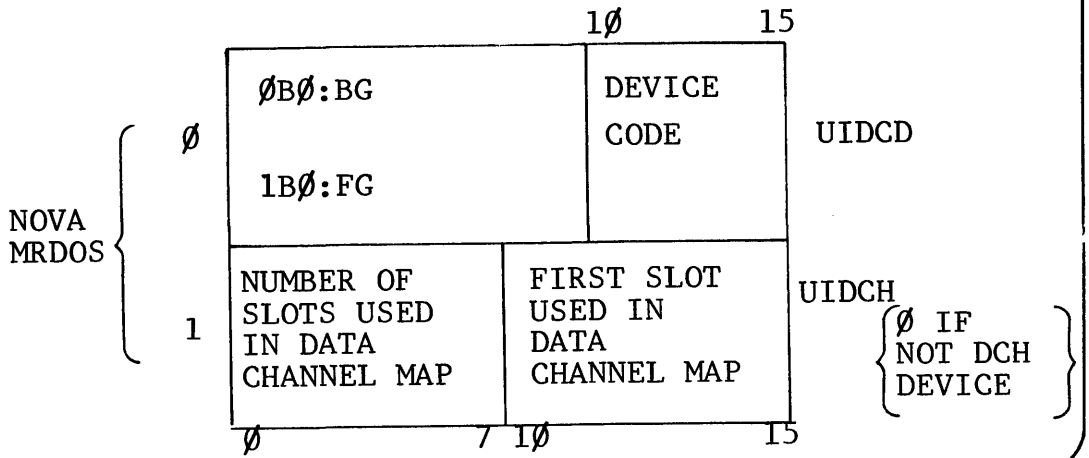
USER DEVICE TABLE

MAPPED SYSTEMS ONLY

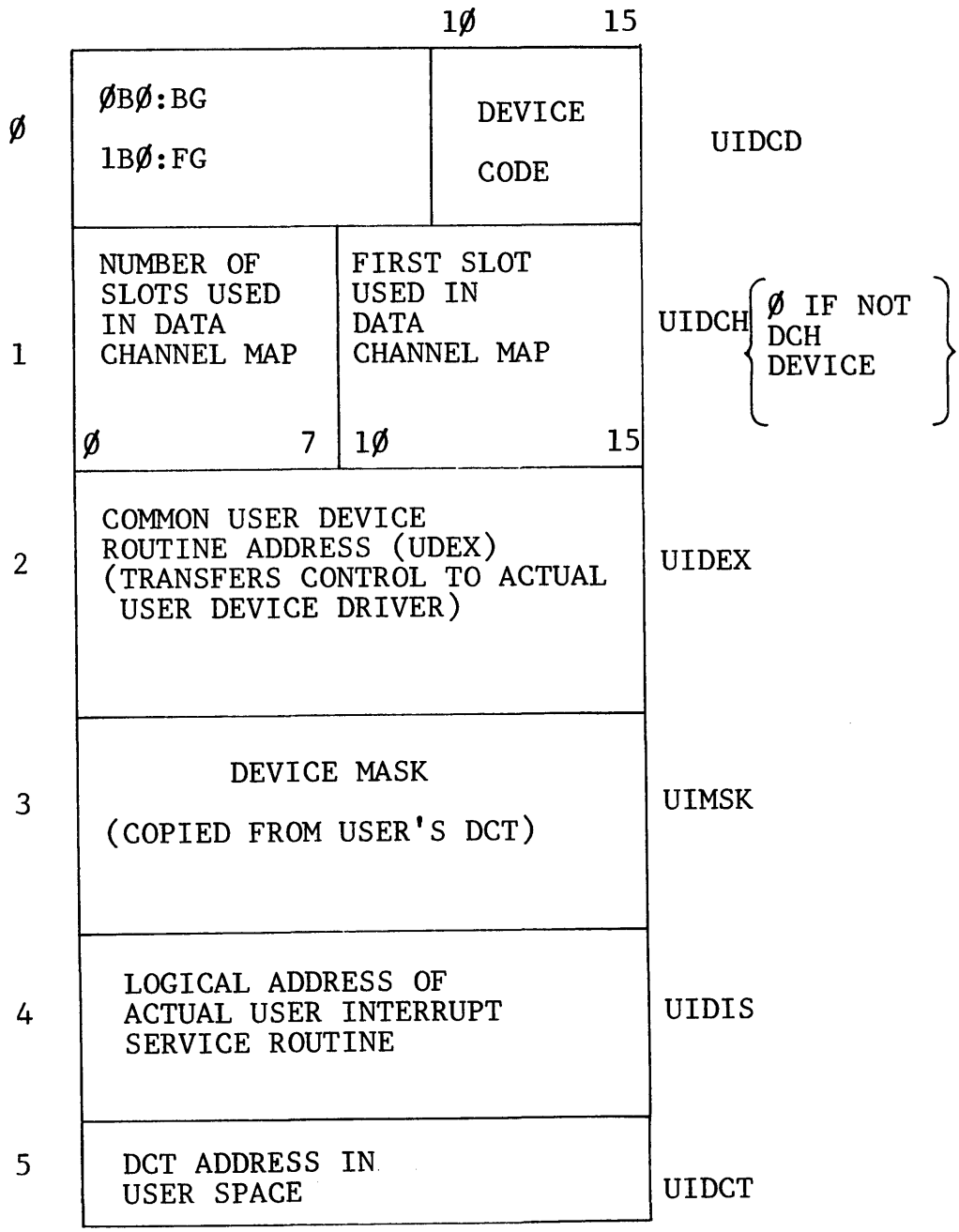


USER DEVICE TABLE

ENTRY



USER DEVICE TABLE ENTRY



ECLIPSE ARDOS

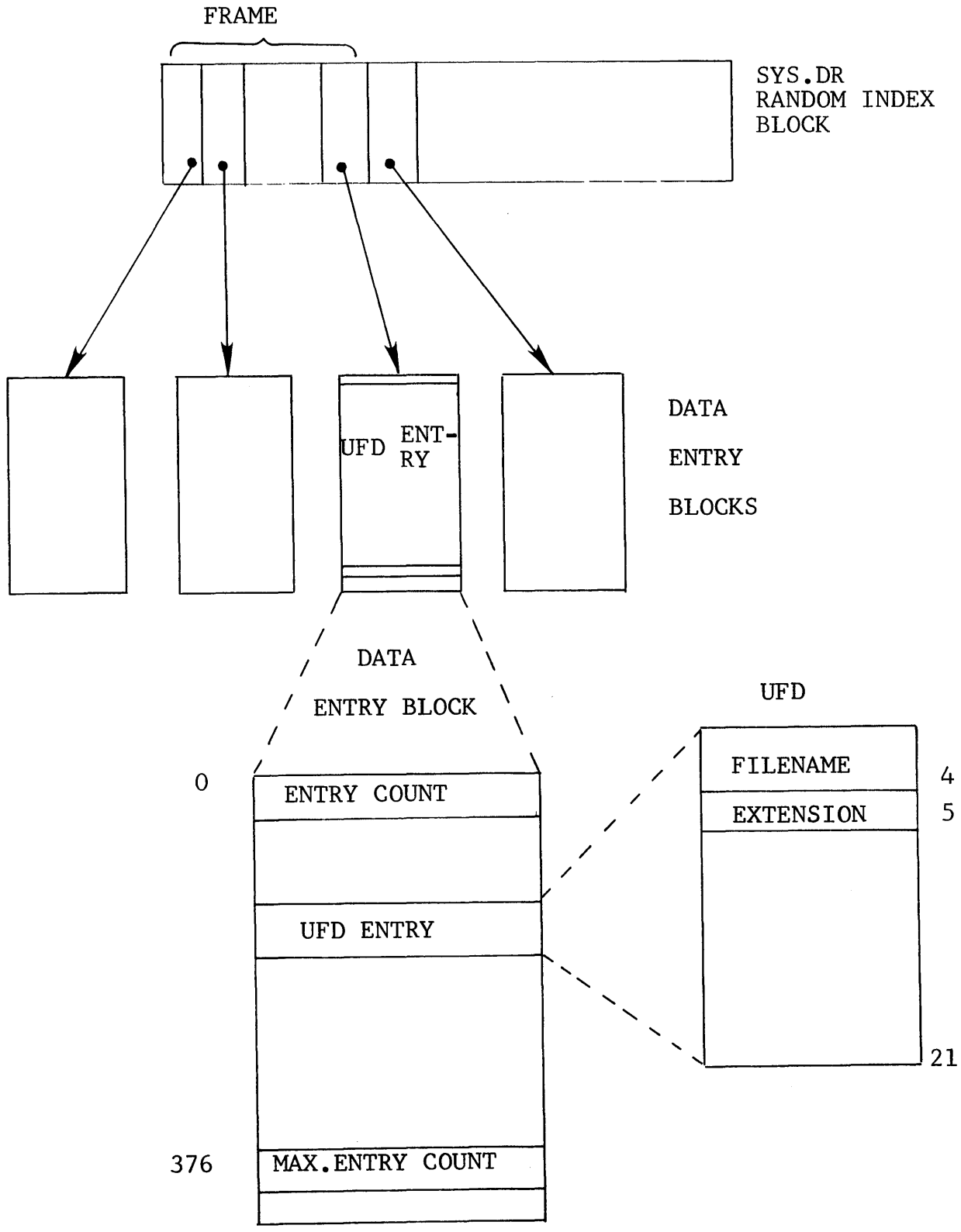
notes

notes

Section 14

DISK STRUCTURES

SYS.DR STRUCTURE



DISK ORGANIZATION

Information required about files stored on a disk, whether in the primary partition, a sub-partition, or a sub-directory, is contained in a system file directory called SYS.DR. The information within every SYS.DR includes file names, the length in bytes of the files, and the file attributes and characteristics.

A current record is maintained of all the disk blocks in use in a disk map directory called MAP.DR. There is a distinct MAP.DR for the primary partition and for each sub-partition.

The first 20$_8$ blocks of any disk device have fixed assignments. The remainder is free for either system use or user file storage. The initial blocks are assigned when the disk is fully initialized. This initial assignment includes the system overlay file blocks (BOOTSYS.OL). System directories employ a hashing algorithm to speed up access of directory entries. An initial system directory area is allocated (at full initialization time) for entries in the primary partition of a moving head disk. This area (called a frame) is a contiguous set of disk blocks. (Contiguous to minimize moving head travel).

Sub-partitions and sub-directories allocate system directory storage as required.

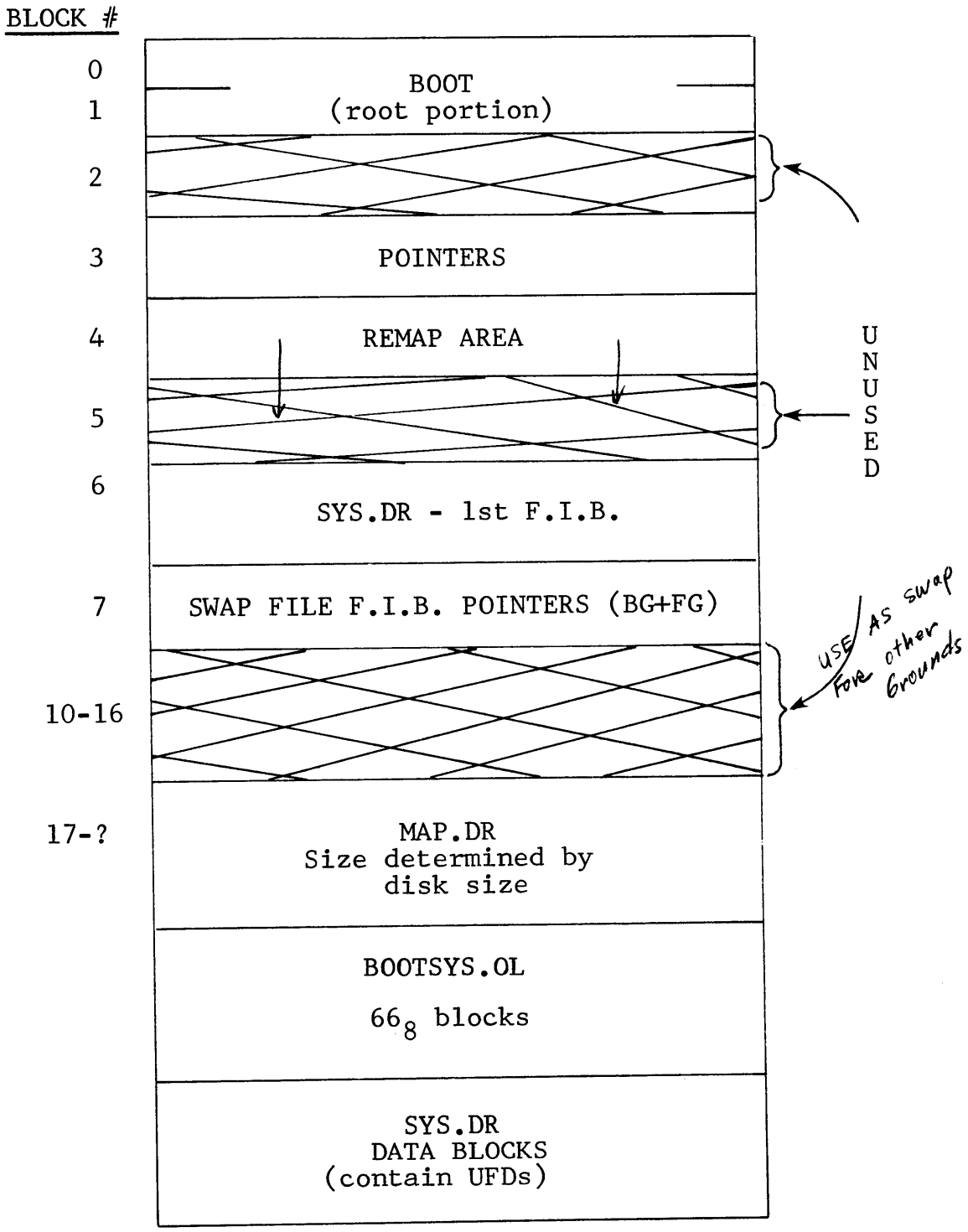
The primary partition frame size is dependant upon the type of disk unit on which it is located. For DGC moving head disk types:

DISK TYPE	REV 3.XX FRAME SIZE	REV 4.XX FRAME SIZE
4047	141	103
4048	301	155
4057	1405	255
4231	N/A	255

For REV 4.XX, a copy of the frame size is recorded as word 21 of block 7 on every disk.

For REV 5.XX and beyond frame size is a user option.

INITIAL BLOCK ASSIGNMENTS



INITIAL DISK BLOCK ASSIGNMENTS

REV 4.XX to 6.XX

The disk bootstrap program consists of two separate sets of disk blocks. The portion of the program loaded by the Program Load hardware occupies the first two blocks of a disk. This section of the bootstrap program is called the root section. The remainder of this program is contained in a disk file called BOOT.SV. BOOT.SV must reside in the primary partition of the same disk from which the root section was loaded or the message "BOOT.SV NOT FOUND" will be typed by the root.

Although the root section of BOOT only occupies 2 blocks, the first 6 blocks of any disk are not available for reassignment. As with REV 3.XX, MAP.DR's first bit corresponds to block 6, the first file index block of SYS.DR.

Block 3 contains information describing the type of disk system that the disk was generated on. This is used during initialization.

Block 4 is reserved but not used under REV 4.XX. REV 5.XX will use this block to keep track of bad blocks discovered on the disk. It is incorporated here for ease of "REVing up" the disk.

Block 7 is empty, except for the first 23 words. It contains the addresses of the swap file index blocks and a copy of this disk's frame size. The pointers are each two words long to allow for double word disk block addresses of the CDC (3330 type) disks.

Blocks 2, 5 and 10-16 are reserved but not used. They support inter REV compatability between REV 3.XX and REV 4.XX.

DISK BLOCK FORMATS

BLOCK 3

0	REV. CODE Ø= 4.Ø2, 2= 5.00
1	CHECKSUM
2	TRACKS PER CYLINDER
3	SECTORS PER TRACK
4	NUMBER OF BLOCKS
5	
6	FRAME SIZE
7	DISK TYPE CODE

BLOCK 4

0	# OF VALID WORDS IN THIS BLOCK
1	START OF REMAP AREA
2	
3	SIZE OF REMAP AREA
4	BAD BLOCK ADDRESS
5	
6	BAD BLOCK ADDRESS
7	
	. . .

DISK BLOCK CONTENTS

BLOCK 3

Word 1 - Calculated so that sum of words Ø → 7 = Ø

Word 2 - Number of heads per cylinder

Words 4 & 5 - Number of blocks - 6
= Number of valid bits in MAP.DR

Word 7 - 1BØ = 2 WORD ADDRESSES
1B1 = 4234 CONTROLLER
1B14 = 6030 DISKETTE
1B15 = 4231 CONTROLLER

BLOCK 4

DOUBLE WORD ADDRESS ADDED
FOR EACH BAD BLOCK ENTERED THROUGH
DK INIT

Copied into memory at each INIT sysgen "Bad Block Pool" reserves this core which must include space for all INITed disks.

DISK BLOCK #7 - FORMAT

INITIAL BLOCK ASSIGNMENT

WORD #	
0	
1	1st BG SWAP FILE INDEX BLOCK ADDRESS
2	
3	2nd BG SWAP F.I.B. ADDRESS
4	
5	3rd BG SWAP F.I.B. ADDRESS
6	
7	4th BG SWAP F.I.B. ADDRESS
10	
11	1st FG SWAP F.I.B. ADDRESS
12	
13	2nd FG SWAP F.I.B. ADDRESS
14	
15	3rd FG SWAP F.I.B. ADDRESS
16	
17	4th FG SWAP F.I.B. ADDRESS
20	
21	FRAME SIZE
22	66_8 (# of overlays)

This block is relative block 1 in a secondary partition

PUSH FILE ORGANIZATION

Any program executing under RDOS can suspend its own execution and invoke another program (SWAP) or another segment of itself (CHAIN). In the case of a swapped program the current user area (location 16 up to NMAX or SST) and the program's UFT's have to be stored on the disk. Up to five levels of program swap can occur so it is possible that four program levels have been pushed to the disk.

The swapped programs are stored as randomly organized files. Their random index blocks have fixed positions in the primary partition or subpartition. In REV 3.XX 4.XX & 5.XX have fixed pointers to the swap file index blocks, but the blocks themselves are assigned as they are needed.

Each time a partition is INITed, RDOS will RELEASE the blocks reserved for the swap files by the last RDOS.

notes

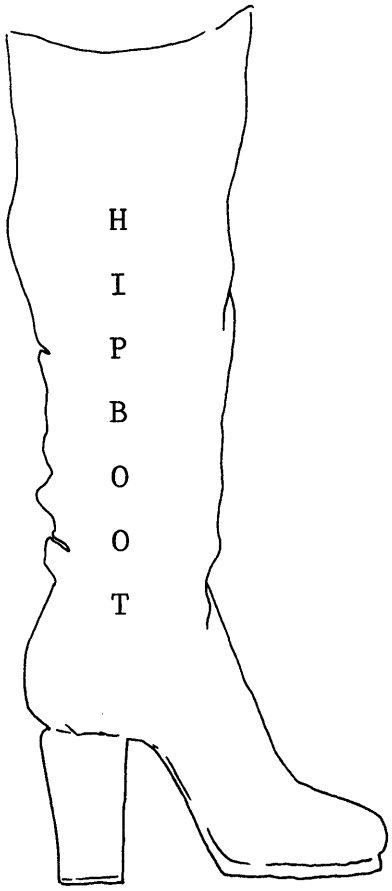
notes

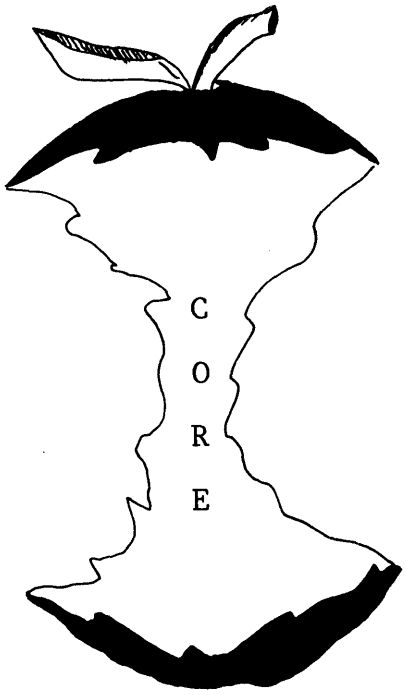
Section 15

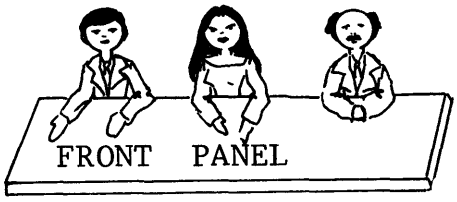
BOOTUP

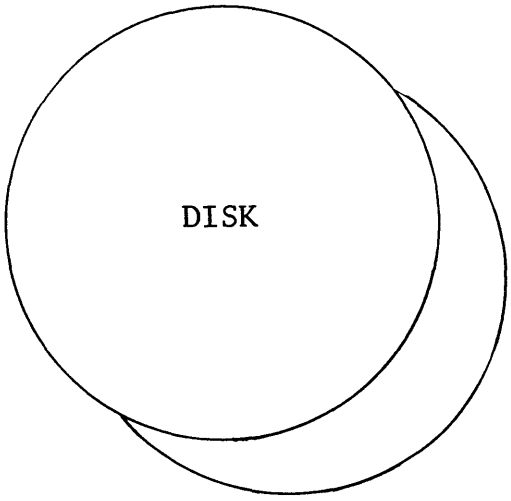
AND

PANIC CODES









MAJOR COMPONENTS OF SYSTEM INITIALIZATION

There are three start addresses for the initialization code. The address chosen depends on what type of initialization is required:

SIFUL	(JMP @1)	FULL
SSINI	(JMP @2)	PARTIAL
SSOVI	(JMP @3)	PARTIAL WITH OVERLAYS

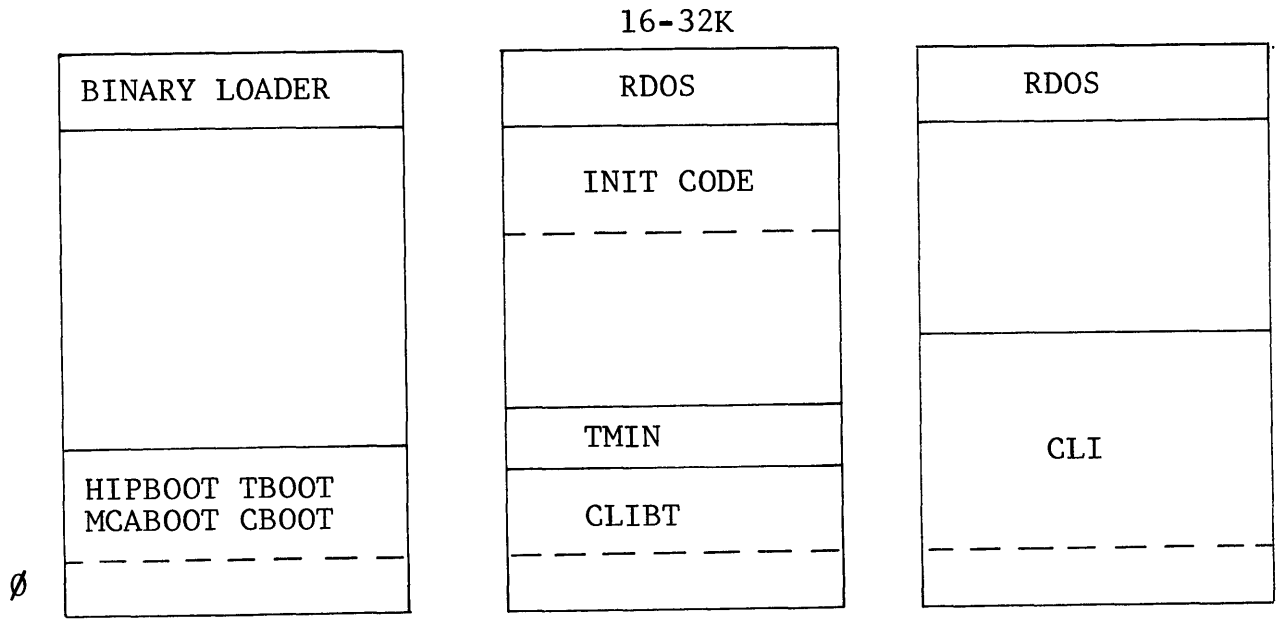
The types of initialization possible depend also on the method employed to bring the operating system into memory. The following table shows the permitted forms of initialization:

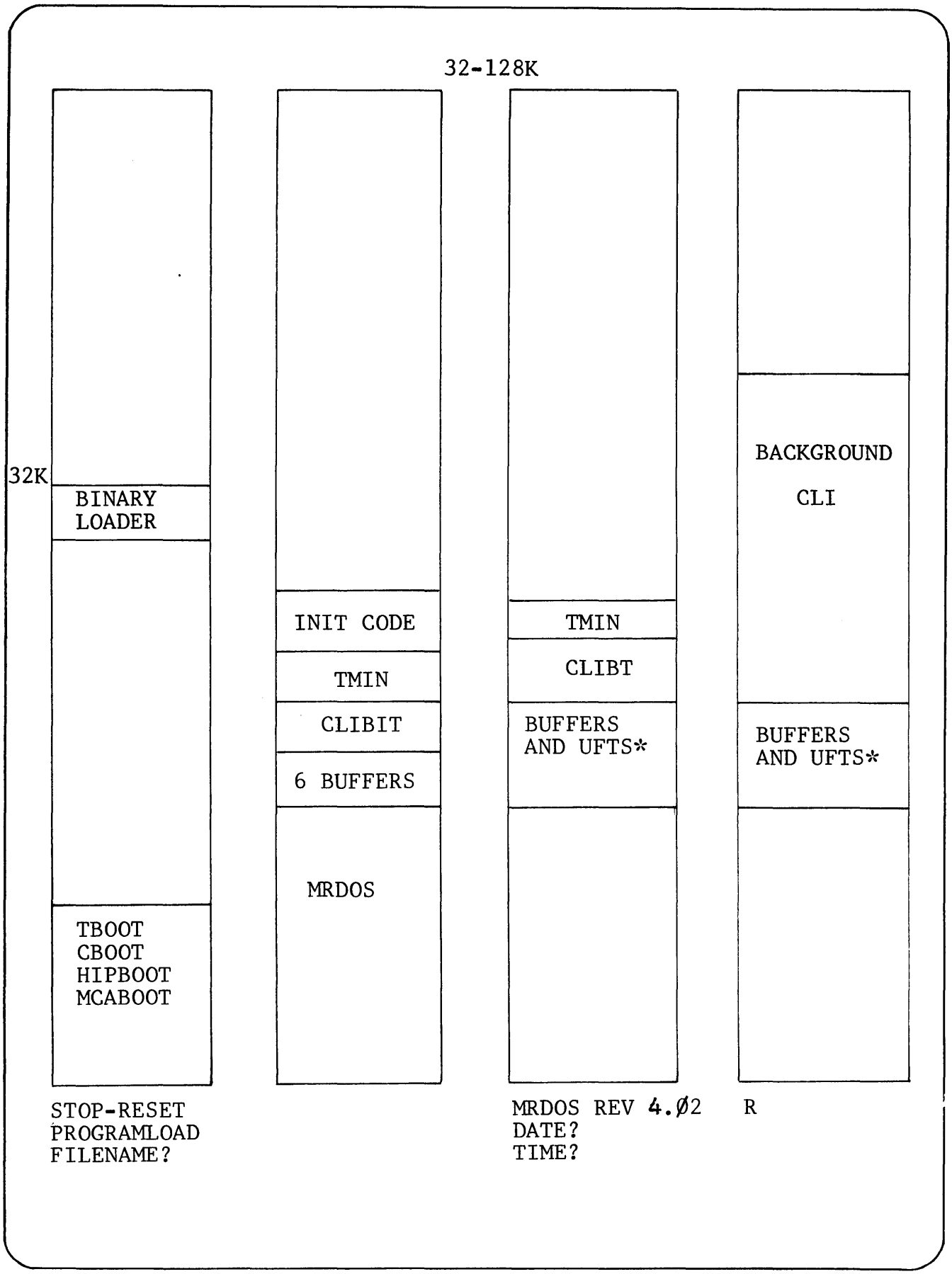
METHOD / TYPES	BINARY LOADER	HIPBOOT	TBOOT	CBOOT	MCABOOT
FULL	YES	NO	YES	YES	YES
PARTIAL	YES	YES	NO	NO	NO
PARTIAL WITH OVERLAYS	YES	NO	YES	YES	YES

RDOS SYSTEM INITIALIZATION SUMMARY

RDOS MAY BE BOOTSTRAPPED VIA ANY OF THE FOLLOWING:

DEVICE	BOOTSTRAP	CONSOLE SWITCH SETTING
~~TTR~~	BINARY LOADER	ØØØØ1Ø
~~PTR~~	(VIA PROGRAM LOAD TAPE)	ØØØØ12
~~CT~~	CBOOT	1ØØØ34
MT	TBOOT	1ØØØ22
DKØ	HIPBOOT	1ØØØ2Ø
DPØ	HIPBOOT	1ØØØ33
MCA	MCABOOT	1ØØØØ7





SUMMARY OF HIPBOOT ERROR MESSAGES

IN ORDER OF (POSSIBLE) APPEARANCE

ALL OCCUR BETWEEN

FILENAME? AND

(M)RDOS REV 4.Ø2

	MESSAGE	PROBLEM
1.	"FILE NOT FOUND:_____"	MISSPELLED/NON-EXISTENT FILENAME OR PARTITION NAME.
2.	"PARTITION IN USE, TYPE C TO CONTINUE"	PREVIOUS USER OR DISK FAILED TO CLEAR FILE USE COUNTS (IF IPB, OTHER CPU IS USING THIS PARTITION, DO NOT TYPE C)
3.	"FILE NOT EXECUTABLE FILE:___"	SYSTEM SAVE FILE IS EITHER NOT RANDOMLY ORGANIZED OR LACKS "S" ATTRIBUTE.
4.	"OVERLAY FILE MUST BE CONTIGUOUS"	
5.	"LINK DEPTH EXCEEDED AT FILE _____"	

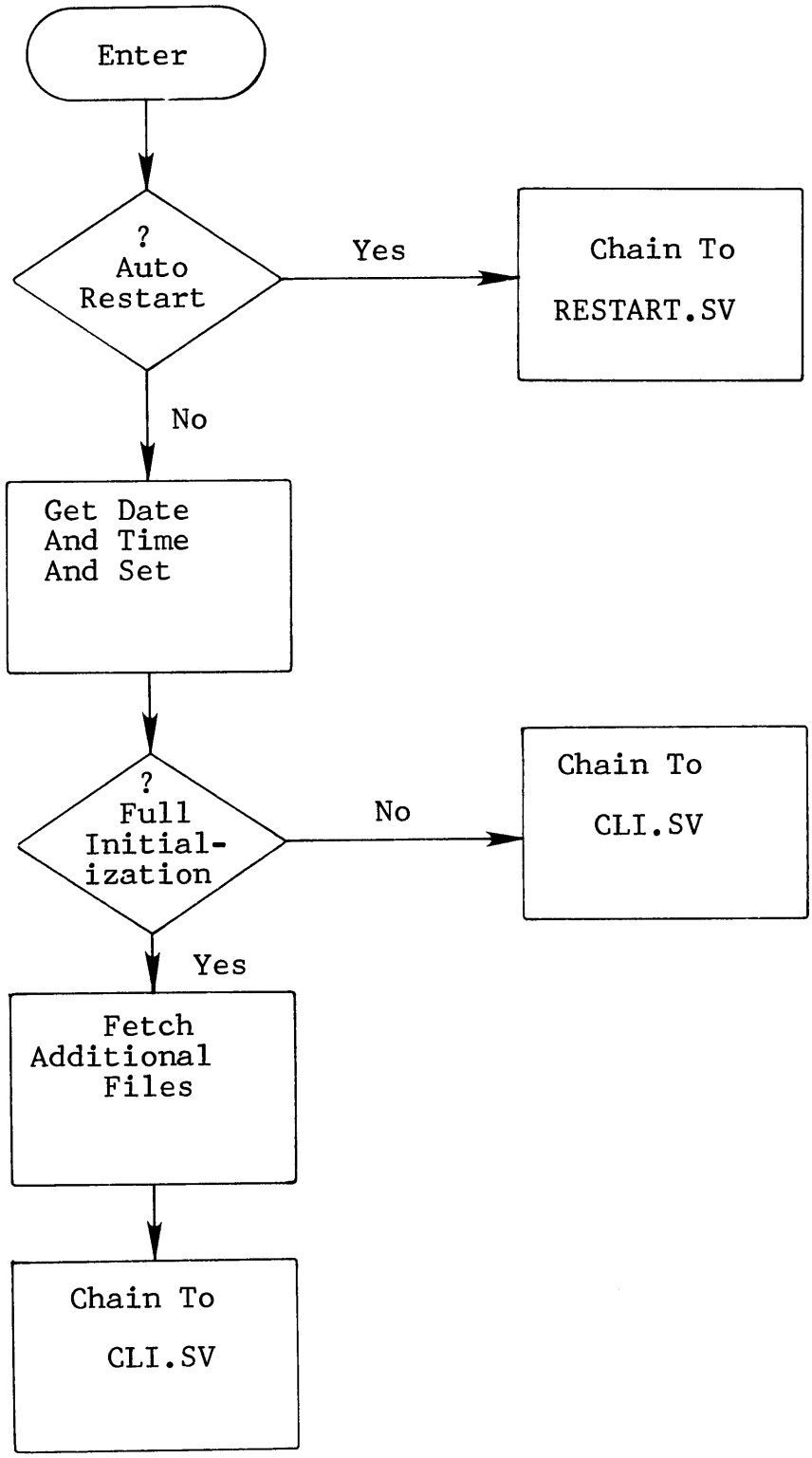
ERRORS 3), 4), 5) REQUIRE THAT YOU BOOT ANOTHER SYSTEM (SUCH AS BOOTSYS) TO FIX THE PROBLEM

	MESSAGE	PROBLEM
6.	"DIRECTORY DATA ERROR"	INTEGRITY OF SYS.DR IN QUESTION, PROBABLE FULL INIT REQUIRED, I.E., FATAL ERROR.
7.	"DISK ERROR-STATUS ___"	RUN DISK RELIABILITY
8.	"HARDWARE FAILURE"	HIPBOOT DID NOT RECOGNIZE PARTITION OR DEVICE NAME, USUALLY ONLY OCCURS ON MACHINE FAILURE
9.	"INSUFFICIENT MEMORY TO LOAD PROGRAM"	BOOT A SYSTEM WHICH WAS SYSGENED FOR SMALLER MEMORY SIZE (UNMAPPED RDOS ONLY)

SUMMARY OF INITIALIZATION ERROR MESSAGES IN ORDER OF (POSSIBLE) APPEARANCE

	MESSAGE	PROBLEM
1.	"OVERLAY FILE VACUOUS"	HIPBOOT DID NOT FOUND SYSTEM OVERLAY FILE FOR INIT CODE
2.	"MASTER DEVICE DRIVER NOT LOADED SYSGEN ERROR"	INIT CODE COULD NOT FIND DCB FOR MASTER DEVICE. (COULD BE SECONDARY PARTITION RECOGNITION PROBLEM SPECIFY AS DPØ:SECPART:FILENAME)
3.	"NO SPARE DCB AVAILABLE - SYSGEN ERROR"	CONFUCIUS SAY "MAN WHO BOOT FROM SEC-PARTITION WITH NO INITIALIZABLE DIRECTORIES (QUESTION #8, SYSGEN) SOL."
4.	"INSUFFICIENT MEMORY FOR OPERATING SYSTEM-SYSGEN ERROR	BUFFERS EXTEND THROUGH TOP OF MEMORY. EITHER YOU CUSTOMIZED (INCORRECTLY) RDOS WITHOUT ADJUSTING NREL OR??
5.	"MASTER DEVICE NOT ON LINE"	DISK TIMEOUT ERROR
6.	"MOVING HEAD DISK ERROR"	PERSISTENT CYLINDER ERRORS DURING MOVING HEAD DISK SIZING
7.	"WARNING:MAP.DR IS ERRONEOUS"	SWAPPING BLOCKS RETURNED DURING INIT WERE MARKED (IN MAP.DR) NOT IN USE. GOOD TIME TO DUMP AND DO FULL INIT.

BOOTSTRAP COMMAND LINE INTERPRETER



Section 16

PANICS

ERRORS

P A N I C S

DISAGREEMENT BETWEEN
BOOKKEEPING TABLES -
RATHER THAN CHANCING MAKING
THINGS WORSE BY BELIEVING THE
WRONG ONE,
RDOS GIVES UP!

MESSAGE:

ACO AC1 AC2 AC3 CODE

(RDOS HALTS AT CORE DUMP)

REACTION:

ANALYZE CODE, RECORD ACs

(p 135 HANDBOOK IF FIRST NUMBER=1, THEN PANIC CODE
p 127 HANDBOOK IF = 0, THEN SYSTEM ERROR CODE)

PRODUCE CORE DUMP

((p G-2, RDOS REF. MANUAL))

PANIC CODES

1: NO LONGER USED

2: SYS.DR ERROR

AC2=ADDRESS IN BUFFER THAT

CONTAINS COPY OF BAD UFD.

GET SYSTEM RLDR LIST TO FIND

CLOSEST BUFFER START = "BQ#"

BQ#-4=DISK UNIT NUMBER

BQ#-3=LEFT HALF OF BLOCK ADDRESS

BQ#-2=RIGHT HALF OF BLOCK ADDRESS

DSKED MIGHT REPAIR IT

3: STACK OVERFLOW

CHECK AC3 AGAINST RLDR LIST

NOVA: AC3=RETN +, CODE OVERFLOW

AC3=OVFLOT, INTERRUPT OVERFLOW

ECLIPSE: AC3=SSOVT, CODE OVERFLOW

AC3=OVFLOT, INTERRUPT OVERFLOW

NO WAY YOU CAN FIX THIS - INDICATES

EITHER BAD SYSTEM CODE OR FAULTY

INTERRUPT HARDWARE

PANIC CODES CON'T

4: INCONSISTANT SYSTEM DATA

BOOKKEEPING POINTS BIGGER ADDRESSES THAN EXIST ON DISK. AC2=BUFFER ADDRESS (BQ#). USE VALUES OF PANIC 2 TO FIND DISK BLOCK INVOLVED. OTHER BLOCKS MIGHT ALSO HAVE TO BE SEARCHED. (SEQUENTIAL LINKS, INDEX BLOCKS)

5: MASTER DEVICE DATA ERROR

HARDWARE ERROR WHILE READING SYSTEM FILES. RUN A "READ ONLY" HARDWARE DIAGNOSTIC SET TO IGNORE "COMPARE" ERRORS TO GET EXACT ERROR REPORT WITHOUT DESTROYING DATA ON DISK.

6: MASTER DEVICE TIMEOUT

HARDWARE DIDN'T RESPOND AT ALL AFTER COMMANDED TO TRANSFER SYSTEM FILES. RUN CONTROLLER DIAGNOSTIC ON BLANK DISK.

PANIC CODES CON'T

7: MOVING HEAD DISK ERROR - REV. 4

THIS PANIC (REMOVED FROM REV 5)
INDICATES ERRORS WHILE INIT'ING A
DISK. RETRY INIT A FEW TIMES.
IF STILL NO GO, RUN "READ ONLY"
NO "COMPARE" ERROR DIAGNOSTIC
TO GET ERROR REPORT.

10: UNDEFINED INTERRUPT

AC2=DEVICE CODE OR
FOR ECLIPSE IF AC2=ADDRESS
OF PFDCT FROM RLDR MAP,
INTERRUPT PRIORITY HARDWARE
CHAIN IS BROKEN.

11: NO SUCH PANIC

12: NOT ENOUGH CONTIGUOUS SPACE

TRY INIT'ING A FEW MORE TIMES.
USE DSKED TO KILL SOME FILES
SO OTHERS MIGHT LIVE.

13: RETURN FROM LEVEL ZERO
RETURN TRYED WITH NO SWAP.
USER PROGRAM ERROR.

14: INCONSISTANT IPB DATA
ONLY VALID IN SYSTEMS OF
2 CPUs SHARING A DISK
THROUGH INTER-PROCESSOR BUS.
IF NO IPB IN SYSTEM, MEMORY
HARDWARE FAILURE COULD CAUSE
THIS MESSAGE ERRONEOUSLY.

15: TRAP IN USER INTERRUPT CODE
FIX IT.

16: MULTI-BIT ERCC MEMORY ERROR
ECLIPSE WITH ERROR CHECKING &
CORRECTION MEMORY CAN FIX A
1 BIT ERROR, BUT THIS IS MORE
THAN IT CAN HANDLE.
AC0 = FAULT CODE IN BITS 0-4
RIGHT BIT OF MEM. ADDRESS = BIT 15
AC 1 = LEFT BITS OF MEM. ADDRESS

notes

Section 17

ADDING

A

DEVICE DRIVER

ADDING A DEVICE DRIVER TO RDOS

GENERAL PROCEDURE

A. Build the Device Driver (Lab PTRDR.SR)

1. Define a DCT for device and make DCT name a .ENT.
2. Define a system queue entry and make QCB name a .ENT
3. Define a dispatch table
4. Define an I/O instruction execution area
5. Assemble with MACRO Assembler
   MAC/Z PARU/S RDOS/S PARS/S $LPT/L name
6. Insert driver in RDOSA.LB
   LFE I URDOSA.LB ZRDOSA.LB/O UGSUB/A name

B. Update system file table overlay (SFTAB.SR)

1. Create peripheral device entry with text name
*2. Assemble with MACRO Assembler
   MAC/Z PARU/S RDOS/S PARS/S $LPT/L name
3. LFE R URDOSO.LB ZRDOSO.LB/O USFTA name

C. Update RDOS tables module (TABLE.SR)

1. Declare DCT name & system queue entry name as normal externals
2. Enter DCT name at proper position in Interrupt Vector Table
3. Enter system queue entry in system queue table
*4. Assembly tables with MACRO Assembler
   MAC/Z PARU/S RDOS/S PARS/S $LPT/L name
5. Replace old table in RDOSC.LB with new one
   LFE R URDOSC.LB ZRDOSC.LB/O UTABL name

D. Sysgen a new system with NO LOAD SWITCH (SYSGEN/N)

E. Create Dummy program with normal external corresponding to DCT name to force loading of new driver.

1. Assemble with MACRO Assembler.
2. Insert in load line just before kDOSA.LB

F. Verify system load command line is proper it's in CLI.CM.

G. Modify XXXX/N number in load line to lower number to account for additional locations new driver uses.

H. Perform system load. @CLI.CM@

J. Boot NEW SYSTEM.

*Delete .RB pseudo-op in TABLE.SR & SFTAB.SR

notes