



RTE II/III
REAL TIME
MULTIPROGRAMMING
student workbook



The information contained in this manual is for training purposes only. Consult the Hewlett-Packard documentation supplied with the system for current information covering the specific computer operating system furnished.

The information contained in this publication may not be reproduced in any form without the expressed consent of the Hewlett-Packard Company.

COPYRIGHT © HEWLETT-PACKARD COMPANY 1974

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

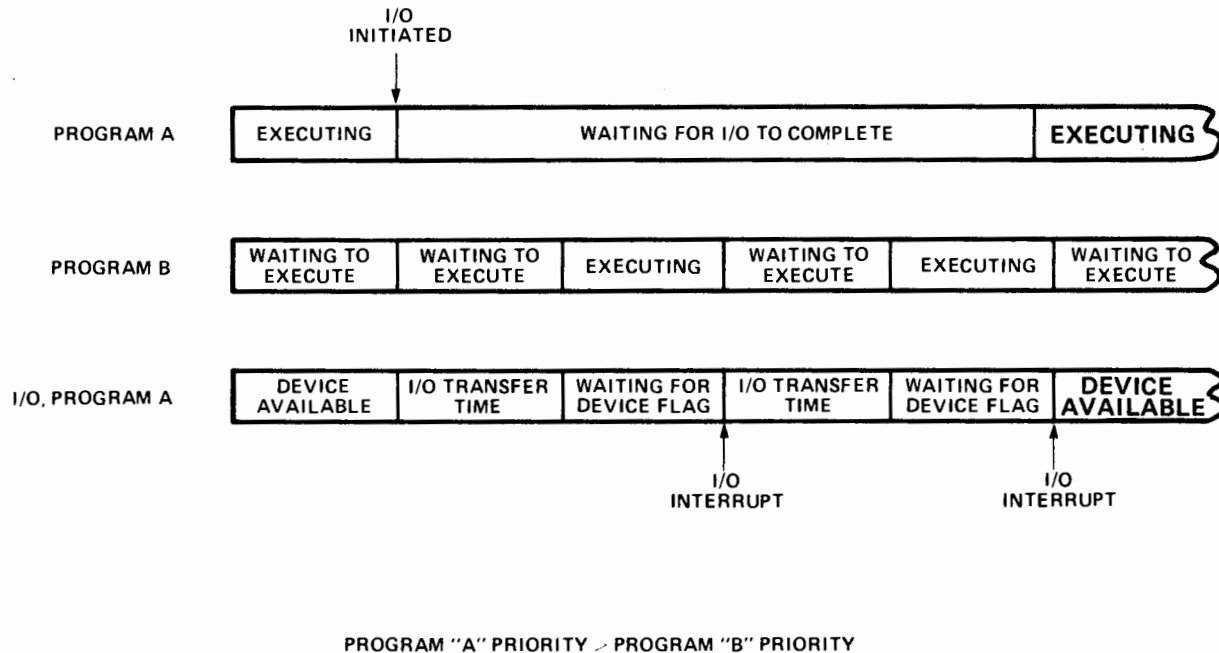
RTE-II/RTE-III

HP REAL-TIME EXECUTIVE SOFTWARE SYSTEM

A MULTIPROGRAMMING SYSTEM THAT ALLOWS
SEVERAL PROGRAMS TO OPERATE CONCURRENTLY;
EACH PROGRAM EXECUTING DURING THE UNUSED
CENTRAL PROCESSOR TIME OF THE OTHERS.

RTEII/III-1

CONCURRENT EXECUTION OF PROGRAMS

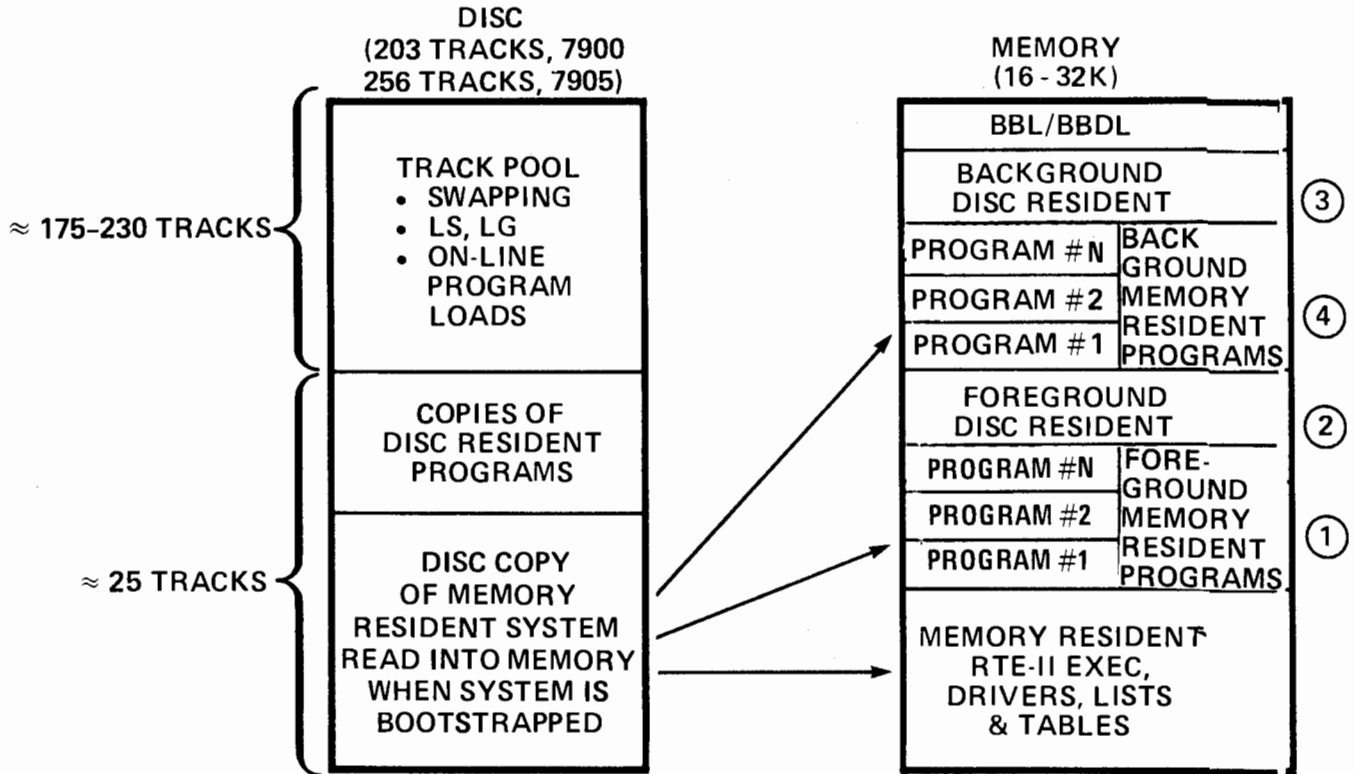


RTEII/III-2

This is but one way a program may *interleave* execution with another. In general, all *active* programs take advantage of any system or program dead time to execute. This dead time includes program *suspension* time, I/O device time and time not used for system housekeeping (approximately 10%).

RTE-II

THE SYSTEM IS INITIATED BY BOOTSTRAPPING THE SYSTEM INTO MEMORY FROM THE DISC

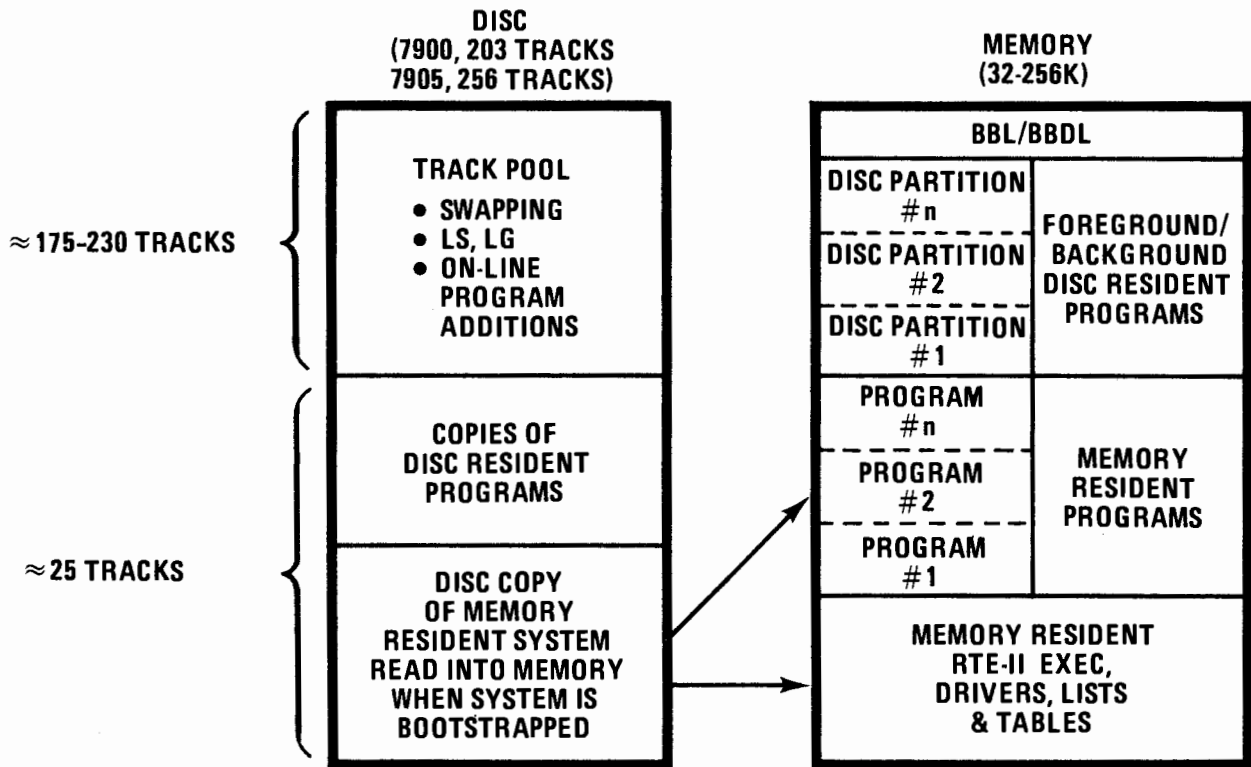


PROGRAMS MAY THEN BE SCHEDULED TO EXECUTE IN UP TO 4 SEPARATE PARTITIONED AREAS OF MEMORY

RTEII/III-3

RTE-III

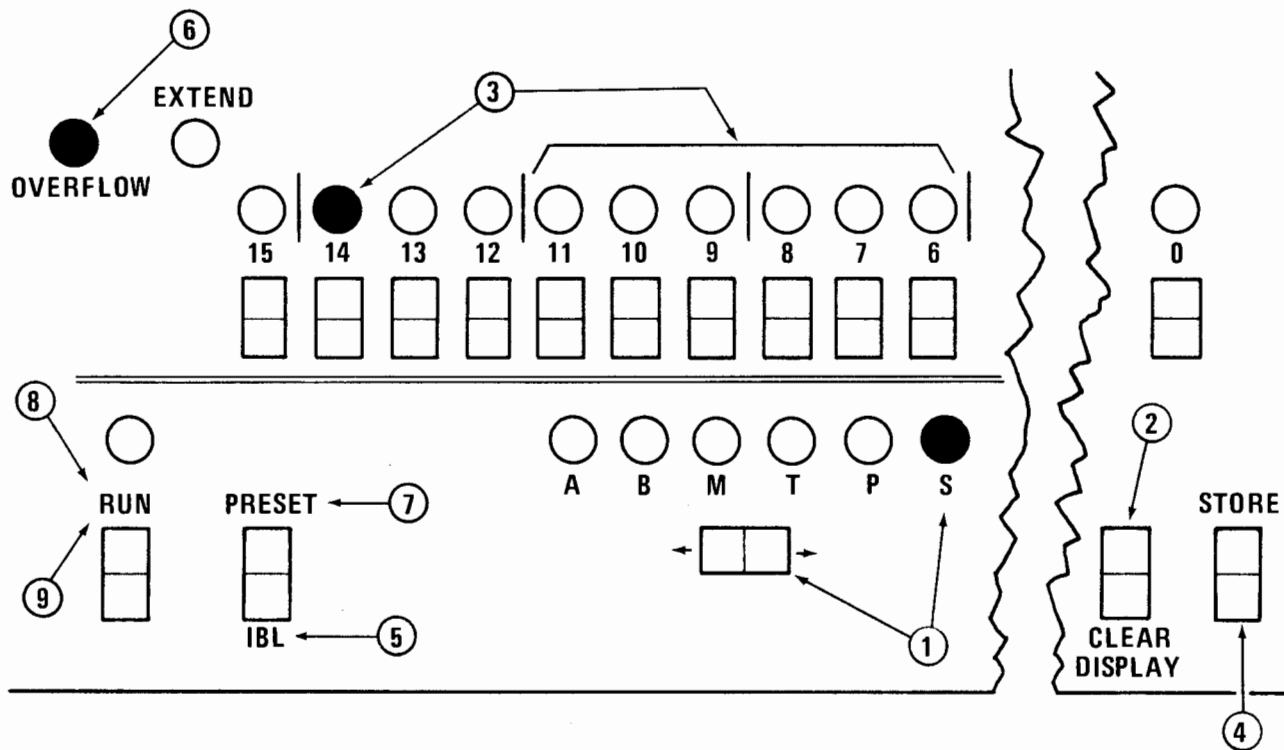
THE SYSTEM IS INITIATED BY BOOTSTRAPPING IT INTO MEMORY FROM THE DISC.



PROGRAMS MAY THEN BE SCHEDULED TO EXECUTE IN UP TO 64 SEPARATE PARTITIONED AREAS OF MEMORY

RTEII/III-4

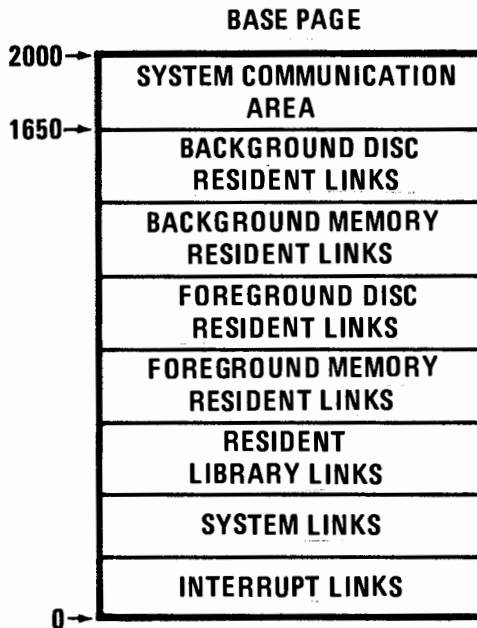
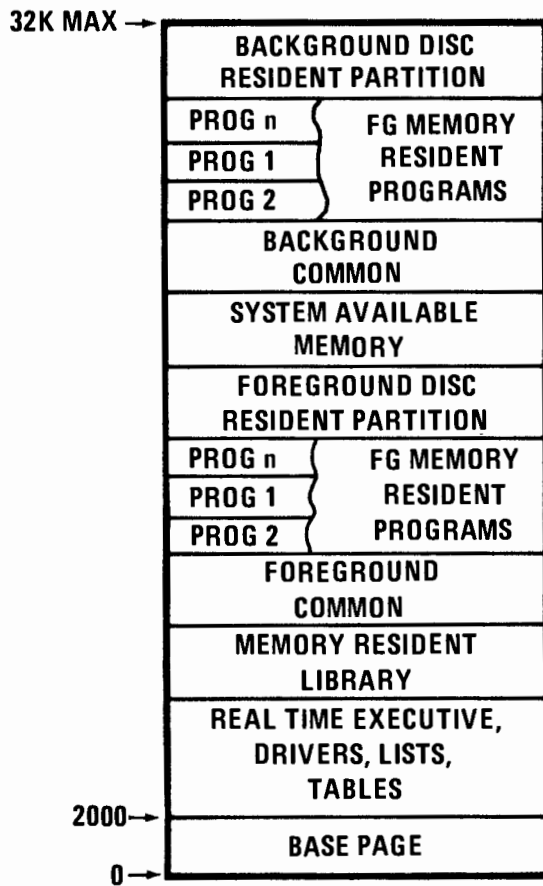
RTE SYSTEM START-UP PROCEEDURE



RTEII/III-5

- 1 Select S Register
- 2 Clear Display Register
- 3 Set 6-10 to Disc Select Code, 14 ON
- 4 Press STORE
- 5 Press IBL 6 Overflow should light
- 7 Press PRESET
- 8 Press RUN. Computer should halt with 102077 in Display Register
- 9 Press RUN. System should commence executing

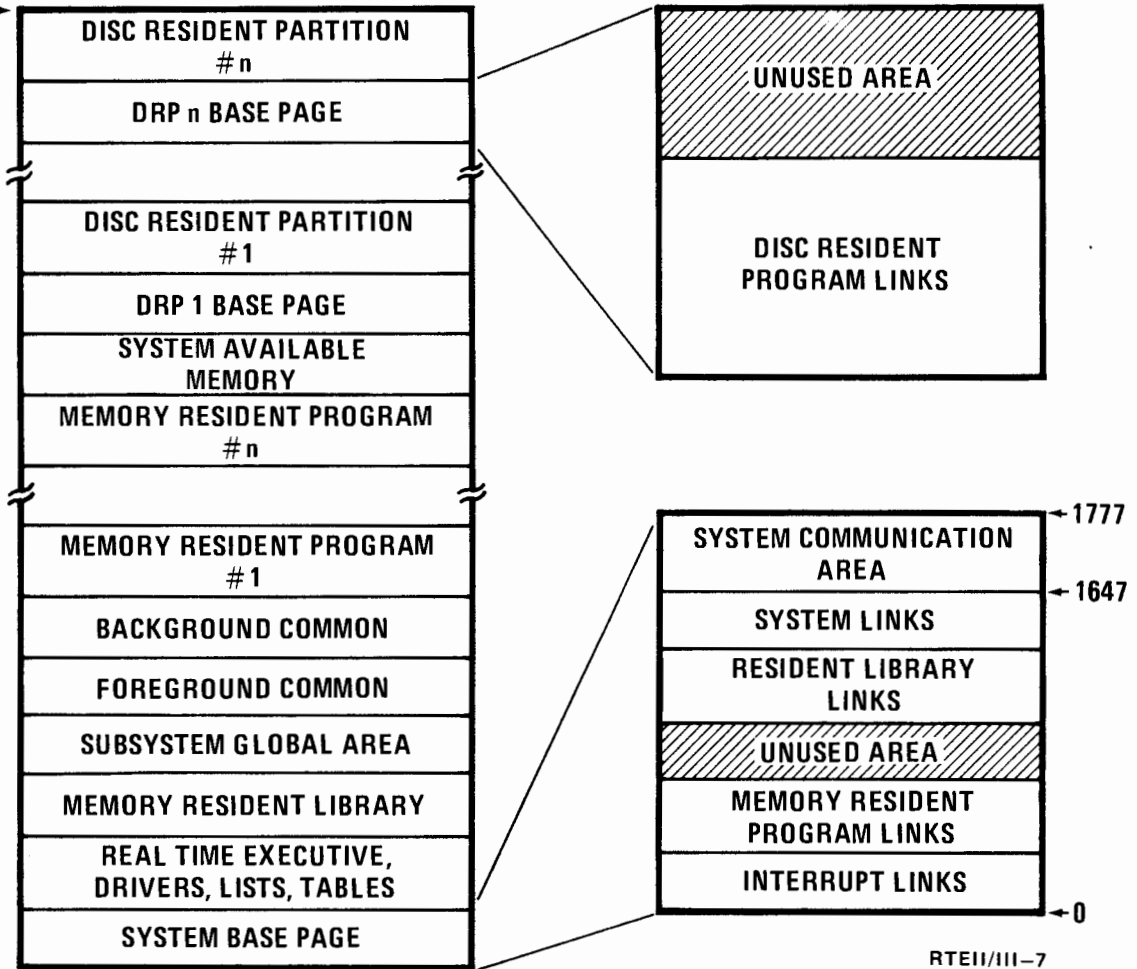
RTE-II PHYSICAL SYSTEM MEMORY MAP



RTEII/III-6

RTE-III PHYSICAL SYSTEM MEMORY MAP

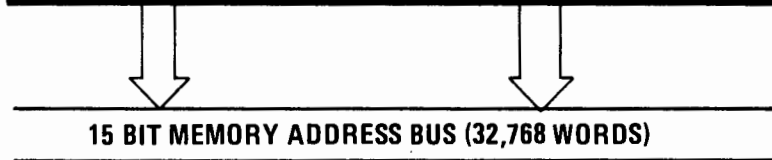
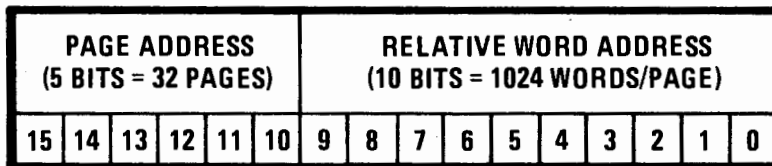
256K MAX →



RTEII/III-7

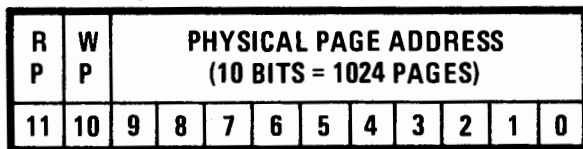
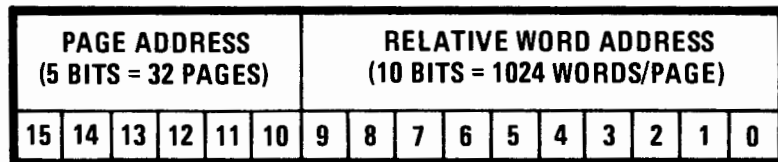
MEMORY ADDRESSING

RTE II



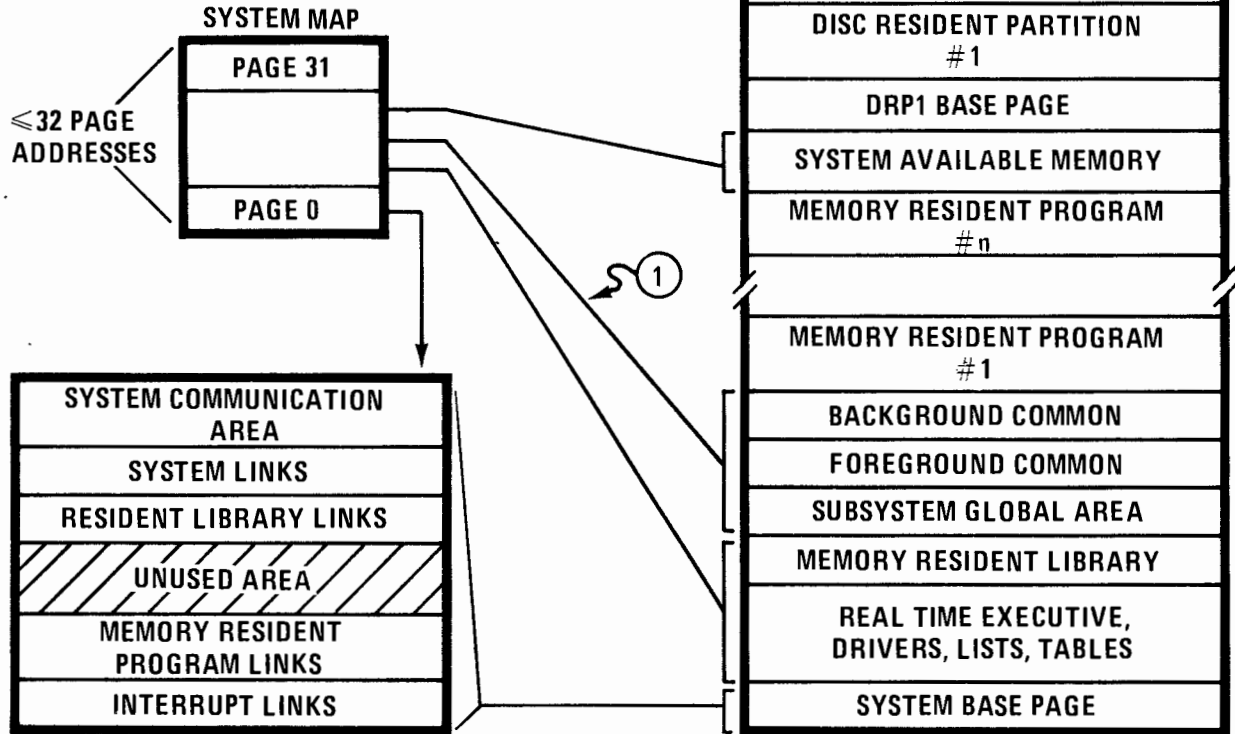
RTE III

MEMORY MAP
(1 OF 32)



RTEII/III-8

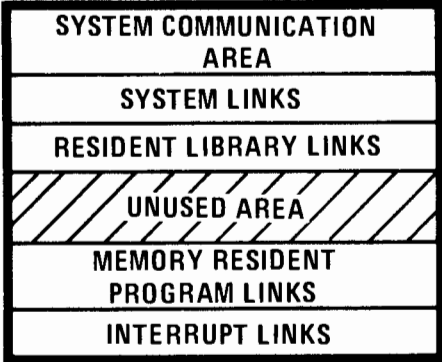
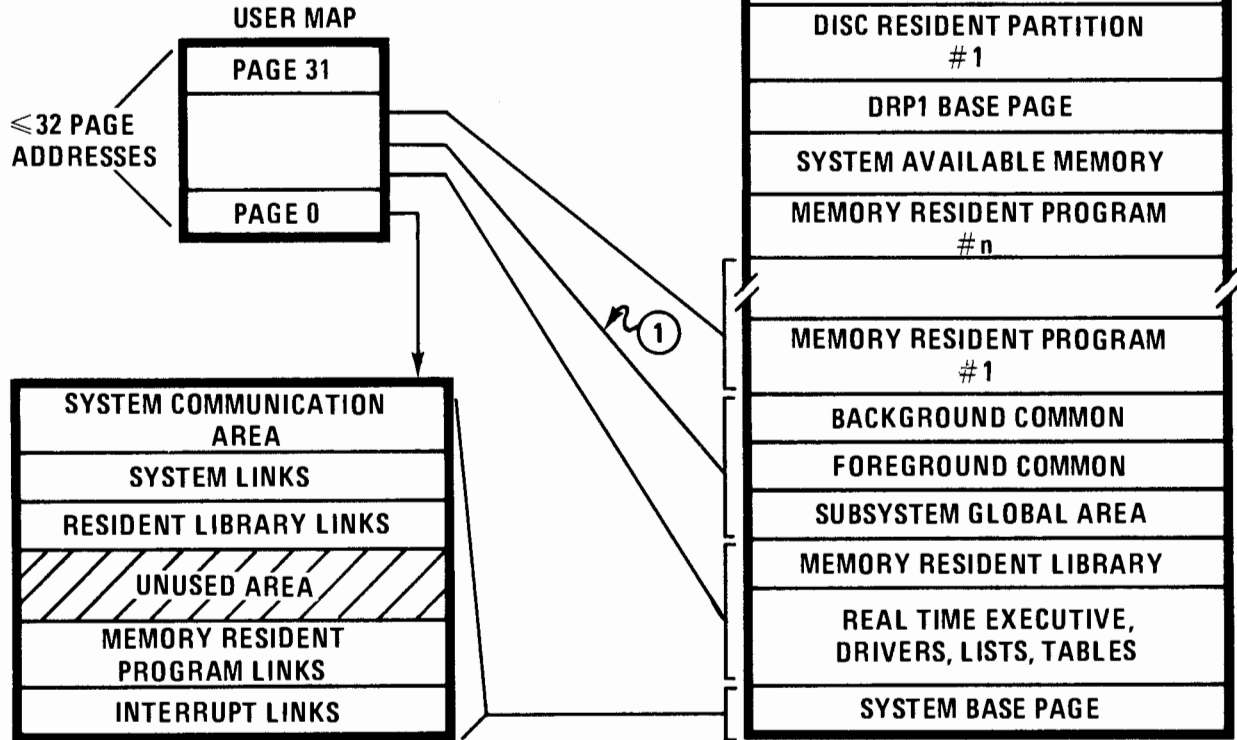
RTE-III SYSTEM MAP ADDRESS AREA



① INCLUDED IF PRIVILEGED DRIVER ACCESS SPECIFIED

RTEII/III-9

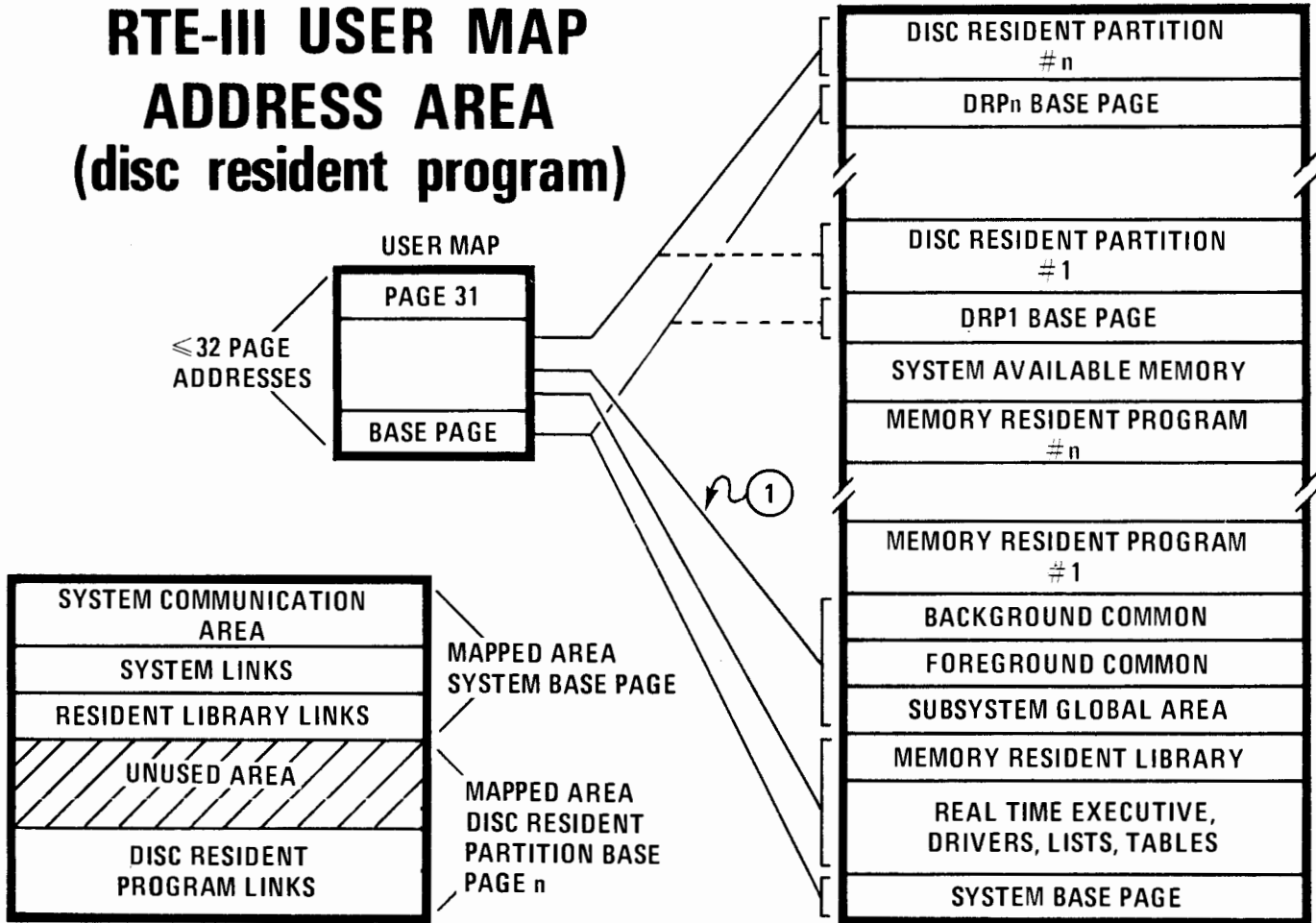
RTE-III USER MAP ADDRESS AREA (memory resident program)



① INCLUDED IF A PROGRAM DECLARES COMMON

RTEII/III-10

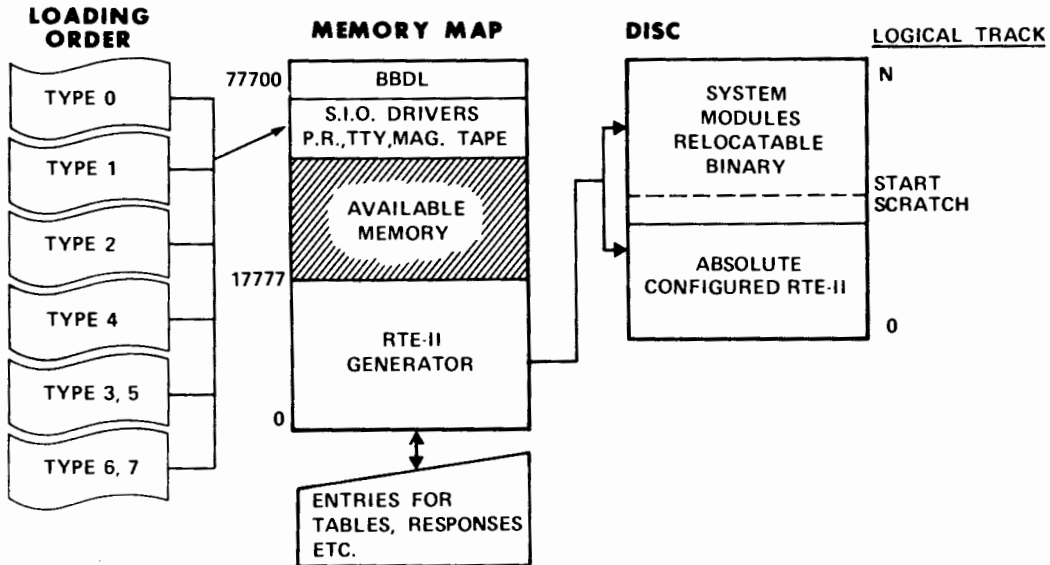
RTE-III USER MAP ADDRESS AREA (disc resident program)



① INCLUDED IF PROGRAM DECLARES COMMON

RTEII/III-11

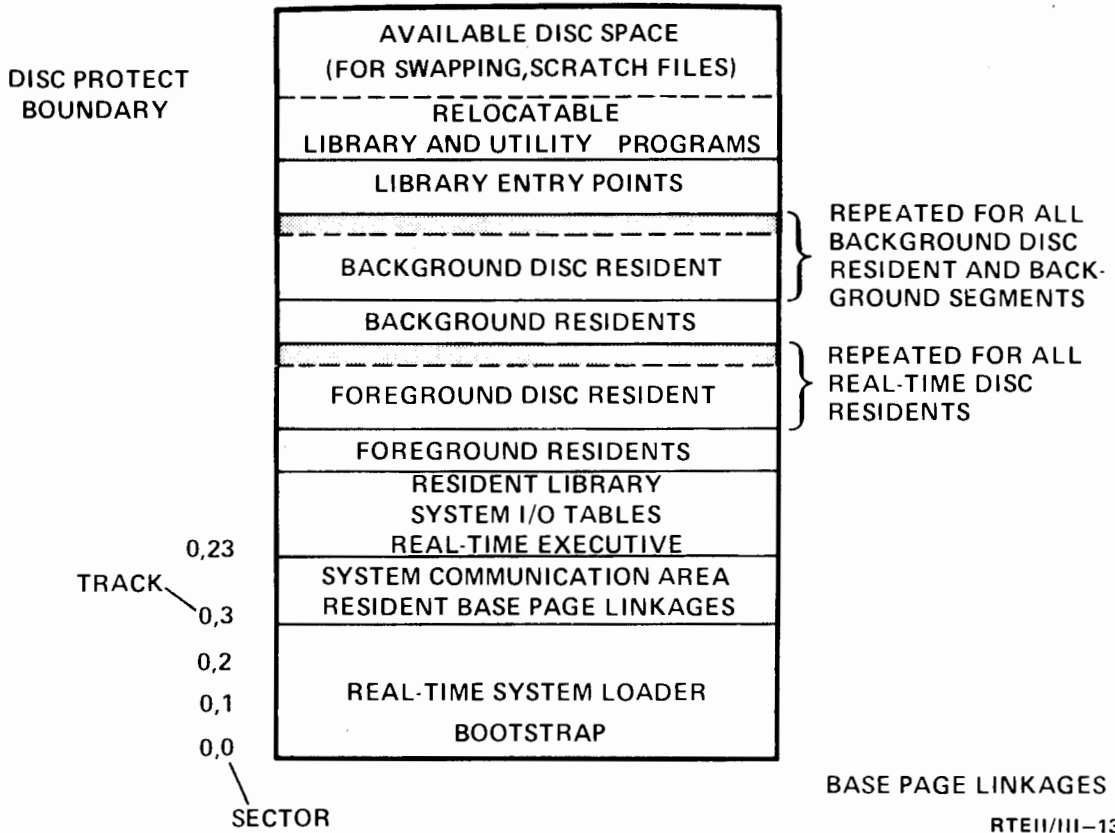
SYSTEM GENERATION (TAPE)



- THE SYSTEM GENERATOR IS LOADED INTO MEMORY USING THE BBL.
- INITIALIZATION PHASE - ESTABLISHES DISC SIZE, TYPE, SYSTEM HARDWARE INFO.
- PROGRAM INPUT PHASE - SYSTEM AND USER PROGRAMS ARE COPIED ON THE DISC.
- PARAMETER INPUT PHASE - PROGRAM PRIORITIES AND TYPE CODES MAY BE CHANGED.
- DISC LOADING PHASE - ALL TABLES ARE CONSTRUCTED AND THE ABSOLUTE SYSTEM IS CREATED ON THE SYSTEM DISC/DRUM.

RTEII/III-12

RTE-II/RTE-III CONFIGURED SYSTEM DISC MAP



MEMORY RESIDENT PROGRAMS

MEMORY RESIDENT – TYPE 1 (RTE-III)

BACKGROUND (REAL TIME) MEMORY RESIDENT – TYPE 1 (RTE-II)

BACKGROUND MEMORY RESIDENT – TYPE 4 (RTE-II)

- ARE ALWAYS IN CORE
- RESPOND QUICKLY TO REAL TIME CONDITIONS ($\approx 100 \mu\text{sec}$)
- USED FOR HIGH PRIORITY TASKS

BOTH TYPES ARE IDENTICAL EXCEPT THEY NORMALLY USE DIFFERENT SYSTEM COMMON AREAS

RTEII/III-14

DISC RESIDENT PROGRAMS



BACKGROUND DISC RESIDENT – TYPE 2

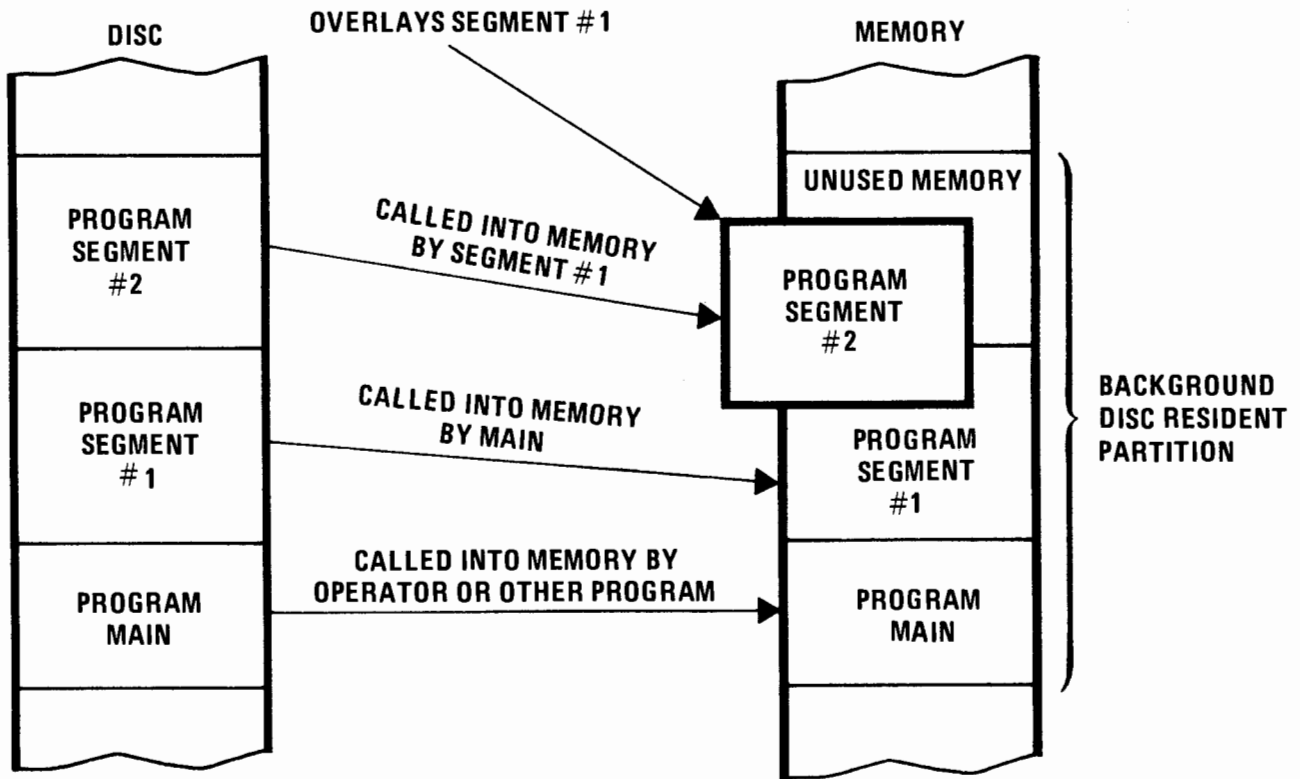
BACKGROUND DISC RESIDENT – TYPE 3, 5

- RESIDE ON THE DISC IN ABSOLUTE FORM
- RESPOND SLOWER TO REAL TIME CONDITIONS (≈ 50 msec) THAN MEMORY RESIDENT PROGRAMS
- ARE SWAPPABLE BY THE SYSTEM

BOTH AREAS ARE IDENTICAL EXCEPT THEY NORMALLY USE DIFFERENT SYSTEM COMMON AREAS. BACKGROUND DISC PROGRAMS MAY BE SEGMENTED (A TYPE 5 PROGRAM IS A SEGMENT) THUS LARGE UTILITY PROGRAMS (COMPILERS, ETC.) ARE OF THIS TYPE.

RTEII/III-15

SEGMENTED PROGRAMS (type 3,5)



RTEII/III-16

STANDARD RTE UTILITY PROGRAMS (TYPE 3,5)

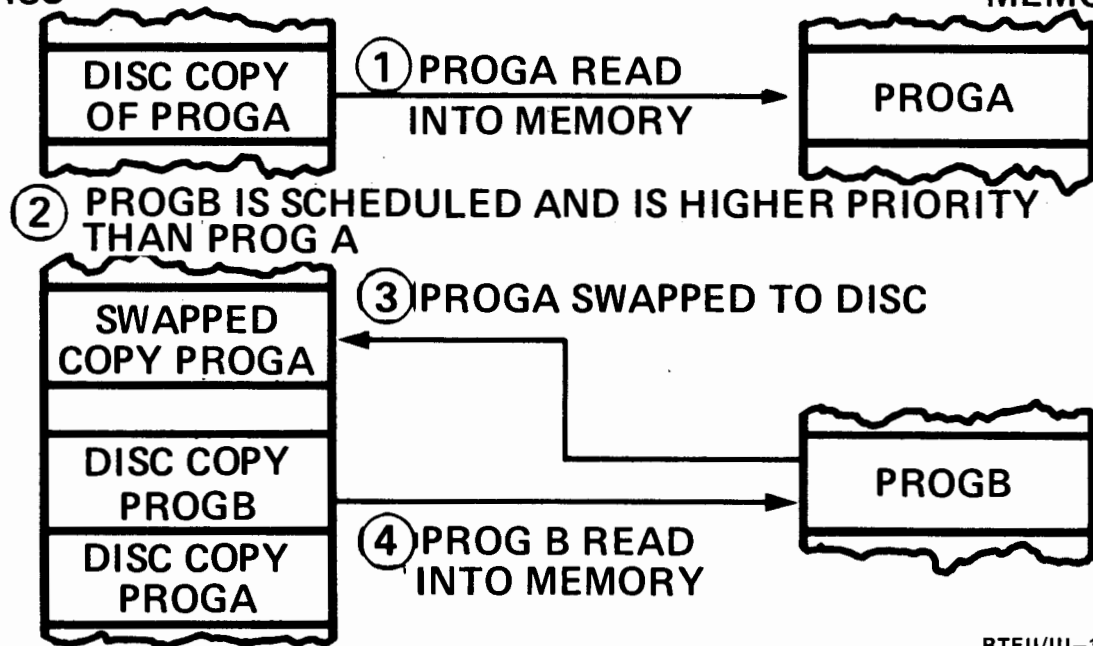
- ASSEMBLER
- FORTRAN COMPILER
- FORTRAN IV COMPILER
- ALGOL COMPILER
- SYMBOLIC EDITOR
- INTERACTIVE EDITOR
- RELOCATING LOADER
- BATCH/SPOOL MONITOR – FILE MANAGER
- GASP

RTEII/III-17

PROGRAM SWAPPING

DISC RESIDENT PROGRAMS MAY BE SWAPPED TO THE DISC BY THE SYSTEM TO ALLOW HIGHER PRIORITY DISC RESIDENT PROGRAMS TO EXECUTE.

PROGA HIGHEST PRIORITY IN ACTIVE (SCHEDULED) STATE
DISC MEMORY



RTEII/III-18

A FOREGROUND OR BACKGROUND DISC RESIDENT PROGRAM MAY NOT BE SWAPPED IF:

- **IT IS DOING I/O TO/FROM AN AN INTERNAL BUFFER**
- **IT HAS LOCKED ITSELF INTO CORE**

RTEII/III-19

PARTITION USAGE

- **MAY BE ALL FOREGROUND, ALL BACKGROUND OR A MIXTURE.**
- **FOREGROUND PROGRAMS RUN IN FOREGROUND PARTITIONS, BACKGROUND PROGRAMS RUN IN BACKGROUND PARTITIONS.**
- **IF ONE TYPE OF PARTITION IS DEFINED, ALL PROGRAMS WILL RUN IN THAT TYPE OF PARTITION.**
- **PROGRAMS MAY BE ASSIGNED TO A PARTITION.**

RTEII/III-20

COMMON

SYSTEM COMMON

- IS EXTERNAL TO THE PROGRAM.
- IS ALWAYS IN MEMORY (NOT SWAPPED WITH PROGRAM).
- ALLOWS PROGRAMS TO PASS EACH OTHER DATA.
- ALLOWS DISC RESIDENT PROGRAMS TO SAVE DATA WHILE DORMANT.

THERE IS ONE COMMON AREA FOR FOREGROUND PROGRAMS AND ONE FOR BACKGROUND PROGRAMS.

*REVERSE COMMON

- FOREGROUND PROGRAMS USE BACKGROUND COMMON
- BACKGROUND PROGRAMS USE FOREGROUND COMMON

LOCAL COMMON

- IS WITHIN A PROGRAM
- IS SWAPPED WITH THE PROGRAM
- USEFUL FOR PASSING DATA TYPE 3&5, PROGRAMS

* ATTEMPTED DATA STORAGE FROM A BACKGROUND PROGRAM INTO FOREGROUND COMMON CAUSES A MEMORY PROTECT VIOLATION.

RTEII/III-21

SUBROUTINES (OF THE RELOCATABLE LIBRARY)

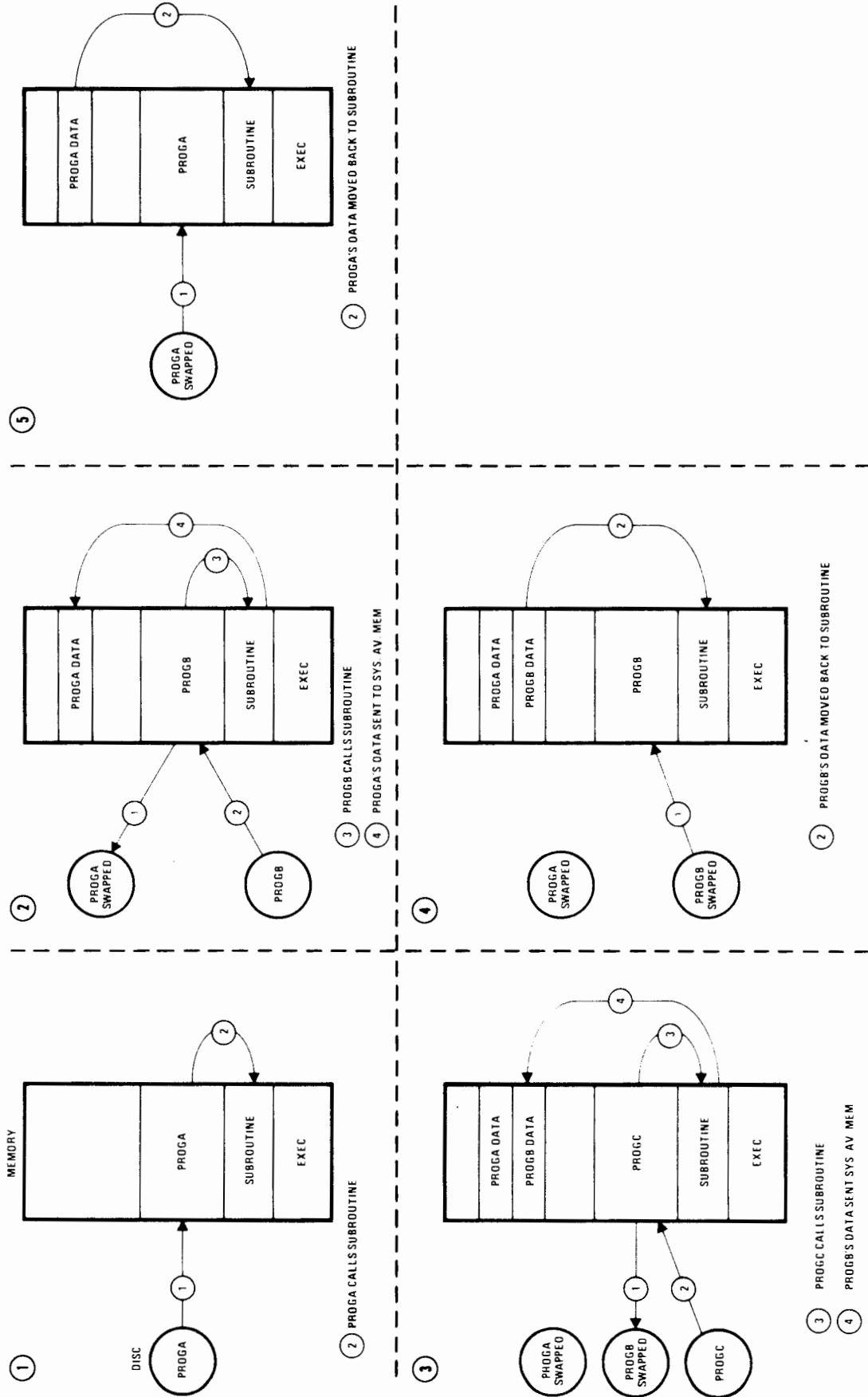
- RE-ENTRANT – CAN BE INTERRUPTED
 - PRIVILEGED – CANNOT BE INTERRUPTED
 - UTILITY – APPENDED TO EACH PROGRAM THAT CALLS IT
- } TYPE 6 PROGRAMS
- TYPE 7 PROGRAMS

AT SYSTEM GENERATION TIME:

- TYPE 6 ROUTINES CALLED BY MEMORY RESIDENT PROGRAMS, OR IF FORCED BY THE OPERATOR, ARE PUT IN THE SYSTEM MEMORY RESIDENT LIBRARY.
- TYPE 7 ROUTINES ARE APPENDED TO EACH PROGRAM THAT CALLS THEM.
- ANY TYPE 6 OR 7 ROUTINES THAT WERE NOT CALLED ARE PUT ON THE DISC FOR LATER USE BY THE ON-LINE LOADER.

RTEII/III-22

RE-ENTRANT PROCESSING



PROGRAM STATES

- DORMANT
- SCHEDULED
- EXECUTING
- SUSPENDED
- GENERAL WAIT

RTEII/III-24

PROGRAM STATES

PROGRAMS IN THE SCHEDULED, SUSPENDED AND
GENERAL WAIT STATE ARE ORDERED BY PRIORITY.

PRIORITIES RANGE FROM 1 (THE HIGHEST) TO 32767
(THE LOWEST). MORE THAN ONE PROGRAM MAY BE
AT THE SAME PRIORITY.

RTEII/III-25

PROGRAM STATES

FOR EACH PROGRAM STATE, EXCEPT EXECUTING, RTE MAINTAINS AN ORDERED LIST OF THE PROGRAMS IN THAT STATE, ACCORDING TO THE PRIORITY OF THE PROGRAMS.

THERE ARE *ONE DORMANT LIST, ONE SCHEDULED LIST, ONE GENERAL WAIT LIST, AND FOUR TYPES OF SUSPENSION LISTS:

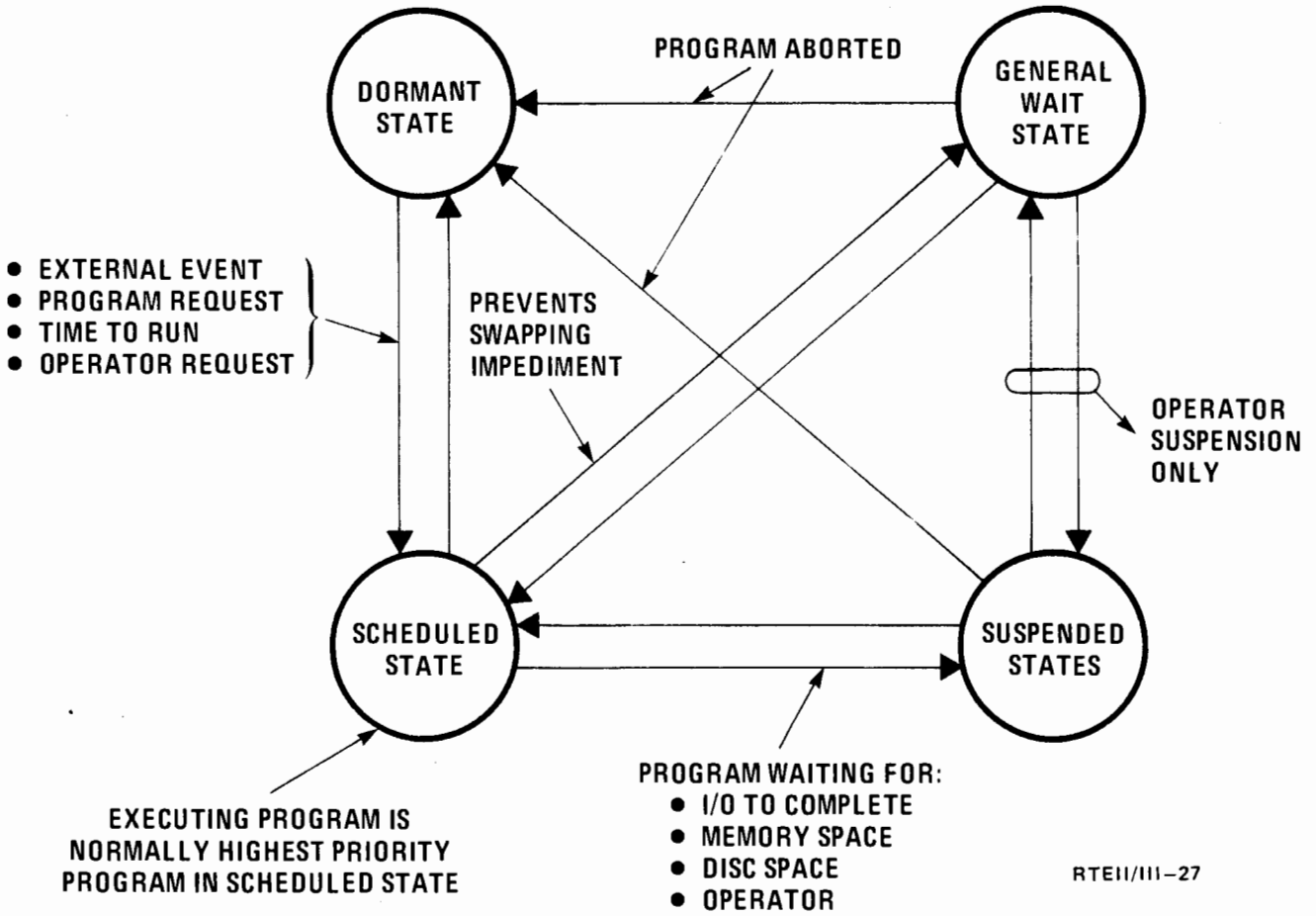
- I/O SUSPENSION LISTS (ONE FOR EACH DEVICE)
- UNAVAILABLE MEMORY LIST
- OPERATOR SUSPENSION LIST
- DISC ALLOCATION LIST

PROGRAMS SCHEDULED BY THE COMPLETION OF A TIME INTERVAL WILL BE IN AN ADDITIONAL LIST, THE TIME LIST.

* RTE-II DOES NOT MAINTAIN A DORMANT LIST

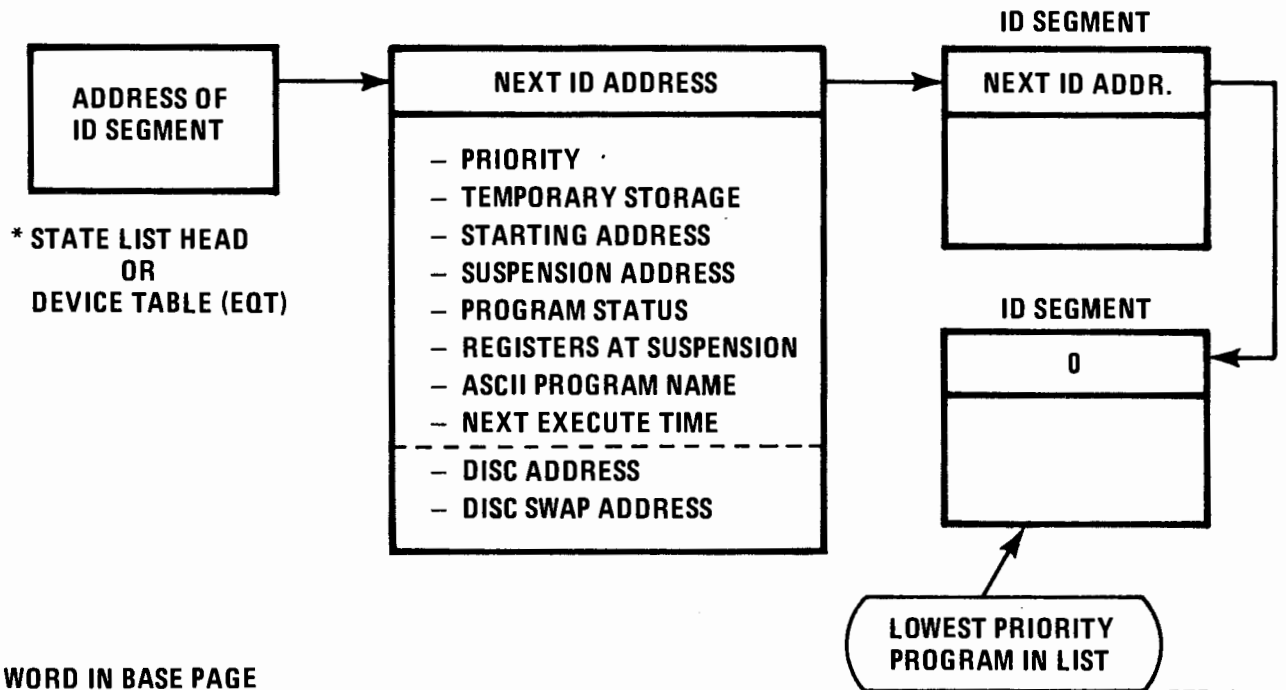
RTEII/III-26

PROGRAM STATES



ID SEGMENTS

A PROGRAM EXISTS IN THE SYSTEM BECAUSE AN ID SEGMENT IS FILLED OUT FOR IT



RTEII/III-28

DORMANT STATE

DORMANT PROGRAMS ARE INACTIVE,

I.E.,

THEY ARE NOT SCHEDULED, EXECUTING
OR SUSPENDED.

RTEII/III-29

SCHEDULED STATE

SCHEDULED PROGRAMS ARE WAITING FOR COMPUTER EXECUTION TIME (EXECUTING STATE).

PROGRAMS IN THE SCHEDULED STATE WAIT FOR EXECUTION TIME ACCORDING TO THEIR PRIORITY.

PROGRAMS MAY BECOME SCHEDULED BY:

- OPERATOR REQUEST
- THE SYSTEM REAL-TIME CLOCK
- *• ANOTHER PROGRAM
- AN EVENT CAUSING AN INTERRUPT

RTEII/III-30

* The scheduling program may or may not wait, at the programmer's option, for the program it scheduled to complete execution.

EXECUTING STATE

THE HIGHEST PRIORITY SCHEDULED PROGRAM WILL BE
IN THE EXECUTING STATE.

A PROGRAM EXECUTES UNTIL:

- IT REACHES COMPLETION.
- IT IS RE-SCHEDULED -- A HIGHER PRIORITY PROGRAM WAS SCHEDULED.
- IT IS SUSPENDED.
- IT IS TERMINATED BY ANOTHER PROGRAM.
- IT IS TERMINATED BY OPERATOR REQUEST.
- IT IS ABORTED BY THE SYSTEM DUE TO IT PERFORMING AN ILLEGAL OPERATION.

RTEII/III-31

SUSPENDED STATE

A SUSPENDED PROGRAM IS INACTIVE, AND ITS CURRENT STATE OF EXECUTION IS PRESERVED.

WHEN A PROGRAM BECOMES SUSPENDED, THE NEXT HIGHER PRIORITY PROGRAM IS EXECUTED.

A SUSPENDED PROGRAM IS RE-SCHEDULED FOR EXECUTION WHEN THE REASON FOR ITS SUSPENSION IS REMOVED.

A PREVIOUSLY SUSPENDED PROGRAM CONTINUES EXECUTION FROM ITS POINT OF SUSPENSION.

RTEII/III-32

SUSPENDED STATE

A PROGRAM MAY BECOME SUSPENDED BECAUSE:

- IT REQUESTED AN I/O OPERATION.
- IT REQUIRES MEMORY AND NONE IS AVAILABLE.
- THE OPERATOR REQUESTED IT BE SUSPENDED.
- IT REQUESTED IT BE SUSPENDED.
- IT IS WAITING COMPLETION OF A DISC ALLOCATION.

RTEII/III-33

GENERAL WAIT STATE

WAITING PROGRAMS ARE IN ONE OF 5 SUBSTATES.
IN GENERAL THIS ALLOWS IMPEDED DISC RESIDENT
PROGRAMS TO BE SWAPPED.

PROGRAMS IN THIS STATE ARE WAITING FOR:

- A DOWNED DEVICE TO BE ENABLED
- ANOTHER PROGRAM IT SCHEDULED TO GO DORMANT
- A CLASS NUMBER ALLOCATION
- A CLASS I/O GET CALL TO COMPLETE
- A RESOURCE NUMBER ALLOCATION
- A RESOURCE NUMBER LOCK

RTEII/III-34

OPERATOR REQUESTS

THE OPERATOR COMMUNICATES WITH THE RTE SYSTEM USING REQUESTS ENTERED THROUGH THE TELEPRINTER KEYBOARD.

THE OPERATOR MAY REQUEST THE SYSTEM TO:

- TURN PROGRAMS ON AND OFF
- SUSPEND AND RESTART PROGRAMS
- SCHEDULE PROGRAMS TO EXECUTE AT SPECIFIC TIMES
- CHANGE THE PRIORITY OF PROGRAMS
- EXAMINE THE STATUS OF ANY PARTITION, PROGRAM, OR I/O DEVICE

RTEII/III-35

- DYNAMICALLY ALTER THE I/O STRUCTURE AND BUFFERING DESIGNATIONS.
- EXAMINE AND DYNAMICALLY ALTER AN I/O DEVICE'S TIME-OUT PARAMETER.
- DECLARE I/O DEVICES UP OR DOWN.
- INITIALIZE THE REAL-TIME CLOCK AND PRINT THE TIME.
- ALLOW/DISALLOW PROGRAM SWAPPING.
- SET DEVICE BUFFERING LIMITS.
- SET UP TEMPORARY RELOCATABLE AND SOURCE FILES.
- RELEASE DISC TRACKS ASSIGNED TO DORMANT PROGRAMS.

RTEII/III-36

MULTIPLE TERMINAL OPERATION



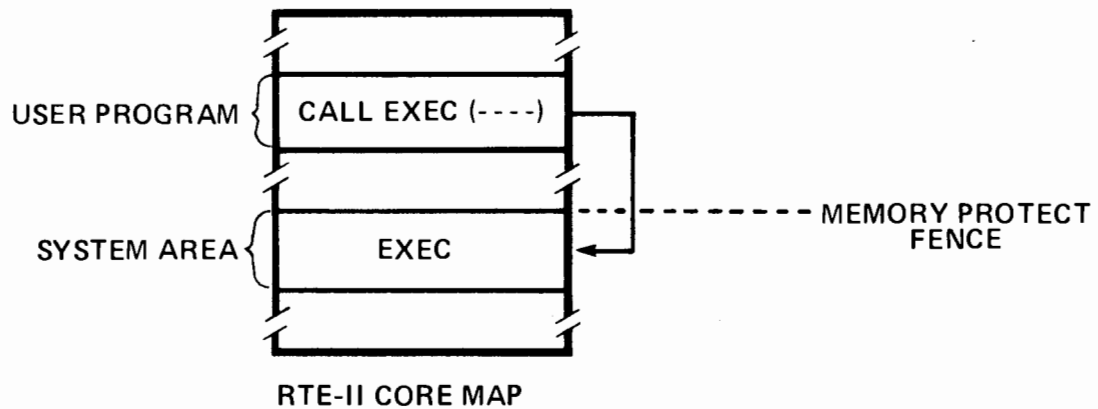
PERMITS MORE THAN ONE OPERATOR CONSOLE FOR:

- ENTERING OPERATOR REQUESTS
- PROGRAM EDITING
- PROGRAM SOURCE CREATION THROUGH THE FILE MANAGER

RTEII/III-37

EXEC COMMUNICATION (EXEC)

USER PROGRAMS COMMUNICATE WITH THE SYSTEM BY FORCING A MEMORY PROTECT VIOLATION INTERRUPT TO OCCUR.



RTE-II RECOGNIZES THIS AND TRANSFERS TO EXEC.

EXEC EXAMINES THE CALL PARAMETERS AND HANDLES THE REQUEST,
OR CALLS A PROCESSOR TO HANDLE IT.

RTEII/III-38

**EXEC CALLS ARE THE LINE OF COMMUNICATION
BETWEEN AN EXECUTING PROGRAM AND RTE.**

USING EXEC CALLS A PROGRAM IS ABLE TO:

- **PERFORM INPUT AND OUTPUT OPERATIONS (WITH OR WITHOUT WAIT).**
- **TERMINATE OR SUSPEND ITSELF OR PROGRAMS IT SCHEDULED.**
- **SCHEDULE OTHER PROGRAMS.**
- **OBTAIN THE TIME OF DAY.**
- **SET PROGRAM EXECUTION TIME CYCLES.**
- **LOAD ITS SEGMENTS (IF BACKGROUND DISC TYPE).**
- **ALLOCATE AND RELEASE DISC TRACKS.**
- **LOCK ITSELF INTO MEMORY IF DISC RESIDENT.**

RTEII/III-39

INPUT/OUTPUT PROCESSING

- ALLOWS REFERENCING I/O DEVICES BY LOGICAL UNIT NUMBER.
- STACKS I/O REQUESTS FOR A DEVICE BY PRIORITY OF THE CALLING PROGRAM.
- PROVIDES BUFFERING FOR I/O DEVICES.
- ALLOCATES DMA CHANNELS TO I/O DEVICES.

RTEII/III-40

When a program performs I/O it is suspended, unless the I/O device is buffered. If the I/O device is buffered, I/O is concurrent with program execution. If the I/O device is not buffered, the requesting program is suspended and the next lower priority scheduled program is executed.

Programs doing class I/O (I/O without WAIT) are not I/O suspended because all class I/O input, output and control operations are buffered.

INTERRUPT PROCESSING

RTE-II/III IS AN INTERRUPT DRIVEN OPERATING SYSTEM

I.E.,

ALL SYSTEM ACTIONS RESULT FROM THE SYSTEM
RECOGNIZING AN INTERRUPT.

THE PRIMARY INTERRUPT SOURCES ARE:

- THE TIME BASE GENERATOR
- I/O DEVICES
- MEMORY PROTECT VIOLATIONS

RTEII/III-41

INTERRUPT PROCESSING

ALL INTERRUPTS ARE DIRECTED TO A CENTRAL
INTERRUPT CONTROL ROUTINE WHICH:

- SAVES AND RESTORES VARIOUS REGISTERS
- ANALYZES THE SOURCE OF THE INTERRUPT
- CALLS THE APPROPRIATE PROCESSING ROUTINE

RTEII/III-42

OPERATOR REQUESTS

THE OPERATOR GAINS THE ATTENTION OF THE EXECUTIVE BY DEPRESSING ANY KEY ON THE SYSTEM CONSOLE OR MTM TERMINAL KEYBOARD. THE SYSTEM RESPONDS BY DISPLAYING A SINGLE ASTERISK (SYSTEM CONSOLE) OR XX> (MTM TERMINAL, XX = LU NO.) AT THAT POINT THE OPERATOR MAY REQUEST ANY ONE OF 22 OPERATIONS. ALL OPERATOR REQUESTS HAVE A TWO CHARACTER FORMAT AND UP TO SEVEN ADDITIONAL PARAMETER FIELDS, SEPARATED BY COMMAS.

FOR EXAMPLE

OPERATOR REQUEST TO RESET THE R-T CLOCK

*TM, 1970, 001, 00, 00, 00 (CR)
YEAR DAYS HOURS MINUTES SECONDS

IF A MISTAKE IS MADE ON INPUT, A "RUBOUT" WILL DELETE THE LINE (PRINTS ""). TO DELETE A SINGLE INCORRECT CHARACTER, DEPRESS THE "CONTROL" KEY AND THE "A" KEY SIMULTANEOUSLY (PRINTS "←").

RTEII/III-43

RUn

SCHEDULES A PROGRAM FOR IMMEDIATE EXECUTION.

RU, program [, P1[, ...[, P5]]]]]

P1 - P5 MAY BE:

DECIMAL

$-32768 \leq P_n \leq 32767$

OCTAL

$0 \leq P_n \leq 177777B$

ASCII

P_n = CHARACTER 1 CHARACTER 2

RTEII/III-44

This command will not affect a program's entry in the time list.

EXAMPLE

FTN,L

50121
PROGRAM PROGA
DIMENSION IPRAM(5) *I PRAM=1:0*
CALL RMPAR(IPRAM) *I PRAM=1*
WRITE(1,100)(IPRAM(I),I=1,5)
100 FORMAT(5I6)
END
END*

*RU .PROGA, -32768, 32767, 17777B, AS, C

-32768 32767 -1 16723 17184

RTEII/III-45

MTM EXAMPLE

FTN,L

```
PROGRAM PROGB
DIMENSION IPRAM(5)
CALL RMPAR(IPRAM)
WRITE(9,100)(IPRAM(I),I=1,5)
100 FORMAT(5I6)
END
ENDS
```

IF FIRST PARAMETER
IS NOT SUPPLIED OR
IS 0, IT DEFAULTS TO
TERMINAL'S LU NO.

09>ON,PROGB

9 0 0 0 0

09> ON,PROGB
ZERO IS PASSED IF SPACE HERE

0 0 0 0 0

09>ON,PROGB,5

5 0 0 0 0

FTN,L

```
PROGRAM PROGA
DIMENSION IPRAM(5)
CALL RMPAR(IPRAM)
K=IPRAM
IF(IPRAM.EQ.0) K=1
WRITE(K,100)(IPRAM(I),I=1,5)
100 FORMAT(5I6)
END
ENDS
```

TECHNIQUE FOR
GENERALIZING A
PROGRAM FOR
ALL MTM TERMINALS.

RTEII/III-46

STatus (RTE-II)

TO REQUEST THE NAME OF THE CURRENTLY EXECUTING PROGRAM, THE NAMES OF DISC RESIDENT PROGRAMS THAT ARE IN CORE OR THE STATUS OF A NAMED PROGRAM.

ST $\left[\begin{array}{c} \text{program} \\ \emptyset \\ 1 \\ , 2 \end{array} \right]$

program – PRINT, PRIORITY STATE AND TIME VALUES FOR PROGRAM.

\emptyset – PRINT NAME OF CURRENTLY EXECUTING PROGRAM. \emptyset IF NONE.

1 – PRINT NAME OF IN-CORE FOREGROUND DISC RESIDENT PROGRAM. \emptyset IF NONE.

2 – PRINT NAME OF IN-CORE BACKGROUND DISC RESIDENT PROGRAM. \emptyset IF NONE.

RTEII/III-47

STatus (RTE-III)

TO REQUEST THE NAME AND PARTITION NUMBER OF THE CURRENTLY EXECUTING PROGRAM, THE NAME OF THE PROGRAM IN A SPECIFIED PARTITION OR THE STATUS OF A NAMED PROGRAM.

ST, $\left[\begin{array}{c} \text{program} \\ 0 \\ \text{part numb} \end{array} \right]$

- program** – PRINT, PRIORITY STATE AND TIME VALUES FOR PROGRAM
- 0** – PRINT NAME AND PARTITION NUMBER OF CURRENTLY EXECUTING PROGRAM. 0 IF NONE.
- part numb** – PRINT NAME OF PROGRAM IN SPECIFIED PARTITION NUMBER. 0 IF NONE.

RTEII/III-48

*ST,0
MEM

*ST,PROGA
1000 1 0 0 0 0 0

*ST,1
PROGA

*ST,PROGB
100 1 0 0 0 0 0

*ST,2
PROGB

*ST,PROGC
10 0 2 100 8 20 50 0 T
PR / S / R / M / TIME / T /

PR - PROGRAM PRIORITY
1 THRU 32767

S - PROGRAM STATE
0 - DORMANT
1 - SCHEDULED
2 - I/O SUSPENDED
3 - GENERAL WAIT
4 - UNAVAILABLE MEMORY
SUSPEND
5 - DISC ALLOCATION SUSPEND
6 - OPERATOR SUSPEND

R - EXECUTION INTERVAL
RESOLUTION CODE
1 - MILLISECONDS x 10
2 - SECONDS
3 - MINUTES
4 - HOURS

M - EXECUTION INTERVAL
MULTIPLIER, 0 THRU 4095

TIME - NEXT TIME TO EXECUTE.
HOURS, MINUTES, SECONDS,
MILLISECONDS x 10

T - TIME LIST INDICATOR

RTEII/III-49

Off

TERMINATES A PROGRAM OR DELETES A TEMPORARY DISC RESIDENT PROGRAM FROM THE SYSTEM.

OF, program $\left[\begin{array}{l} , 0 \\ , 1 \\ , 8 \end{array} \right]$

0 = ORDERLY PROGRAM TERMINATION

REMOVE PROGRAM FROM TIME LIST AND IF:

1. DORMANT, SCHEDULED, OR EXECUTING MAKE PROGRAM DORMANT.
2. I/O, DISC OR MEMORY SUSPENDED, MAKE PROGRAM DORMANT AT END OF SUSPENSION.
DISC TRACKS ARE NOT RELEASED.

1 = ABORT PROGRAM

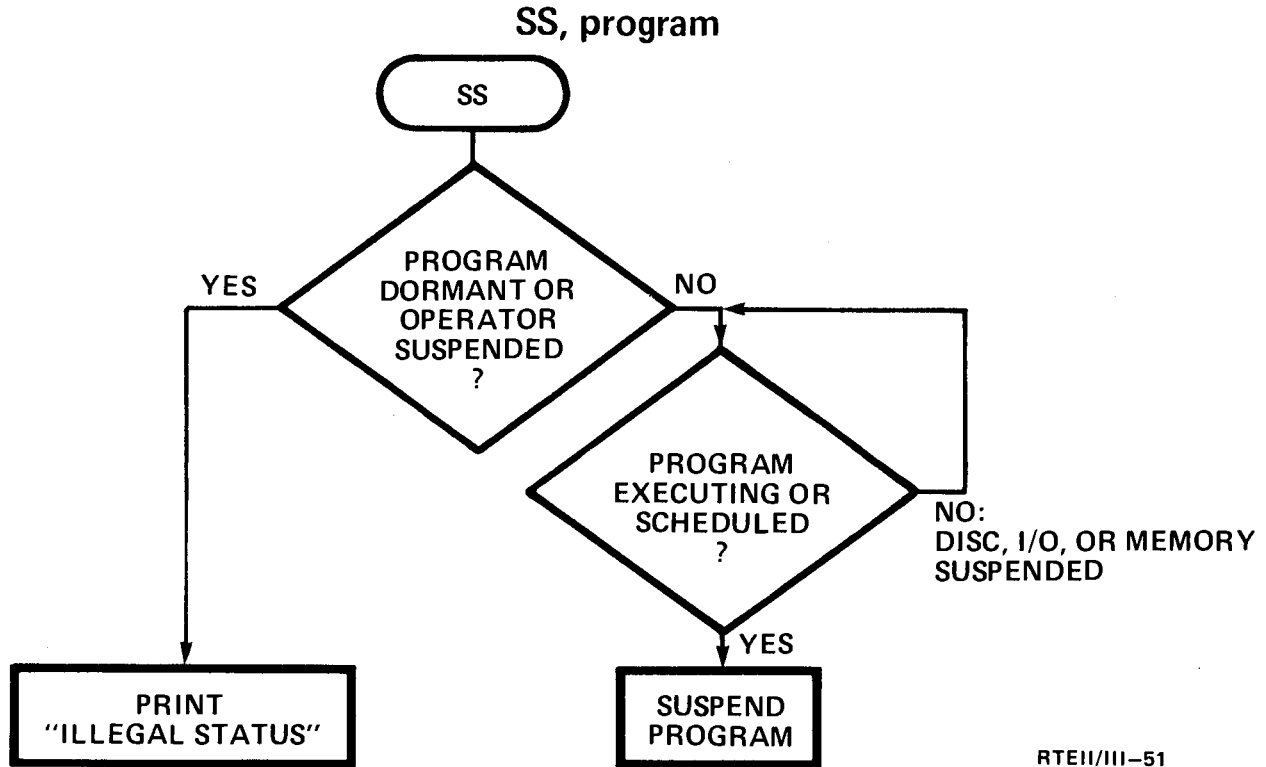
MAKE PROGRAM DORMANT, RELEASE DISC TRACKS AND TERMINATE I/O.

8 = DELETE TEMPORARY DISC RESIDENT PROGRAM FROM SYSTEM.

RTEII/III-50

SuSpend

SUSPENDS A PROGRAM FROM EXECUTION
(OPERATOR SUSPENSION)



RTEII/III-51

An operator suspended program is re-scheduled using the "GO" command. Parameters are not passed to the program.

GO

RESCHEDULES A PROGRAM THAT HAS BEEN SUSPENDED BY AN OPERATOR SS COMMAND OR A SUSPEND EXEC CALL. UP TO FIVE PARAMETERS MAY BE PASSED TO THE PROGRAM.

GO, program[, P1[, ...[, P5]]]]]

P1 - P5 MAY BE:

DECIMAL

$-32768 \leq P_n \leq 32767$

OCTAL

$0 \leq P_n \leq 177777B$

ASCII

$P_n = \text{CHARACTER 1 CHARACTER 2}$

RTEII/III-52

Parameters are not passed to a program that was suspended by an SS operator command.

EXAMPLE

```
FTN,L
PROGRAM PROGA
DIMENSION IPRAM(5)
PAUSE 7
CALL RMPAR(IPRAM)
WRITE(1,100)(IPRAM(I),I=1,5)
100  FORMAT(5I6)
      END
      ENDS

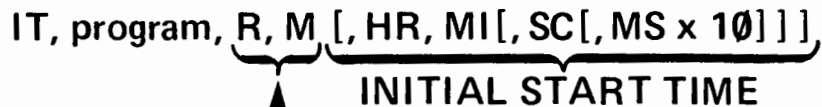
*RU,PROGA
PROGA : PAUSE 0007
*GO,PROGA,-32768,32767,177777B,AS,C
-32768 32767 -1 16723 17184
```

RTEII/III-53

Interval Time

TO SET TIME VALUES FOR A PROGRAM SO THAT THE PROGRAM EXECUTES AUTOMATICALLY AT SELECTED TIMES WHEN TURNED ON WITH THE "ON" COMMAND.

IT, program, R, M, [, HR, MI [, SC [, MS x 10]]]



EXECUTION INTERVAL

R – RESOLUTION CODE

1 - MILLISECONDS x 10

2 - SECONDS

3 - MINUTES

4 - HOURS

M – MULTIPLIER

0 THRU 4095

RTEII/III-54

ON

SCHEDULES A PROGRAM FOR EXECUTION. UP TO FIVE PARAMETERS MAY BE PASSED TO THE PROGRAM.

ON, program [, NOW] [, P1[, ...[, P5]]]]

NOW = (APPLIES TO TIME SCHEDULED PROGRAMS) IF PRESENT, IGNORE TIME SCHEDULING AND PUT PROGRAM IN TIME AND SCHEDULED LISTS. IF ABSENT, PUT PROGRAM IN TIME LIST ONLY.

P1 - P5 MAY BE:

DECIMAL

$-32768 \leq P_n \leq 32767$

OCTAL

$0 \leq P_n \leq 177777B$

ASCII

$P_n = \text{CHARACTER 1 CHARACTER 2}$

RTEII/III-55

If P1 is ASCII "NO" then it must be repeated (the system thinks it is "NOW" of the ON request).

Example: ON, PROG, NO, NO

EXAMPLE

```
*TI
1974  1  15  56  39

*ST,PROGA
   99  0  1   0  17  0  0  0

*ON,PROGA

*ST,PROGA
   99  0  1   0  17  0  0  0 T

*ON,PROGA,NOW

*ST,PROGA
   99  1  1   0  15  57  8  82 T
```

RTEII/III-56

PRiority

CHANGES THE PRIORITY OF A PROGRAM

PR, program, priority



1 THRU 32767

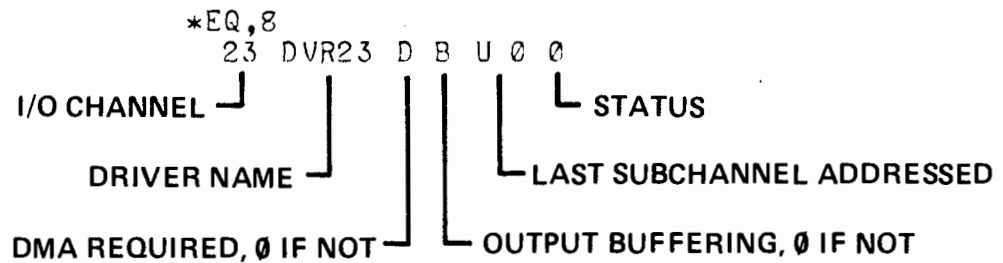
RTEII/III-57

The priority of program resets (to that set by RTGEN or LOADR whenever the RTE-II system restarts from disc).

EQipment table (STATUS)

PRINTS THE DESCRIPTION AND STATUS OF AN I/O DEVICE AS RECORDED IN THE EQT ENTRY.

EQ, eqt



STATUS

- 0 — AVAILABLE
- 1 — DOWN
- 2 — BUSY
- 3 — WAITING FOR DMA ASSIGNMENT

RTEII/III-58

EQquipment table

(BUFFERING)

CHANGES THE AUTOMATIC OUTPUT BUFFERING DESIGNATION
FOR A PARTICULAR I/O DEVICE.

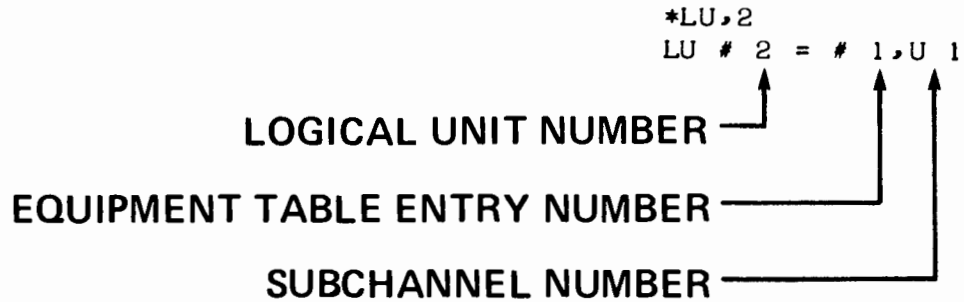
EQ, eqt [UNbuffer
, BUffer]

RTEII/III-59

Logical Unit (STATUS)

PRINTS THE STATUS OF A LOGICAL UNIT ASSIGNMENT

LU, logical unit no.



RTEII/III-60

Logical Unit (REASSIGNMENT)

CHANGES A LOGICAL UNIT NUMBER ASSIGNMENT

LU, lu $\left[\begin{array}{l} ,eqt \\ ,\emptyset \end{array} \right]$ [subchannel number]

*LU, 8
LU # 8 = # 3, U 1

*LU, 8, 4, 2

*LU, 8
LU # 8 = # 4, U 2

RTEII/III-61

Down

DECLARES AN I/O DEVICE DOWN (I.E., UNAVAILABLE FOR USE BY THE RTE SYSTEM).

DN, eqt

UP

DECLARES AN I/O DEVICE UP (I.E., AVAILABLE FOR USE BY THE SYSTEM).

UP, eqt

RTEII/III-62

Time Out



PRINTS OR CHANGES THE TIME-OUT OF AN I/O DEVICE.

TO, eqt[, P1]

P1 = MILLISECONDS x 10 AND $0 \leq P1 \leq 32767$
IF P1 = 0 THEN DEVICE WILL NOT TIME OUT

*T0,2
T0# 2= 9999 } TIME-OUT OF EQT2
≈ 100 SECONDS

*T0,2,5000
*T0,2
T0# 2= 5000 } TIME-OUT OF EQT2
CHANGED TO 50 SECONDS

RTEII/III-63

Time

PRINTS THE CURRENT YEAR, DAY AND TIME AS RECORDED
IN THE REAL-TIME CLOCK.

*TI
1973 181 23 51 59

RTEII/III-64

BReak

SETS AN ATTENTION FLAG IN A PROGRAM'S ID SEGMENT.

BR, program name

RTEII/III-65

EXAMPLE

```
FIN,L
PROGRAM BREAK
DIMENSION IPRAM(5)
      .
      .
      .
C
C TEST BREAK BIT
C
10 IF (IFBRK(0M))20,30
C
C BREAK BIT SET (OPERATOR ENTERED "BR,BREAK")
C
C PAUSE
20
C
C PROGRAM RE-SCHEDULED BY "GO" COMMAND
C
CALL RMPAR(IPRAM)
      .
      .
      .
30 GO TO 10
      .
      .
      .
END
```

NOTE: ROUTINE "IFBRK" IS IN THE RTE-II SYSTEM LIBRARY.

RTEII/III-66

ABort

TERMINATES THE CURRENT PROGRAM (IF ANY) THAT WAS SCHEDULED VIA THE BATCH MONITOR.

AB $\left[\begin{array}{c} \emptyset \\ , 1 \end{array} \right]$

\emptyset = (SAME AS FOR "OF" COMMAND)

1 = (SAME AS FOR "OF" COMMAND)

EXAMPLES

```
*ON,FMGR
:RU,PROG
```

```
*AB
ABEND  PROG  ABORTED
:EX
SEND  FMGR
```

```
*ON,FMGR
:RU,PROG
```

```
*AB,1
PROG  ABORTED
ABEND  PROG  ABORTED
:EX
SEND  FMGR
```

RTEII/III-67

SWap (RTE-II only)

(CONTROL)

ENABLES OR DISABLES PROGRAM SWAPPING

SW

,0
,1
,2
,3

FOREGROUND
DISC RESIDENT
PROGRAM AREA

BACKGROUND
DISC RESIDENT
PROGRAM AREA

	FOREGROUND DISC RESIDENT PROGRAM AREA	BACKGROUND DISC RESIDENT PROGRAM AREA
0	DISABLE	DISABLE
1	ENABLE	DISABLE
2	DISABLE	ENABLE
3	ENABLE	ENABLE

RTEII/III-68

SWap (STATUS)

DISPLAYS THE SWAPPING WORD (BASE PAGE LOCATION 1736B)

*SW
31017

3 1 0 1 7
0|0 1 1|0 0 1|0 0 0|0 0 1|1 1 1

SWAP DELAY = 500 MS
(VALUE IN MS x 10)

BACKGROUND CORE LOCK ENABLED

BACKGROUND CORE LOCK ENABLED

BACKGROUND SWAPPING ALLOWED

BACKGROUND SWAPPING ALLOWED

RTEII/III-69

Buffer Limit

EXAMINES OR MODIFIES BUFFERING LIMITS

BL [,lower limit, upper limit]

EXAMINE

*BL
100 400

MODIFY

*BL, 50, 450

*BL
50 450

RTEII/III-70

OPERATOR REQUEST ERRORS

MESSAGE	MEANING
OP CODE ERROR	ILLEGAL OPERATOR REQUEST WORD.
NO SUCH PROG	THE name GIVEN IS NOT A MAIN PROGRAM IN THE SYSTEM.
ILLEGAL STATUS	A PROGRAM IS NOT IN THE APPROPRIATE STATE.
INPUT ERROR	A PARAMETER IS ILLEGAL.

RTEII/III-71

PARTITION STATUS

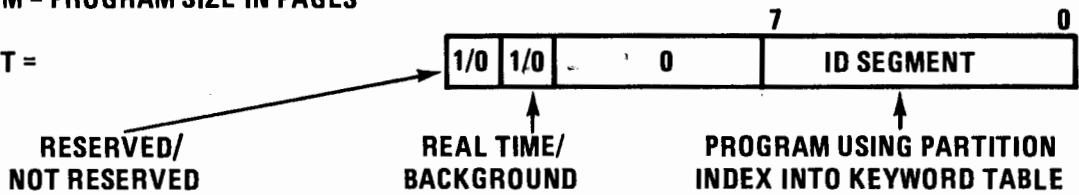
CALL EXEC (25,IPART,IPAGE,IPNUM,ISTAT)

IPART = PARTITION NUMBER

IPAGE = STARTING PAGE NUMBER

IPNUM = PROGRAM SIZE IN PAGES

ISTAT =



RTEII/III-132

LABORATORY EXERCISE GUIDE #1

OBJECTIVE

To provide the student an exercise in using the RTE-II operator commands to determine the I/O configuration of the class system.

PROCEDURE

Bootstrap the system into memory and start it executing. Using operator commands, fill in the following matrix.

DEVICE	LOGICAL UNIT NO.	EQT NO.	SELECT CODE	DVR NO.	DMA ?	BUFFERED ?	TIME-OUT VALUE
	1,4,5	1	12	05	0	1	
	21,22,23	2	11	32	1	0	
	17	3	25	43	1	0	
	18	4	26	43	1	0	
	19	5	27	43	1	0	
	6	6	13	12	1	0	
	7	7	20	33	1	0	
	8	8	15	23	1	0	
	9	9	22	00	0	B	
	10	10	23	00	0	B	
	13,11,12	11	17	05	0	B	
	20	12	30	43	0	1	
	21	13	31	43	1	0	
	16,14,15	14	24	05	1	B	
	22	15	32	43	0	0	
	25	16	4	43	1	0	
		17	14	47	1	0	
	26	18	14	47	1	0	
	27	19			1	0	

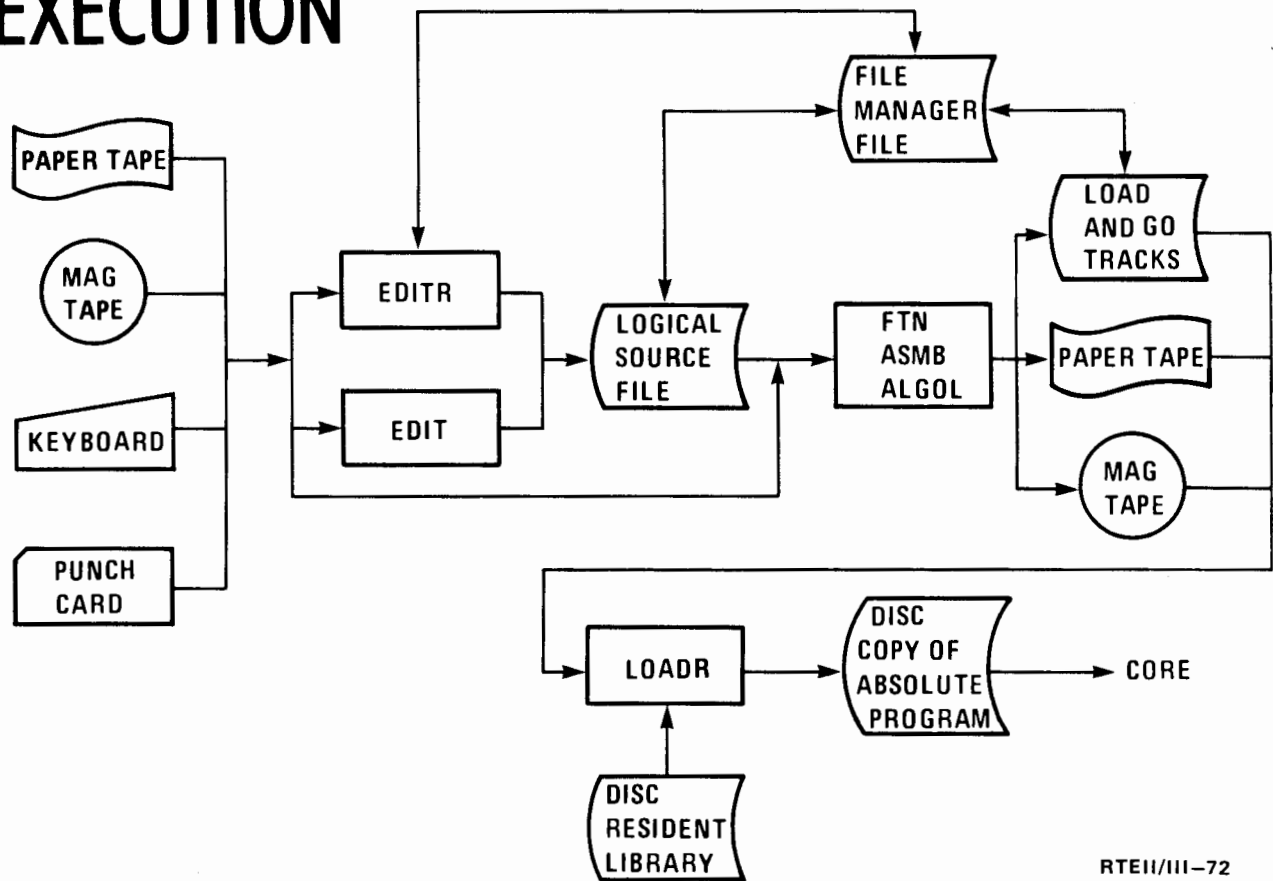
DEVICE DRIVERS

DVR00 = TTY, DVR01 = TAPE READER, DVR02 = TAPE PUNCH

DVR11 = CARD READER, DVR12 = LINE PRINTER, DVR23 = MAG TAPE

DVR31 = DISC, DVR61 = 6940, DVR62 = 2313, DVR43 = SPOOL OR POWER FAIL

PROGRAM PREPARATION, LOADING AND EXECUTION



RTEII/III-72

RTE II PROGRAM CONTROL STATEMENTS

<u>FORTTRAN</u>	FTN4,A,B,L,C,M, numb
<u>ALGOL</u>	HPAL,A,B,L,P
<u>ASSEMBLY</u>	A = ASSEMBLY LISTING B = BINARY OBJECT TAPE C = CROSS REFERENCE L = SOURCE LISTING M = ASSEMBLY LISTING BY STATEMENT P = A PROCEDURE ONLY IS TO BE COMPILED n = 1 - 9 A SUFFIX FOR A USER WRITTEN ERROR ROUTINE, ERR n.

RTEII/III-73

In case of errors during execution of library routines ALOG, SIN, COS, SQRT, .RTOR, RT10, EXP, .ITOI, or TAN. User written routine ERR numb is called. If numb is not supplied in the control statement, a library routine ERRØ will be used.

ASSEMBLY

ASMB,A,R,B,L,T,N/Z,C,F,X

- A = ABSOLUTE ASSEMBLY
- R = RELOCATABLE ASSEMBLY
- B = BINARY OUTPUT
- L = LIST OUTPUT
- T = SYMBOL TABLE PRINT
- N/Z = SELECTIVE ASSEMBLY
- C = CROSS REFERENCE TABLE PRINT
- F = FLOATING POINT INSTRUCTIONS
- X = NO EAU HARDWARE

RTEII/III-74

RTE II PROGRAM name STATEMENTS

FORTRAN

PROGRAM name [type, pri, res, mult, hr, min, sec, msec]

ASSEMBLY

NAM name [,type, pri, res, mult, hr, min, sec, msec]

ALGOL

"name" [numb, type, pri, res, mult, hr, min, sec, msec]

RTEII/III-75

RTE II PROGRAM name STATEMENT PARAMETERS

numb	=	1-9 A SUFFIX FOR A USER WRITTEN ERROR ROUTINE, ERR numb	
type	=	PROGRAM TYPE NUMBER	
pri	=	PROGRAM PRIORITY	
res	=	RESOLUTION CODE	} EXECUTION, INTERVAL/ START TIME
mult	=	MULTIPLE	
hr	=	HOURS	
min	=	MINUTES	
sec	=	SECONDS	
msec	=	MILLISECONDS X10	

RTEII/III-76

In case of errors during execution of library routines ALOG, SIN, COS, SQRT, .RTOR., RT10, EXP, .ITOI, or TAN user written routine ERR numb is called. If numb is not supplied a library routine ERR0 will be used.

EXAMPLE

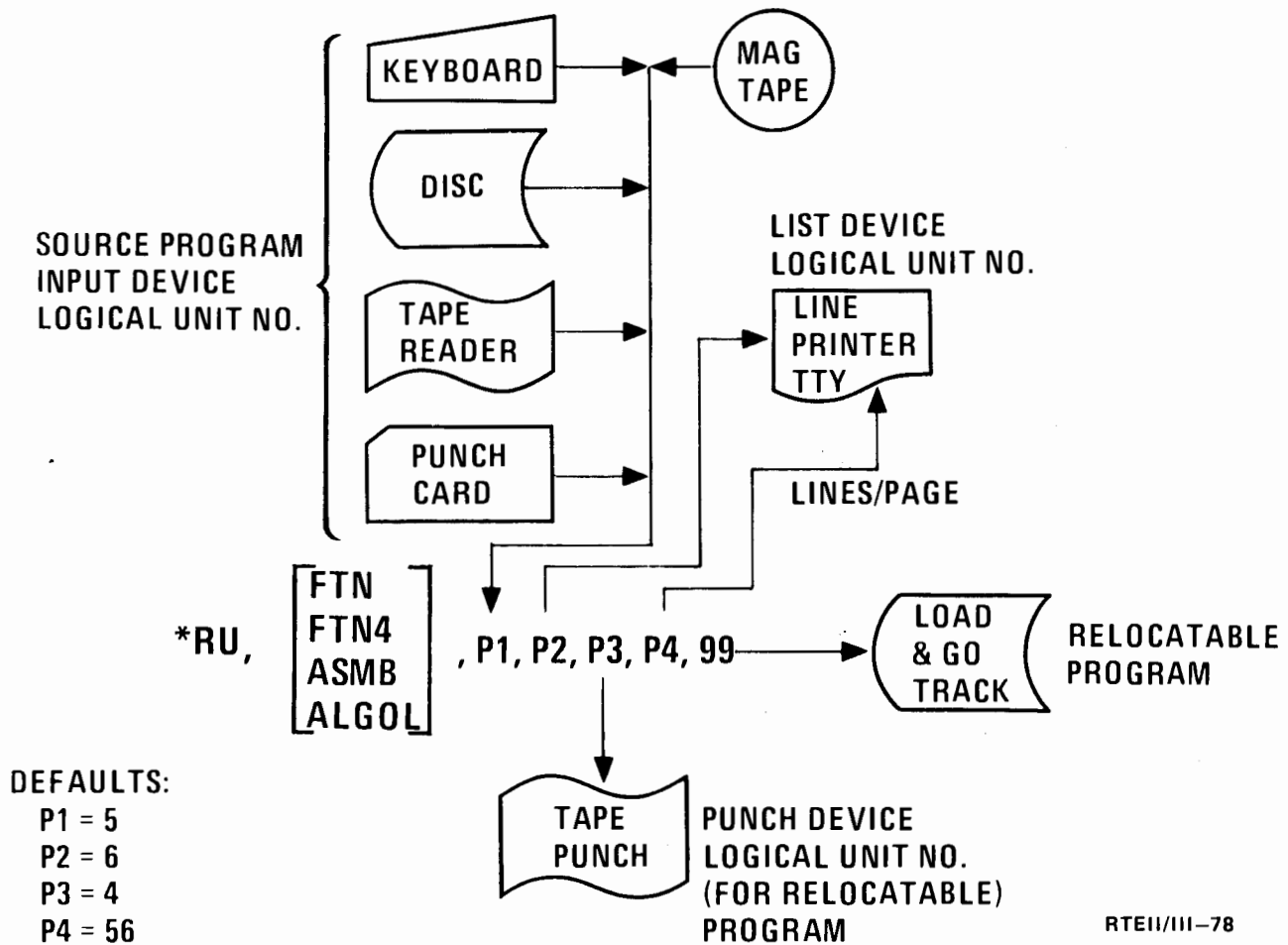
A FOREGROUND DISC RESIDENT PROGRAM WITH PRIORITY OF 10, AND WHEN PLACED IN THE TIME LIST WILL BE SCHEDULED TO EXECUTE AT 8:00 A.M. AND EVERY HOUR THEREAFTER.

HEWLETT-PACKARD

PROGRAMMER										DATE									
										STATEMENT									
1	Label	5	Operation	10	Control	15	20	25	30	35	40	45	50	55	60	65	70	75	80
	FTNH, L, A, B, C		PROGRAM	PROGA	2,	10,	4,	1,	8,	0,	0,	0,							
			.																
			.																
			.																
			END																

RTEII/III-77

COMPILER/ASSEMBLER OPERATION



P1 - P4 assume default values if not supplied.

*ON,FTN4,*ON,FTN,5,6,4,56 and *ON,FTN4,,,4 are all equivalent.

Load and Go parameter, 99, terminates the parameter list.

*ON,FTN4,5,6,4,99 and *ON,FTN4,99 are equivalent.

Tape will be punched if a program's control statement so specifies (i. e. FTN4,B).

Handwritten notes:
 UCINSA 196000001
 FTN4, B
 FTN4, B (not better)

Load AND Go

**ALLOCATES OR RELEASES A CONTIGUOUS
GROUP OF TRACKS FOR TEMPORARY STOR-
AGE OF RELOCATABLE CODE OUTPUT FROM
ASSEMBLER, COMPLIER OR FILE MANAGER**

***LG, number of tracks**

**number of tracks = 0 RELEASE LOAD
AND GO TRACKS
> 0 NUMBER OF
TRACKS TO
BE ALLOCATED**

RTEII/III-79

PROGRAM FMGR (file manager)

*RU, FMGR
: ← unprompt from FMGR

- FMGR COMMANDS -

STORE INTO A FILE

:ST, <from lu>, <to file name>

MOVE SOURCE INTO LOGICAL SOURCE TRACK

:MS, <from file name>, <lu#>

ALLOCATE LOAD AND GO TRACKS

:LG, <# of tracks>

RUN A PROGRAM

:RU, <program name>

LIST A FILE

:LI, <file name>

PURGE A FILE

:PU, <file name>

EXIT FMGR

:EX

list & return

RTEII/III-80

DL FILE DIRECTORY LIST
LIST LU CHANGE
= LL < LU# >

SAVE LG IN A FILE

: SA, LG < FILE NAME >

MOVE FILE INTO LG

: MOVE FILE

: MR < FILE NAME >

EXAMPLE

CREATE A SOURCE PROGRAM FILE

```
*RU, FMGR  
:ST,1,MYFILE  
FTN,L
```

PROGRAM NAME

•
•
•

END

END \$

D©

COMPILE THE SOURCE PROGRAM

```
:LG,1  
:MS,MYFILE  
:RU, FTN4, 2, 99
```

FILE
MYFILE

LOGICAL
SOURCE
FILE

LOAD
AND GO
TRACK

RELOCATABLE
PROGRAM

RTEII/III-81

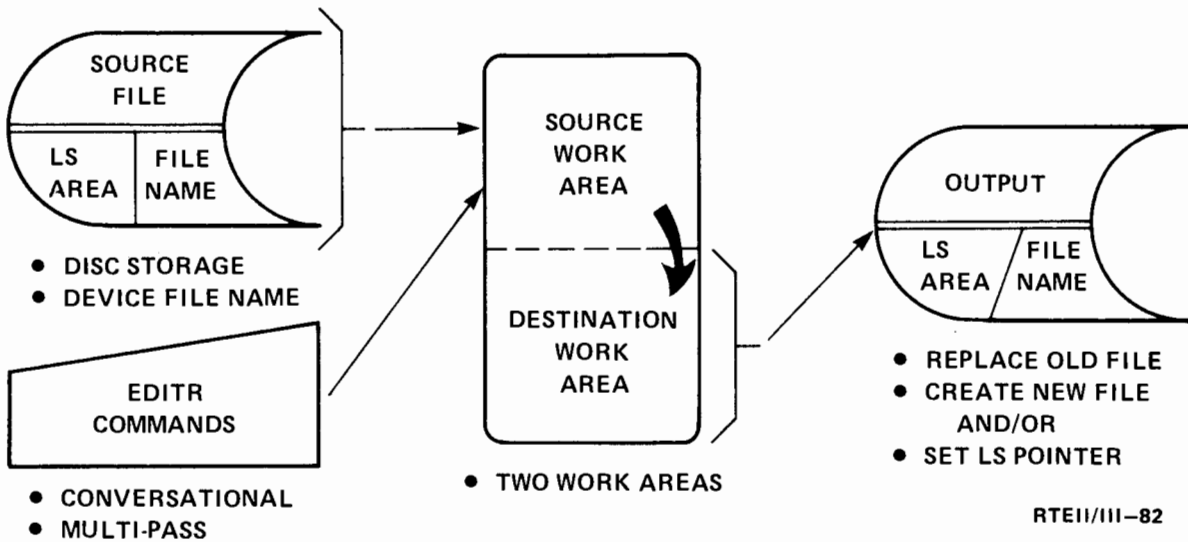
RTE INTERACTIVE EDITOR

PROVIDES A CONVENIENT AND FLEXIBLE MEANS FOR

- MODIFICATION OF NEW OR EXISTING FILES
- CREATION OF NEW PROGRAMS OR DATA FILES IN ASCII CODE
- APPENDING SEVERAL FILES TOGETHER

IN EITHER ON-LINE OR BATCH MODE.

ENVIRONMENT



RTEII/III-82

EDITR FUNCTIONS

CATEGORY	FUNCTION
LINE EDITS	EDIT ONE LINE (RECORD) AT A TIME
CHARACTER EDITS	EDIT INDIVIDUAL CHARACTERS WITHIN A LINE
PATTERN EDITS	EDIT PATTERN SEQUENCES AND BLOCKS OF TEXT

WITHIN EACH EDIT CATEGORY, THERE EXISTS A WIDE RANGE OF EDIT
COMMANDS GIVING THE USER A POWERFUL MEANS TO CONVERSATIONALLY
EDIT TEXT.

RTEII/III-83

EDITR EXECUTION



*ON,EDITR[P1[,P2]]

LU FOR COMMAND
INPUTS (DEFAULT=1)

MAXIMUM RECORD LENGTH
(DEFAULT=150)

EXAMPLE 1

INPUT COMMANDS FROM LU=1 AND
SET MAXIMUM RECORD LENGTH TO
DEFAULT OF 150 CHARACTERS:

*ON,EDITR

EXAMPLE 2

SET INPUT COMMANDS TO LU7 AND
MAXIMUM RECORD LENGTH TO
72 (LARGER RECORDS TRUNCATED):

*ON,EDITR,7,72

SEE NEXT SLIDE FOR EDITOR'S NEXT RESPONSE

RTEII/III-84

SOURCE INPUT ?

CONSOLE DIALOGUE:

EDITOR	SOURCE FILE?	EDITOR REQUESTS WHERE FILE TO BE EDITED FROM
USER	/MYFILE:SC:100	SOURCE INPUT FROM FILE NAMED "MYFILE" WITH SECURITY CODE "SC" ON CARTRIDGE "100". THE FILE NAME CAN BE A TYPE 1 DEVICE FILE NAME.
	-OR-	
USER	/^	INPUT SOURCE FROM LOGICAL SOURCE AREA. "^" IS THE <u>SPACE</u> CHARACTER
EDITOR	FTN4,L,B /	EDITOR PRINTS FIRST LINE OF TEXT AND THEN PROMPTS WITH THE "/" CHARACTER. EDITING CAN NOW BEGIN.

LEGEND:

 SHADED AREA REPRESENTS OPERATOR'S INPUT.

NOTE: IF AN ILLEGAL FILE NAME IS GIVEN OR THE FILE IS INACCESSIBLE, THE EDITOR WILL PRINT THE FMGR ERROR MESSAGE.
TO ABORT, TYPE A SPACE (AS IF FROM THE LS AREA) AND THEN A.

RTEII/III-85

EDITING EXAMPLE

THE FOLLOWING EDITING SESSION WILL PUT A DATA FILE BACK INTO ITS LOGICAL ORDER.

```
*ON,EDITR
SOURCE FILE?
/DATAZ
FILE MANAGER ERROR -06
SOURCE FILE?
/DATA01::-2
      EMPLOYEE NAME          SODIAL SECURITY
/LI00
      EMPLOYEE NAME          SODIAL SECURITY
      LOU LEN                 545:74:7777
      DEGRAFF SHEILA         334:12:3890
      SINDLER MIKE           098:23:4451
      [.?:XMNC"#!O\\ M<>'::=&$-
*****
      MCGILLICUDY RAY        345:45:0098
      GILFEATHER LOIS       555:06:3472
      WINKER SANDY          466:89:2491
      SMITH JOHN             556:90:9012
      EMERSON SHARON        438:45:0233
      SY JOE                 223:78:42855
      PAVONNE SEGE          R45:65:1293   GA
      HEWLETT BILL          444/92/6745

EOF
/N
  15
/14
      HEWLETT BILL          444/92/6745
```

RTEII/III-86

EDITING EXAMPLE (CONT.)

```

/1 EMPLOYEE NAME          SOCIAL SECURITY
/P////M//////C
/ PAVONNE SEGE          445-66-1293
/R SINDLER MIKE          098 23-4451
/P SYNDLER MICK          098-22-4456
// SYNDLER MICK          098-22-4456
[-2 [. ? ^ XMNC " # ! @ \ \ M < > ' : : = & $ % -
/P 1 MCGILLICUDY RAY    345:45:0098
/+4 MCGILLICUDY RAY    345:45:0098
/+1 EMERSON SHARON      438:45:0233
/P 2 SMITH JOHN        566:90:9012
SMITH JOHN             566:90:9012
/+2 SY JOE             223:78:42855
/G55/5 SY JOE           223:78:4285
// PAVONNE SEGE       R45:66:1293 GA
/H 36 PAVONNE SEGE     445-66-1293
/P 445-66-1293
/ISXWARTZ ZEEK
Q
??
/-

```

delete rest of line

EDITING EXAMPLE (CONT.)

/I SXWARTZ ZEEK 349-05-2379

/+1 SXWARTZ ZEEK 349-05-2379

/P////C//////////5
SCWARTZ ZEEK 359-05-2379

/2 LOU LEN 545:74:7777

/W20,20

/U1/A

/9999

EOF

/2 LOU LEN A 545:74:7777

/W1,80

/X:/-

/FEND

LOU LEN A 545-74-7777

DEGRAFF SHEILA A 334-12-3890

MCGILICUDY RAY A 345-45-0098

GILFEATHER LOIS A 555-06-3472

WINKER SANDY A 466-89-2491

SMITH JOHN A 566-90-9012

EMERSON SHARON A 438-45-0233

SY JOE A 223-78-4285

EOF

/1

EMPLOYEE NAME SOCIAL SECURITY

/L100,6

I/O ERR NR EOT # 6

*UP,6

EOF

RTEII/III-88

ABORT EDITR

/A → ABORTS THE EDITOR PROCESS LEAVING SOURCE FILE INTACT AND LS POINTER.

EDITR TERMINATIONS

FILE ONLY

/EC namr:sc:cr → CREATE A NEW FILE

/ER namr:sc:cr → REPLACE OLD FILE

LS AREA ONLY

/EL → MOVE TO LS AREA AND SET POINTER.

LS AREA AND FILE

/EL (namr:sc:cr → CREATE A NEW FILE, MOVE TO LS AREA, AND SET POINTER.

/ELR namr:sc:cr → REPLACE OLD FILE, MOVE TO LS AREA, AND SET POINTER.

RTEII/III-90

ELC

CREATING A SOURCE FILE USING THE TAB FEATURE

```

*LS,2,0

*ON,EDITOR
SOURCE FILE?
/^
EOF
/T;7,19,52
/ ASMB,R,L ** MEMORY RETRIEVE ROUTINE **
/ ;NAM IGET,7;;<A>
/P
        NAM IGET,7                                <A>
/^IGET;NOP;<<ENTRY POINT>>;<A>
/P;;;<A>
        IGET  NOP                                <<ENTRY POINT>>    <A>
/^;DLD IGET,I;GET RETURN & ARGUMENT ADDRESSES.; A
/^;SWP;INTERCHANGE FOR PROPER RETURN;<A>
/^;LDA 0,I;GET ACTUAL MEMORY CONTENTS.;<A>
/^;JMP 1,I;RETURN TO CALLER;<A>
/^;END;;;<A>
/1
        ASMB,R,L ** MEMORY RETRIEVE ROUTINE **
/Li00
        ASMB,R,L ** MEMORY RETRIEVE ROUTINE **
        NAM IGET,7
        IGET  NOP                                <<ENTRY POINT>>    <A>
        DLD IGET,I  GET RETURN & ARGUMENT ADDRESSES. <A>
        SWP          INTERCHANGE FOR PROPER RETURN   <A>
        LDA 0,I      GET ACTUAL MEMORY CONTENTS.    <A>
        JMP 1,I      RETURN TO CALLER                <A>
        END
EOF
/ELCGET:HP:-2
LS FILE 2 27
END OF EDIT

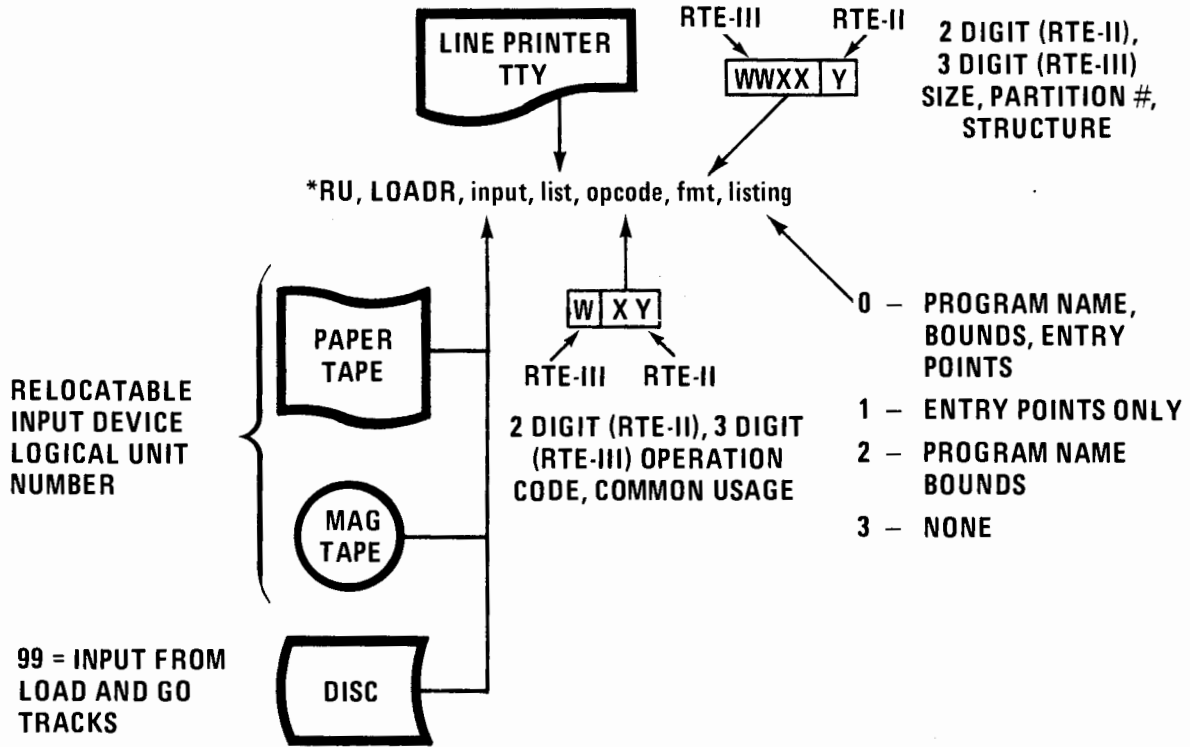
```

THE COMMAND 'LS,2,0' SETS THE LS POINTER TO ZERO. THIS INFORMS THE EDITOR TO ASSUME AN EMPTY SOURCE AREA.

RTEII/III-91



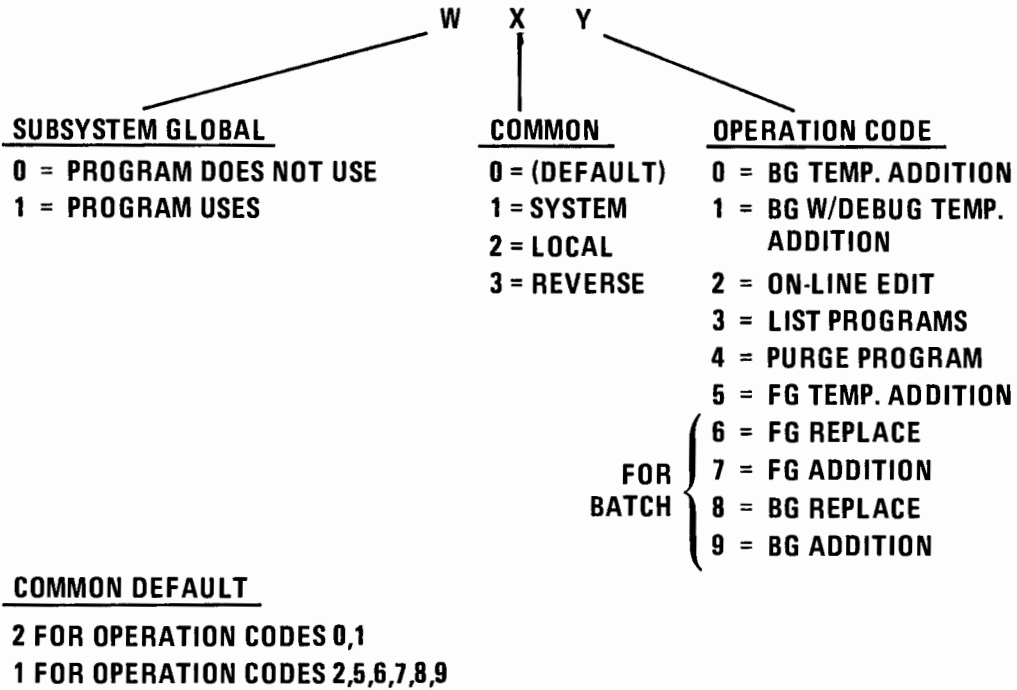
LOADER OPERATION



RTEII/III-92

Default for *ON,LOADR is *ON,LOADR,5,6,0,0,0

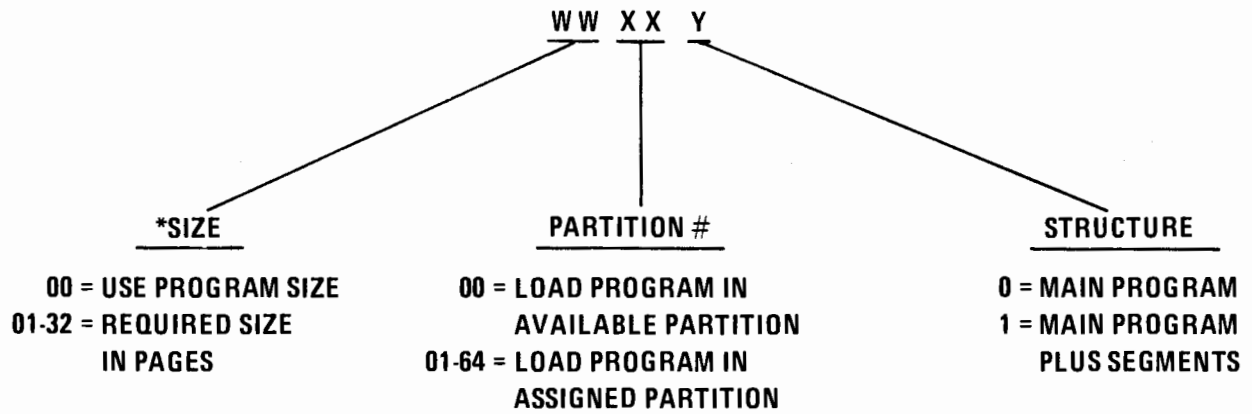
LOADER OPERATION CODES (opcode)



RTEII/III-93

on loader

(FMT)



* HOW MANY PAGES TO USE IN PARTITION

RTEII/III-94

LOADER SUSPENDS IF:

INPUT WAS NOT FROM LOAD-AND-GO

-OR-

UNDEFINED EXTERNALS EXIST

*GO,LOADR, input, map, Libry

input

- 0 = LOAD FROM LU5
- 99 = LOAD FROM LOAD-AND-GO FOR THE FIRST TIME
- 2 = LOAD FROM LOAD-AND-GO AGAIN
- n = $5 \leq n \leq 63$, A LOGICAL UNIT NO.
- 1 = SCAN DISC RESIDENT LIBRARY
- 3 = SCAN DISC RESIDENT LIBRARY, LAST SEGMENT LOAD
- 4 = CONTINUE, IGNORE UNDEFINED EXTERNALS
- 98 = LIST UNDEFINED EXTERNALS

map

- 0 = LIST ENTRY POINTS
- 1 = DO NOT LIST ENTRY POINTS

Libry

- 0 = LOAD ALL input
- 1 = SCAN input FOR UNDEFINED EXTERNALS

RTEII/III-95

*99 gets info from LU loader
kill, loader, 99, LU 1, 2
or LU, FTR4, 2, 99*

LOADR SUSPENDS IF opcode = 2 (on-line-edit)

GO,LOADR, operation, prog type [,priority]

<u>operation</u>	<u>priority</u>
1 = ADD A PROGRAM	Ø = USE VALUE IN NAM RECORD OR IF NONE SET TO 32767
2 = REPLACE A PROGRAM	1 to 32767 SETS PRIORITY VALUE
<u>prog type</u>	
2 = FORGROUND DISC RESIDENT	
3 = BACKGROUND DISC RESIDENT	

RTEII/III-96

EXAMPLE

COMPILE THE SOURCE PROGRAM

```
*RU, FMGR  
:LG,1  
:MS, MYFILE  
:RU, FTN4,2,99
```

LOGICAL
SOURCE
FILE

LOAD THE PROGRAM

```
:RU,LOADR,99
```

LOAD
AND GO
TRACKS

RUN THE PROGRAM

```
:RU, MINE  
:EX
```

RTEII/III-97

EXAMPLE

1. ADD A FOREGROUND PROGRAM WITH INPUT FROM LOAD-AND-GO TRACKS

```
*ON, LOADR, 99, , 2
```

```
/LOADR: "GO" WITH EDIT PARAMETERS
```

```
*GO, LOADR, 1, 2, 10
```

```
/LOADR: PROGA READY
```

```
/LOADR: $END
```

2. REPLACE A BACKGROUND PROGRAM. INPUT FROM LOAD AND GO AND REVERSE COMMON

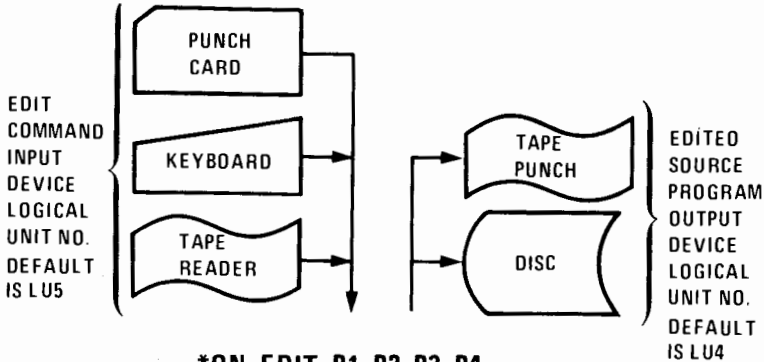
```
*ON, LOADR, 99, , 38
```

```
/LOADR: PROGB READY
```

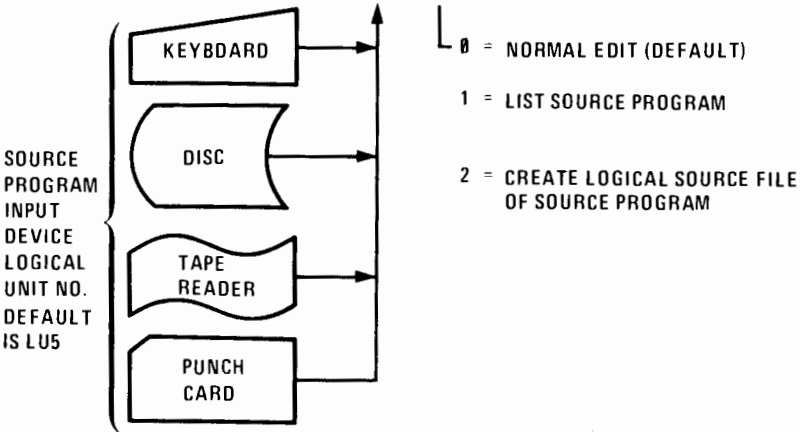
```
/LOADR: $END
```

RTEII/III-98

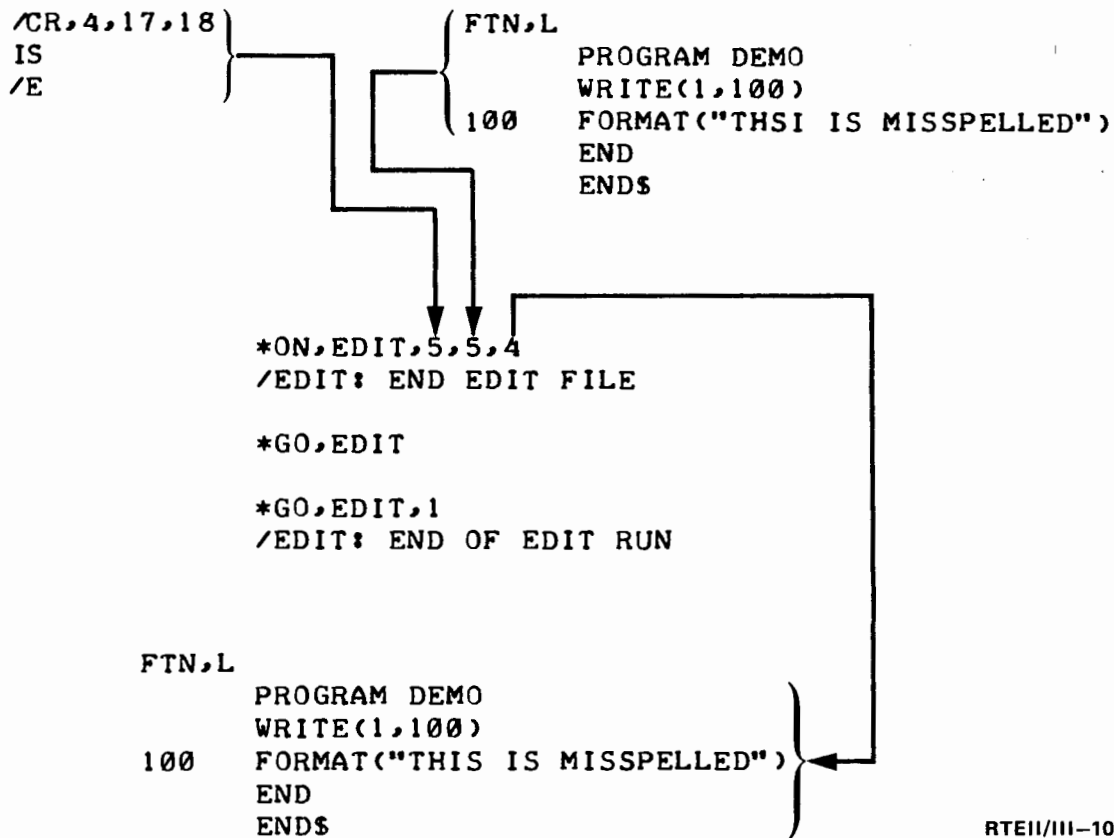
EDITOR OPERATION



*ON, EDIT, P1, P2, P3, P4



PAPER TAPE -PAPER TAPE EDITING



RTEII/III-100

Logical Source

SETS A POINTER TO A SYSTEM MANAGED
SOURCE FILE THAT IS TO BE ACCESSED
BY EDIT, EDITR, FTN, FTN4, ALGOL OR
ASMB



LS, disc lu, track number

disc lu 2 - SYSTEM DISC OR
 3 - AUXILIARY DISC

track number 0 - 202

RTEII/III-101

PAPER TAPE → DISC → DISC EDITING

CREATE A
LOGICAL
SOURCE
FILE

{ *ON,EDIT,,5,,2
*GO,EDIT,1
/EDIT: TRACKS IN NEW FILE :
/EDIT: 02, 0025
/EDIT: END OF EDIT RUN

CREATE A
BACK-UP
COPY OF
THE
SOURCE

{ *LS,2,25
*ON,EDIT,,2,,2
/EDIT: TRACKS IN NEW FILE :
/EDIT: 02, 0026
/EDIT: END OF EDIT RUN

EDIT
SOURCE

{ *LS,2,26
*ON,EDIT,1,2,2
/EDIT: ENTER EDIT FILE:
/CR,4,17,18
IS
/E
/EDIT: TRACKS IN NEW FILE :
/EDIT: 02, 0027
/EDIT: END OF EDIT RUN

RTEII/III-102

LISTING A LOGICAL SOURCE FILE

*LS,2,27

*ON,EDIT,,2,1,1

PAGE 0001

```
0001  FTN,L
0002          PROGRAM DEMO
0003          WRITE(1,100)
0004  100    FORMAT("THIS IS MISSPELLED")
0005          END
0006          ENDS
```

/EDIT: END OF EDIT RUN

RTEII/III-103

OTHER EDITOR OPERATIONS

EDIT COMMANDS
FROM KEYBOARD,
SOURCE TO BE
EDITED FROM
PAPER TAPE.

```
*ON,EDIT,1,5,4
/EDIT: ENTER EDIT FILE:
/CR,4,17,18
IS
/E
/EDIT: END EDIT FILE

*GO,EDIT

*GO,EDIT,1
/EDIT: END OF EDIT RUN
```

CREATING A
LOGICAL
SOURCE FILE
FROM
KEYBOARD

```
*ON,EDIT,,257,,2
FTN,L
PROGRAM DEMO
WRITE(1,100)
100 FORMAT("THIS IS MISSPELLED")
END
END$
"CONTROL D"
← TERMINATES INPUT

*GO,EDIT,1
/EDIT: TRACKS IN NEW FILE :
/EDIT: 02,0025
/EDIT: END OF EDIT RUN
```

*LOGICAL UNIT NO. +256

RTEII/III-104

*If 256 is not added to the logical unit number then keyboard input will not be printed.

LABORATORY EXERCISE

OBJECTIVE

TO PROVIDE AN EXERCISE IN USING THE EDITOR COMPILER AND LOADER FOR PROGRAM DEVELOPMENT.

PROBLEM

BELOW ARE A FORTRAN PROGRAM (PRIME) AND A FORTRAN SUBROUTINE (TIME). THE SUBROUTINE IS ERROR FREE BUT THE PROGRAM CONTAINS SEVERAL ERRORS. THE STUDENT IS TO CORRECT THESE ERRORS, COMPILE THE PROGRAM AND SUBROUTINE, LOAD THEM AS A PROGRAM AND THEN EXECUTE IT.

PROCEDURE

USING EDITR, CORRECT THE ERRORS IN PROGRAM PRIME. A TAPE OF PRIME WILL BE SUPPLIED BY YOUR INSTRUCTOR. CREATE A SOURCE FILE OF SUBROUTINE TIME BY TYPING IN THROUGH ONE OF THE SYSTEM TELETYPES. COMPILE PRIME AND TIME, LOAD THEM USING LOADR THEN EXECUTE PRIME. THE RESULT SHOULD AGREE WITH THE THE SAMPLE PRINTOUT BELOW.

FTN,L

```
PROGRAM PRIME
DIMENSION I(5)
INTEGER START(5),STOP(5),TOTAL(5),HOUR,MINUTE,SECOND
EQUIVALENCE(HOUR,STOP(4)),(MINUTE,STOP(3)),(SECOND,STOP(2))
DATA TOTAL/5*0/
1 * WRITE(6,100)
   CALL EXEC(11,START)
100 FORMAT(12X,"PRIMES FROM ONE TO FIVE HUNDRED TEN"////
C9X,"1",19X,"2",19X"3")
   J=4
   M=1
15   C=0.0
      DO 65 L=2,510
40   IF (J-J/L*L)65 60,65
60   C=C+1
65   CONTINUE
70   IF (C-1.0)200,77,200
77   I(M)=J
      M=M+1
      IF (M-6) 200,150
150  M=1
      CALL EXEC(11,STOP)
      CALL TIME(START,STOP,1)
      CALL TIME(TOTAL,STOP,0)
      WRITE (6,400) I
400  FORMAT(5(I10))
200  J=J+1
      IF (J-510) 15,15,500
500  SECOND=SECOND+TOTAL/100.
      * WRITE(6,101) HOUR,MINUTE,SECOND
101  FORMAT(///" ELAPSED TIME:"I3" HOURS" I3" MINUTES "F5.2" SECONDS"
      END
      ENDS
```

Lu.

510 0
CON
ACT
28
100

```

FTN4,L
SUBROUTINE TIME(START,STOP,OPTION)
INTEGER START(5),STOP(5),OPTION,BASE(4)
DATA BASE/100,60,60,24/
C
C THIS ROUTINE ADDS OR SUBTRACTS TWO TIME VALUES IN ARRAYS
C START AND STOP. THE RESULT IS RETURNED IN ARRAY STOP.
C
C START(1)/STOP(1)=MILLISECONDS X10
C START(2)/STOP(2)=SECONDS
C START(3)/STOP(3)=MINUTES
C START(4)/STOP(4)=HOURS
C IF OPTION .EQ. 0, THEN ADD, ELSE SUBTRACT.
C
C IF(OPTION.NE.0) GO TO 200
C
C ADD ROUTINE
C
C DO 100 J=1,4
C STOP(J)=STOP(J)+START(J)
C IF(STOP(J).LT.BASE(J)) GO TO 100
C STOP(J)=STOP(J)-BASE(J)
C STOP(J+1)=STOP(J+1)+1
100 CONTINUE
C GO TO 300
C
C SUBTRACT ROUTINE
C
C DO 300 J=1,4
C STOP(J)=STOP(J)-START(J)
C IF(STOP(J).GE.0) GO TO 300
C STOP(J)=STOP(J)+BASE(J)
C STOP(J+1)=STOP(J+1)-1
300 CONTINUE
END
END$

```



PRIMES FROM ONE TO FIVE HUNDRED TEN

1		2		3
5	7	11	13	17
19	23	29	31	37
41	43	47	53	59
61	67	71	73	79
83	89	97	101	103
107	109	113	127	131
137	139	149	151	157
163	167	173	179	181
191	193	197	199	211
223	227	229	233	239
241	251	257	263	269
271	277	281	283	293
307	311	313	317	331
337	347	349	353	359
367	373	379	383	389
397	401	409	419	421
431	433	439	443	449
457	461	463	467	479
487	491	499	503	509

ELAPSED TIME: 0 HOURS 1 MINUTES 34.00 SECONDS

RTE II EXEC CALLS

EXEC CALLS ARE THE LINE OF COMMUNICATION BETWEEN AN EXECUTING PROGRAM AND RTE-II.

CALL FORMATS

ASSEMBLY

EXT EXEC
:
:
JSB EXEC
DEF * + n + 1
DEF P1
:
:
DEF Pn
return point
:
:

P1
:
:
Pn } ACTUAL
VALUES

FORTTRAN

P1 = a
:
:
Pn = z
CALL EXEC (P1 ... Pn)

- OR -

CALL EXEC (a ... z)

P1 IS AN INTEGER THAT
SPECIFIES THE TYPE
OF REQUEST

P2 - Pn ARE THE CALL
PARAMETERS

RTEII/III-105

(* + n + 1) may not be an indirect address

**SOME EXEC CALLS RETURN INFORMATION IN THE
A AND B REGISTERS.**

ACCESSING THE A AND B REGISTERS IN FORTRAN

FTN, L

```
PROGRAM PROGA  
DIMENSION IREG (2)  
EQUIVALENCE (REG, IREG, IA), (IREG(2), IB)  
:  
:  
REG = EXEC (ICODE, ...)
```

IA = CONTENTS OF A REGISTER

IB = CONTENTS OF B REGISTER

RTEII/III-106

A PROGRAM MAY BE ABORTED BY THE SYSTEM IF THE PROGRAM MAKES AN ILLEGAL EXEC CALL.

EXEC CALL ERROR HANDLING BY A PROGRAM

FTN, L

**PROGRAM PROGA
 DIMENSION IREG (2)
 EQUIVALENCE (REG, IREG, IA), (IREG(2), IB)**

SETS SIGN BIT IN ICODE

**CALL EXEC (ICODE + 100000B ...)
 GO TO 10 ← (ERROR RETURN POINT)
 : ← (NO ERROR RETURN POINT)**

1

10

**REG = AB(J)
 CALL IER (IA, IB)**

**ERROR CODE FROM
 A AND B REGISTERS**

RTEII/III-107

Dummy routine for obtaining A and B Registers:

ASMB, L

**NAM AB
 ENT AB
 AB NOP
 STA TMP
 LDA AB,1
 STA AB
 LDA TMP
 JMP AB,1
 TMP NOP
 END**

Save A Register
 Get return address
 and make adjustment
 Restore A register
 Return

The following assembly code is generated for REG=AB(J)

**JSB AB
 DEF *+2
 DEF J
 JSB .DST
 DEF REG**

PROGRAM SCHEDULE

SCHEDULES A PROGRAM FOR EXECUTION. UP TO FIVE PARAMETERS MAY BE PASSED TO THE PROGRAM.

DIMENSION NAME (3)

DATA NAME(1), NAME(2), NAME(3) /2Hxx, 2Hxx, 2Hx^ /

ICODE = 9, 10, 23 OR 24

REG = EXEC (ICODE, NAME, IPRM1 . . . IPRM5)

OPTIONAL INTEGER
PARAMETERS

ICODE

* 9 = IMMEDIATE SCHEDULE, WITH WAIT

10 = IMMEDIATE SCHEDULE, NO WAIT

23 = QUEUE SCHEDULE, WITH WAIT

24 = QUEUE SCHEDULE, NO WAIT

*ON RETURN, A REGISTER CONTAINS SCHEDULED PROGRAM'S STATUS. 0 = DORMANT, 1 = SCHEDULED, ETC.

RTEII/III-108

PROGRAM SCHEDULE EXAMPLE

```

0001  FTN,L
0002      PROGRAM PROGA
0003      DIMENSION NAME(3),I(5),ICODE(5)
0004      EQUIVALENCE(REG,IREG)
0005      DATA NAME(1),NAME(2),NAME(3)/2HPR,2HOG,2HR /
0006      1,I(1),I(2),I(3),I(4),I(5)/0,1,2,3,4/
0007      CALL RMPAR(ICODE)
0008  10    REG=EXEC(ICODE,NAME,I(1),I(2),I(3),I(4),I(5))
0009      IF(IREG.EQ.0) GO TO 20
0010      PAUSE
0011      GO TO 10
0012  20    CALL RMPAR(I)
0013      WRITE(1,100) (I(J),J=1,5)
0014  100   FORMAT(5I6)
0015      END
0016      ENDS

```

```

0001  FTN,L
0002      PROGRAM PROGB
0003      DIMENSION I(5),K(5)
0004      DATA K(1),K(2),K(3),K(4),K(5)/5,6,7,8,9/
0005      CALL RMPAR(I)
0006      PAUSE
0007      WRITE(1,100) (I(J),J=1,5)
0008  100   FORMAT(5I2)
0009      CALL PRTN(K)
0010      END
0011      ENDS

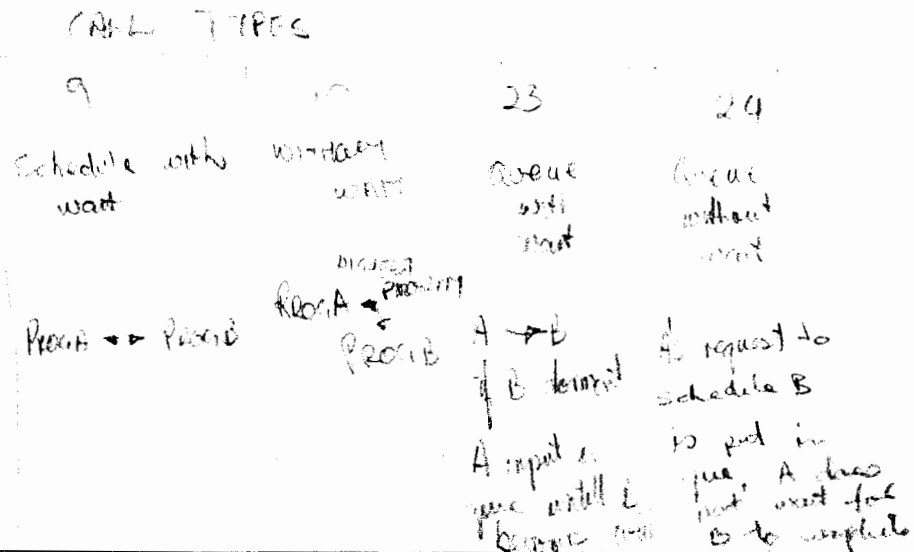
```

RTEII/III-109

PROGA schedules PROGB and passes it 5 integer parameters (line 8), then accepts 5 integers passed back from PROGB when PROGB completes (line 12).

PROGA tests the "A" register (line 9) to see if PROGB was actually scheduled by PROGA.

PROGB is scheduled by PROGA, accepts 5 integers from PROGA (line 5) and then returns 5 integers to PROGA (line 9).



SCHEDULE PROGB WITH WAIT.

```
*ON,PROGA,9
  PROGB 1 PAUSE 0000
*GO,PROGB
 0 1 2 3 4
  5   6   7   8   9
```

SCHEDULE PROGB WITH WAIT.

PROGB ABORTED OR OFFED BY "OF" OPERATOR REQUEST WHILE
PROGA WAS WAITING FOR PROGB TO COMPLETE. FIRST
RETURNED PARAMETER IS SET TO 1000000 (-32768 DECIMAL).

```
*ON,PROGA,9
  PROGB 1 PAUSE 0000
*OF,PROGB,1
PROGB ABORTED
-32768 0 0 0 0
```

SCHEDULE PROGB WITHOUT WAIT.

```
*ON,PROGA,10
  50 18432 18486 0 12349
  PROGB 1 PAUSE 0000
*GO,PROGB
 0 1 2 3 4
```

RTEII/III-110

SCHEDULE PROGB WITH WAIT AND QUEUE (PROGB NOT DORMANT),

```
*ON,PROGB
  PROGB : PAUSE   0000
*ON,PROGA,23
*ST,PROGA
  50 3 0  0  0  0  0  0
*ST,PROGB
  60 6 0  0  0  0  0  0
*OF,PROGB
  PROGB : PAUSE   0000
*GO,PROGB
  0 1 2 3 4
    5   6   7   8   9
```

SCHEDULE PROGRAM WITHOUT WAIT BUT WITH QUEUE (PROGB NOT DORMANT),

```
*ON,PROGB
  PROGB : PAUSE   0000
*ON,PROGA,24
*ST,PROGA
  50 3 0  0  0  0  0  0
*ST,PROGB
  60 6 0  0  0  0  0  0
*OF,PROGB
  50 18432 18488  0 12349
  PROGB : PAUSE   0000
```

RTEII/III-111

PROGRAM SEGMENT LOAD

TO LOAD A BACKGROUND SEGMENT OF THE
CALLING PROGRAM FROM THE DISC INTO
THE BACKGROUND OVERLAY AREA AND
TRANSFER EXECUTION TO THE SEGMENT

DIMENSION NAME (3)

NAME (1) = xxxxxB

NAME (2) = xxxxxB

NAME (3) = xxxxxB

CALL EXEC (8, NAME, IPRM1 . . . IPRM5)

OPTIONAL PARAMETERS
TO BE PASSED TO
SEGMENT

RTEII/III-112

EXAMPLE

FTN,L

```
PROGRAM HOW(3)
DIMENSION NAME(3)
DATA NAME/2HH0,2HW1,2H /
WRITE(1,10)
10 FORMAT("HOW ~")
CALL EXEC(8,NAME)
END
```



```
PROGRAM HOW1(5)
DIMENSION NAME(3)
DATA NAME/2HH0,2HW2,2H /
WRITE(1,10)
10 FORMAT("ARE ~")
CALL EXEC(8,NAME)
CALL HOW
END
```

```
PROGRAM HOW2(5)
WRITE(1,10)
10 FORMAT("YOU?")
CALL EXEC(6)
CALL HOW
END
END$
```

RTEII/III-113

PROGRAM SUSPEND

SUSPENDS THE CALLING PROGRAM UNTIL RESCHEDULED
BY THE GO OPERATOR REQUEST

CALL EXEC(7)

- OR -

PAUSE STATEMENT

RTEII/III-114

When the program is rescheduled with the "GO" operator request and parameters it may retrieve the parameters as follows:

DIMENSION IPRAM(5)

PAUSE

CALL RMPAR (IPRAM)

TIMED EXECUTION (INITIAL OFFSET)

PUTS A PROGRAM IN THE TIME LIST FOR SCHEDULING AT SPECIFIED TIME INTERVALS, STARTING AFTER AN INITIAL OFFSET TIME

DIMENSION NAME(3)

DATA NAME(1), NAME(2), NAME(3)/2Hxx, 2Hxx, 2Hx^/
CALL EXEC (12, NAME, IRESL, MTPLE, -IOFST)

NAME NAME OF PROGRAM TO PUT IN TIME LIST OR IF
NAME = Ø PUT CALLING PROGRAM IN TIME LIST

IRESL EXECUTION INTERVAL UNITS CODE
 1 = MILLISECONDS x 10
 2 = SECONDS
 3 = MINUTES
 4 = HOURS

- EXECUTION INTERVAL = IRESL x MTPLE (MTPLE ≤ 4095)
- INITIAL OFFSET = IRESL x -IOFST

RTEII/III-115

EXAMPLE

GO DORMANT FOR 10 SECONDS THEN EXECUTE EVERY ONE SECOND

		*ON, TIMER, NOW
	FTN,L	
	PROGRAM TIMER	13 41 22 42
	DIMENSION I(5)	13 41 32 46
	CALL RMPAR(1)	EXECUTED
	IF(I(1).NE.0) GO TO 10	13 41 33 50
	CALL TIME	EXECUTED
	CALL EXEC(12,0,2,1,-10)	13 41 34 50
10	CALL TIME	EXECUTED
	I(1)=1	13 41 35 51
	WRITE(1,100)	EXECUTED
100	FORMAT("EXECUTED")	13 41 36 48
	CALL EXEC(6,0,0,I(1))	EXECUTED
	END	13 41 37 49
	ENDS	EXECUTED
		13 41 38 49
		EXECUTED
		13 41 39 49
	FTN,L	
	SUBROUTINE TIME	EXECUTED
	DIMENSION I(5)	13 41 40 49
	CALL EXEC(11,I)	EXECUTED
	WRITE(1,100) (I(5-K),K=1,4)	
100	FORMAT(4I3)	
	END	
	ENDS	

RTEII/III-116

EXAMPLE

FTN,L

```
PROGRAM PROGA
DIMENSION NAME(3),I(5)
DATA NAME(1),NAME(2),NAME(3)/2HPR,2HOG,2HB /
CALL RMPAR(1)
CALL EXEC(12,NAME,I(1),I(2),I(3))
CALL TIME
END
ENDS
```

FTN,L

```
PROGRAM PROGB
CALL TIME
WRITE(1,10)
10 FORMAT("EXECUTED")
END
ENDS
```

```
*ON,PROGA,2,0,-5
13 47 59 14
13 48 4 17
```

**EXECUTE PROGB ONCE AFTER 5
SECONDS HAVE ELAPSED**

EXECUTED

```
*ON,PROGA,2,3,-10
13 46 56 45
13 47 6 49
```

**EXECUTE PROGB EVERY 3 SECONDS
AFTER 10 SECONDS HAVE ELAPSED**

EXECUTED

```
13 47 9 49
EXECUTED
```

RTEII/III-117

TIMED EXECUTION (ABSOLUTE START TIME)

PUTS A PROGRAM IN THE TIME LIST FOR SCHEDULING AT SPECIFIED TIME INTERVALS, STARTING AT AN ABSOLUTE TIME OF DAY

DIMENSION NAME(3)

DATA NAME(1), NAME(2), NAME(3)/2Hxx, 2Hxx, 2Hx /

CALL EXEC (12, NAME, IRESL, IMTPLE, IHRS, MINS, ISECS, MSECS x 10)

SAME AS INITIAL
OFFSET VERSION

ABSOLUTE START TIME,
HOURS, MINUTES, SECONDS,
MILLISECONDS x TEN

EXAMPLES

- EXECUTE ONCE AT 6:30 PM
CALL EXEC (12, NAME, 2, 0, 18, 30, 0, 0)
- EXECUTE EACH DAY AT 2:15 AM
CALL EXEC (12, NAME, 4, 24, 2, 15, 0, 0)

RTEII/III-118

PROGRAM COMPLETION

TERMINATES THE CALLING PROGRAM OR ANOTHER
(A SON) THAT WAS SCHEDULED BY THE CALLING
PROGRAM (THE FATHER)

TERMINATE SELF

DIMENSION IPRAM(5)

CALL RMPAR (IPRAM)

⋮

CALL EXEC (6, 0, INUMB, IPRAM(1), . . . IPRAM(5))

OPTIONAL

TERMINATE ANOTHER PROGRAM

DIMENSION NAME(3)

DATA NAME(1), NAME(2), NAME(3)/2Hxx, 2Hxx, 2Hz /

⋮

CALL EXEC (6, NAME, INUMB)

RTEII/III-119

INUMB -1 = SERIAL REUSABILITY COMPLETION
0 = NORMAL COMPLETION
1 = MAKE PROGRAM DORMANT BUT SAVE
SUSPENSION POINT AND RESOURCES
2 = SAME AS "OF; NAME, 0"
3 = SAME AS "OF, NAME, 1"

NAME NAME OF PROGRAM TO BE MADE DORMANT

IPRAM PARAMETERS TO BE SAVED FOR RETRIEVAL
BY RMPAR WHEN THE PROGRAM IS RESCHEDULED
(TERMINATE SELF)

FORTRAN AND ALGOL COMPILERS GENERATE A PROGRAM
COMPLETION EXEC CALL WHEN THEY COMPILE AN END OR
STOP STATEMENT.

JSB EXEC ICODE DEC 6
DEF * +2
DEF ICODE

RTEII/III-120

EXAMPLE

```
0001  FTN,L
0002  PROGRAM TERM
0003  DIMENSION I(5)
0004  CALL RMPAR(I)
0005  IF(I(1).NE.0) GO TO 10
0006  CALL EXEC(12,0,2,1,-1)
0007  10  I(1)=I(1)+1
0008  IF(I(1).EQ.10) GO TO 20
0009  CALL EXEC(6,0,0,I(1))
0010  20  CALL EXEC(6,0,2)
0011  END
0012  ENDS
```

WHEN PROGRAM TERM IS SCHEDULED BY ANOTHER PROGRAM, OR THE "ON" OPERATOR REQUEST, IT WILL EXECUTE 10 TIMES, AT ONCE PER SECOND. THE NUMBER OF EXECUTIONS IS SAVED IN ITS ID SEGMENT FROM I(1) WHEN TERM COMPLETES AND RETRIEVED BY RMPAR WHEN TERM EXECUTES.

RTEII/III-121

TIME REQUEST

RETURNS THE CURRENT TIME FROM THE REAL TIME CLOCK

DIMENSION ITIME(5)

CALL EXEC (11, ITIME, IYEAR)

↑
OPTIONAL

CONTENTS OF ITIME AND IYEAR AFTER CALL

ITIME(1) MILLISECONDS X10

ITIME(2) SECONDS

ITIME(3) MINUTES

ITIME(4) HOURS

ITIME(5) DAYS

IYEAR YEAR

RTEII/III-122

READ/WRITE

TRANSFERS DATA TO OR FROM AN I/O DEVICE

DIMENSION IBUFR (size)

ICODE = (1 = READ, 2 = WRITE)

ICNWD = (LOGICAL UNIT = AND TYPE OF TRANSFER)

IBUFL = (DATA BUFFER LENGTH. WORDS (+), CHARACTERS (-))

REG = EXEC (ICODE, ICNWD, IBUFR, IBUFL, IPRM1, IPRM2)

DATA
BUFFER
NAME

(OPTIONAL)

- DISC I/O (TRACK AND SECTOR)
- SUBSYSTEM PROGRAMMING
- CONTROL BUFFER NAME AND LENGTH

WHEN CALL COMPLETES:

"A" REGISTER = EQT WORD = 5

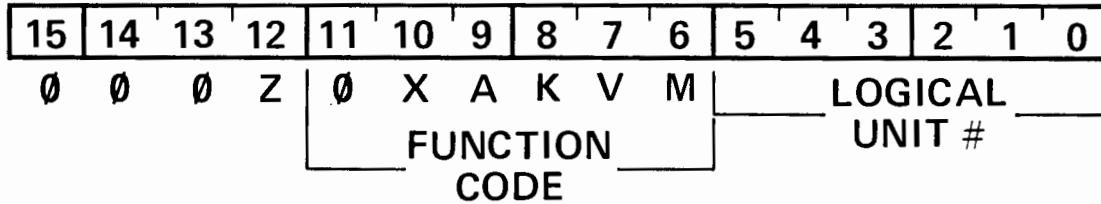
"B" REGISTER = TRANSMISSION LOG

THIS CALL IS GENERATED BY THE FORMATTER IN
RESPONSE TO AN ALGOL OR FORTRAN READ OR
WRITE STATEMENT.

RTEII/III-123

ICNWD

(control word)



LOGICAL UNIT # = 0 - 63 DECIMAL
IF = Ø, NO DATA IS TRANSFERRED.

FUNCTION CODE DETERMINES NATURE OF DATA TRANSFER
SEE RTE-II MANUAL FOR DETAILS

"Z" BIT = 1, IPRM1 IS A CONTROL BUFFER NAME AND IPRM2
ITS LENGTH

RTEII/III-124

SUBROUTINE REIO

REG = REIO (ICODE, ICNWD, IBUFR, IBUFL)

**A DISC RESIDENT PROGRAM IS SWAPPABLE IF IT
DOES ITS I/O THRU REIO**

– AND –

- **BUFFER SIZE < 130 WORDS**
- **BUFFER ADDRESS \geq 4TH WORD ABOVE
MEMORY PROTECT FENCE**

– IF NOT –

- **I/O IS DONE BUT PROGRAM IS NOT SWAPPABLE**
- **MAY BE USED FOR NON-DISC READ, WRITE, CONTROL**
- **A AND B REGISTER RETURN INFORMATION SAME AS
STANDARD EXEC CALLS**

RTEII/III-125

DISC READ/WRITE

CALL EXEC(ICODE,ICNWD,IBUFR,IBUFL,ITRK,ISECT)

SAME AS NON-DISC I/O

DISC
TRACK#
AND
SECTOR #

ITRK DISC TRACK #, 0 - 202 (7900), 0-255 (7905)

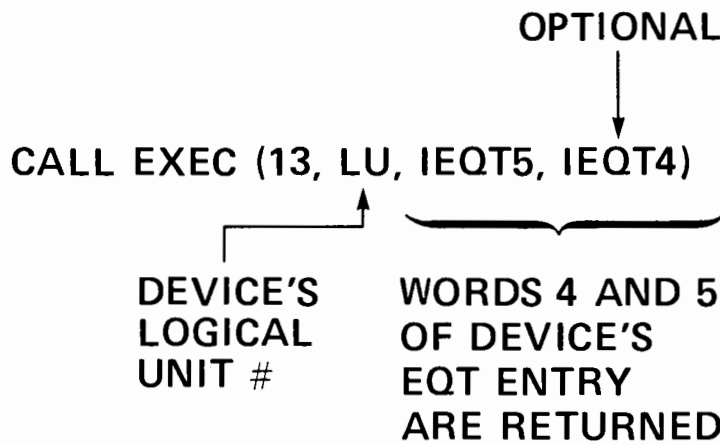
ISECT SECTOR # 0 - 95

THE HP 7900 AND HP 7901 DISCS HAVE 203 TRACKS PER
PLATTER, HP 7905 DISC HAS 256 TRACKS PER LOGICAL
PLATTER.

RTEII/III-126

I/O STATUS

TO REQUEST THE STATUS OF A DEVICE PRIOR TO ISSUING
A READ, WRITE OR CONTROL REQUEST TO IT



RTEII/III-127

I/O CONTROL

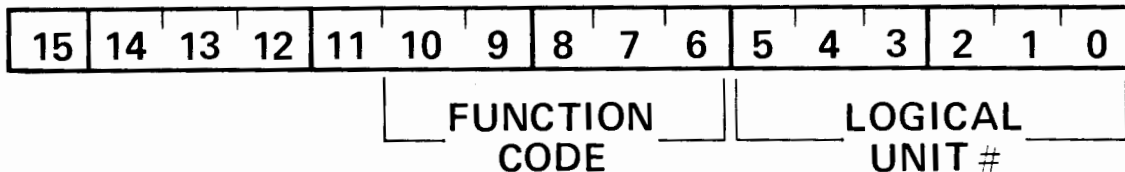
TO PERFORM VARIOUS I/O CONTROL OPERATIONS SUCH AS
BACKSPACE, WRITE END-OF-FILE, REWIND, ETC.

ICNWD = (LOGICAL UNIT # AND TYPE OF OPERATION)

CALL EXEC (3, ICNWD, IPRAM)

(OPTIONAL)
LIST LINE SPACING

ICNWD



SEE RTE-II MANUAL FOR A LIST OF FUNCTION CODES.

RTEII/III-128

LABORATORY EXERCISE GUIDE #1

Objective

To provide an exercise in using the RTE-II Exec calls through writing a program.

Problem

Write a program, TIMER, for the RTE-II that will schedule another program, XXXX, and compute its elapsed execution time.

TIMER is to request, on the system TTY, the name and any parameters required by program XXXXX.

If program XXXXX cannot be scheduled TIMER is to suspend itself to allow a re-try. When TIMER is rescheduled it is to be passed a parameter where: if its value is 0 then re-try, if it is 1 then terminate.

When XXXXX completes, TIMER is to print XXXXX's elapsed execution time.

Example:

```
*ON, TIMER
PROGRAM NAME? MEM
PARAMETERS?
MEM NOT READY.
TO TRY AGAIN . . . GO, TIMER, 0 ELSE 1.
    TIMER : PAUSE 0000
*OF, MEM
*GO, TIMER, 0
PROGRAM MEM RUN TIME = 0:0:5.26
```

LABORATORY EXERCISE GUIDE #2

OBJECTIVE

To provide an exercise using the RTE Exec calls through writing a program.

PROBLEM

A message coded in ASCII has been written somewhere on the disc. You are to write a program that will find this message and print the message correctly on the list output device.

HP1B

The complete message is stored on one track. The correct track contains the ASCII code for "HP" in the word 0 of sector 0 and also in the word 0 of the last sector. Each of the 10 lines are stored on a sector within that track. Word 0 of each correct sector contains the ASCII number of that line. Each line contains 60 characters.

PROCEDURE

Create a program that will provide a solution to the problem outlined.

Remember any program can Read from an 'own' Disc track but only the "holder" of the track may Write on it.

When the program prints the correct message it should make a completion call.

The results should agree with the sample print-out below.

```
1 THE HP 9600 REAL-TIME EXECUTIVE CUSTOM TAILORES AN HP
2 21MX COMPUTER AND DISC STORAGE INTO A TRUE REAL-TIME
3 MULTI-PROGRAMMING SYSTEM.
4
5 PROGRAMS FOR THE 9600 CAN BE WRITTEN USING THE HP RT
6 ASSEMBLY LANGUAGE OR HP REAL-TIME FORTRAN. THE NUMBER OF
7 PROGRAMS IN THE SYSTEM IS LIMITED ONLY BY DISC CAPACITY
8 AND THE DYNAMICS OF THE APPLICATION. MEMORY IS MINI-
9 MIZED BY CONFIGURING A TAILORED SYSTEM FROM STANDARD
10 MODULES.
```

TRACK ALLOCATION

TO REQUEST THAT RTE-II ASSIGN A SPECIFIC NUMBER OF CONTIGUOUS DISC TRACKS FOR DATA STORAGE

ICODE = 4, ASSIGN TRACKS TO PROGRAM

= 15, ASSIGN TRACKS GLOBALLY

ITRAK = NUMBER OF TRACKS REQUIRED

CALL EXEC (ICODE, ITRAK, ISTRK, IDISC, ISECT)

RETURNED INFORMATION

- STARTING TRACK #
- DISC LOGICAL UNIT #
- NUMBER OF 64 WORD SECOTRS/TRACK

IF BIT 15 OF ITRAK = 0, PROGRAM WILL SUSPEND IF TRACKS ARE NOT AVAILABLE; IF BIT 15 = 1 THE PROGRAM WILL NOT SUSPEND AND ISTRK WILL SET TO -1.

RTEII/III-129

DISC TRACK RELEASE

TO RELEASE SOME CONTIGUOUS DISC TRACKS

ITRAK = NUMBER OF TRACKS TO RELEASE

ISTRK = STARTING TRACK NUMBER

IDISC = DISC LOGICAL UNIT NUMBER

PROGRAM TRACKS

CALL EXEC (5,ITRAK,ISTRK,IDISC)

NOT REQUIRED
IF ITRAK = -1
(RELEASE ALL TRACKS)

GLOBAL TRACKS

REG = EXEC (16,ITRAK,ISTRK,IDISC)

ON RETURN:

"A" REGISTER = 0 TRACKS RELEASED
= -1 TRACKS NOT RELEASED, ONE
OR MORE IN USE
= 2 TRACKS NOT RELEASED, ONE
OR MORE NOT GLOBAL

RTEII/III-130

Release Tracks



RELEASE ALL DISC TRACKS ASSIGNED TO A PROGRAM

* RT, program name

RTEII/III-131

PROGRAM SWAPPING CONTROL

ALLOWS A DISC RESIDENT PROGRAM TO LOCK ITSELF INTO CORE TO PREVENT IT FROM BEING SWAPPED.

CALL EXEC (22,IOPTN)

- ↑
- Ø = PROGRAM MAY BE SWAPPED
 - 1 = PROGRAM MAY NOT BE SWAPPED
 - 2 = SWAP JUST THE PROGRAM AREA
 - 3 = SWAP ALL OF THE DISC RESIDENT AREA

PROGRAMS MAY NOT BE LOCKED INTO CORE UNLESS CORE LOCK WAS SPECIFIED AT SYSTEM GENERATION TIME.

RTEII/III-133

RESOURCE MANAGEMENT (RESOURCE NUMBERING)

ALLOWS COOPERATING PROGRAMS A METHOD OF SHARING RESOURCES

CALL RNRQ (ICODE, IRN, ISTAT)

ICODE =

15	14	5	4	3	2	1	0
WAIT OPTION		ALLOCATE OPTION			SET OPTION		
N O W A I T	N O A B O R T	C L E A R	G L O B A L	L O C A L	C L E A R	G L O B A L	L O C A L

IRN = RESOURCE NUMBER. RETURNED ON ALLOCATE;
REQUIRED OTHERWISE.

ISTAT = (RETURNED)

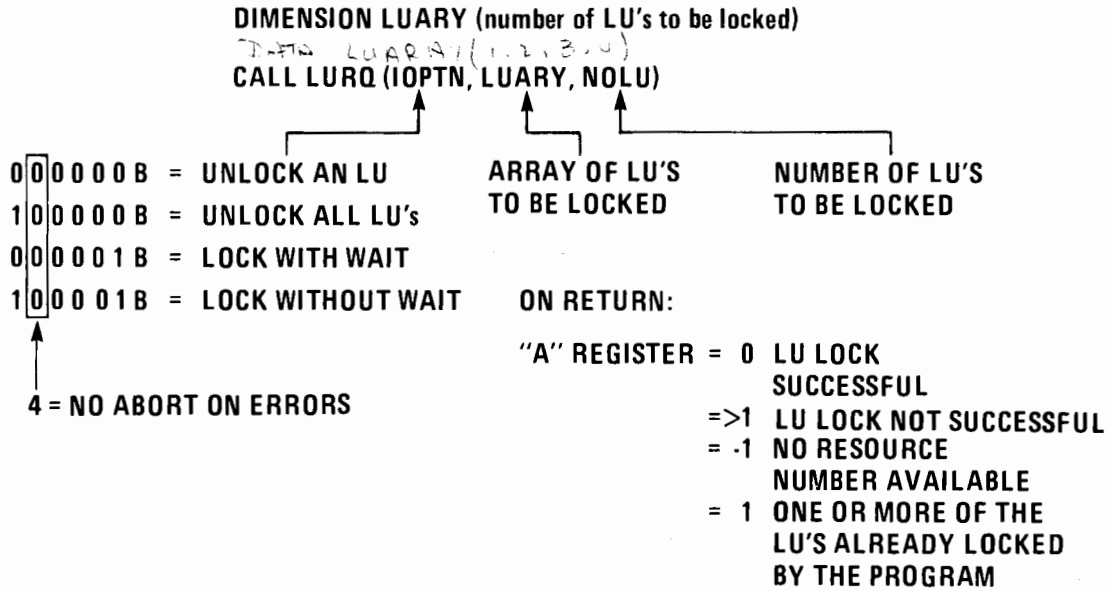
- 0 NORMAL DEALLOCATE
- 1 RN IS CLEAR (UNLOCKED)
- 2 RN IS LOCKED LOCALLY TO CALLER
- 3 RN IS LOCKED GLOBALLY
- 4 NO RN AVAILABLE NOW
- 5 —
- 6 RN IS LOCKED LOCALLY TO ANOTHER PROGRAM
- 7 RN IS LOCKED GLOBALLY

RTEII/III-134

ie Resource numbers are kept in operating mode

LOGICAL UNIT LOCK

ALLOWS A PROGRAM TO EXCLUSIVELY DOMINATE (LOCK) A GROUP OF I/O DEVICES



THIS CALL USES RESOURCE NUMBERS

RTEII/III-135

<p><u>Buffering</u></p> <p>(2, DURS, B, X > 13)</p> <p><u>SAM</u></p> <p>REIO</p> <p>CALL REIO (2, LU, IOE)</p> <p>Prog is swappable</p> <p>use with "call code"</p> <hr/> <p>CALL EXEC (2, MI)</p> <p><u>SWAPPABLE</u></p> <hr/> <p>CLASS I/O</p> <p>MAILBOY I/O</p>	<p>INPUT</p> <p>X</p> <p>Prog. is suspended until read is satisfied</p> <p>✓</p> <p>✓</p> <p>✓</p>	<p>OUTPUT</p> <p>✓</p> <p>✓</p> <p>✓</p> <p>✓</p>
--	--	---

WRITE / READS
FOR MAILING LIST IS
SOME PROGRAMS

CLASS I/O-READ/WRITE re Sam.

TRANSFERS DATA TO OR FROM AN I/O DEVICE. THE CALLING PROGRAM NORMALLY DOES NOT WAIT FOR THE CALL TO COMPLETE.

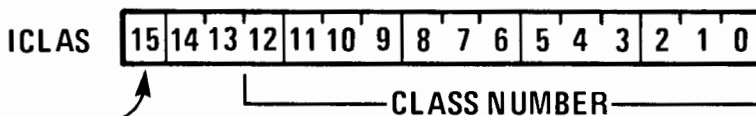
DIMENSION IBUF (size)

ICODE = (17, READ; 18, WRITE; 20, WRITE THEN READ)

ICLAS = 0, ALLOCATE A CLASS NUMBER;
1-255, A CLASS NUMBER TO USE

CALL EXEC (ICODE, ICNWD, IBUF, IBUFL, IPRM1, IPRM2, ICLAS)

SAME AS FOR ICODE = 1 OR 2 (IBUF IS A DUMMY VARIABLE FOR ICODE = 17) USER INFORMATION PASSED TO GET CALL



NO WAIT BIT } = 0, PROGRAM IS PUT IN GENERAL WAIT LIST (STATE 3) IF MEMORY OR CLASS NUMBER NOT AVAILABLE.
= 1, "A" REGISTER = -1, NO CLASS NUMBER AVAILABLE
"A" REGISTER = -2, NO MEMORY AVAILABLE

ON RETURN FROM CALL

RTEII/III-136

Dimension IBuf(10)

Data IBuf/2u

ICLAS = 0

CALL EXEC (20, 0, IBuf, 10, Iparm1, Iparm2, Iclass)

CALL EXEC (21, ICLASS + 100000, IBUF, 10)

↑
sets bit 14

Iparm1 → IBUF

CLASS I/O - I/O CONTROL

TO PERFORM VARIOUS I/O CONTROL OPERATIONS SUCH AS BACKSPACE, WRITE END-OF-FILE, REWIND, ETC.. THE CALLING PROGRAM NORMALLY DOES NOT WAIT FOR THE CALL TO COMPLETE.

CALL EXEC (19, ICNWD, IPRAM, ICLAS)

SAME AS FOR STANDARD I/O CONTROL CALL (REQUEST CODE = 3)	SAME AS FOR CLASS I/O - READ/WRITE
--	--

RTEII/III-137

USE OF I/O CONTROL

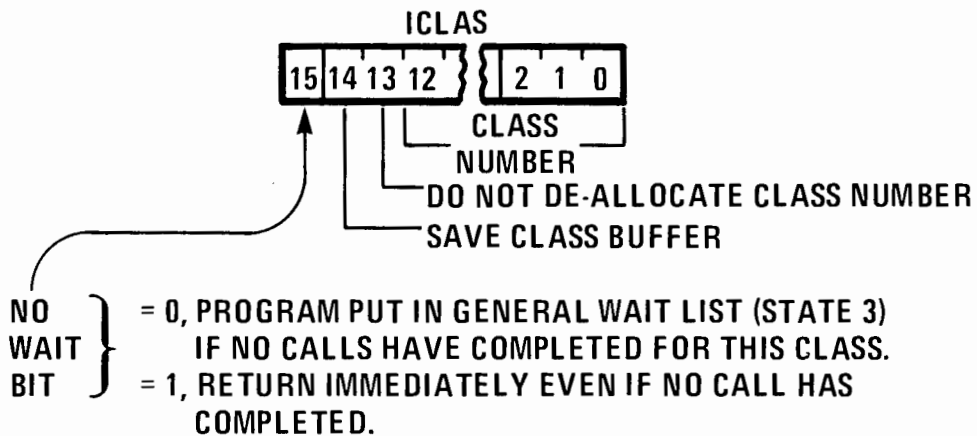
IPRAM

ICNWD

CLASS I/O-GET

COMPLETES THE DATA TRANSFER BETWEEN THE SYSTEM AND USER PROGRAM THAT WAS PREVIOUSLY INITIATED BY A CLASS REQUEST.

CALL EXEC (21, ICLAS, IBUFR, IBUFL, IRTN1, IRTN2, IRTN3)



AND B REGISTERS
ON RETURN:

SUCCESSFUL GET

A REG, BIT 15 = 0
 A REG = STATUS
 B REG = TRANSMISSION LOG

UNSUCCESSFUL GET

A REG, BIT 15 = 1
 A REG = NUMBER OF REQUESTS NOT COMPLETED FOR THIS CLASS

RTEII/III-138

CLASS I/O-GET (cont.)

IBUFR DATA IS RETURNED HERE FROM CLASS READ (17)
OR WRITE/READ (20) CALLS. IT IS A DUMMY VARIABLE
FOR CLASS WRITE (18) AND CONTROL (19) CALLS.

IBUFL DATA BUFFER LENGTH; WORDS (+), CHARACTERS (-)

IRTN1,
IRTN2 USER INFORMATION PASSED FROM CLASS READ, WRITE
OR WRITE/READ CALLS

IRTN3 REQUEST CODE RECEIVED BY DRIVER RETURNED HERE

<u>ORIGINAL REQUEST CODE</u>	<u>VALUE RETURNED IN IRTN3</u>
17/20 (READ, WRITE/READ)	1
18 (WRITE)	2
19 CONTROL	3

EXECUTIVE ERROR CODES

WHEN THE EXECUTIVE DISCOVERS AN ERROR IN AN EXEC CALL, IT MAY TERMINATE THE PROGRAM; RELEASE ANY DISC TRACKS ASSIGNED TO THE PROGRAM; PRINT AN ERROR MESSAGE ON THE OPERATOR CONSOLE, (name ABORTED); AND PROCEED TO EXECUTE THE NEXT PROGRAM IN THE SCHEDULED LIST.

MEMORY PROTECT ERRORS

MEMORY PROTECT VIOLATIONS THAT ARE NOT CALLS TO THE EXEC CAUSE THE FOLLOWING MESSAGE:

MP name address (address IS VIOLATING INSTRUCTION ADDRESS)

REQUEST CODE ERRORS

RQ name address (address IS THE ADDRESS OF THE ILLEGAL REQUEST)

THE GENERAL FORM OF OTHER ERROR CODES

type name address

WHERE: *type = A 4 CHARACTER ERROR CODE
name = THE PROGRAM NAME
address = THE ADDRESS OF THE CALL

RTEII/III-140

*SEE RTE-II MANUAL FOR LIST OF CODES

LAR

- 1 INPUT MESSAGE FROM YOUR SYSTEM
- 2 READ IN THE SECOND MESSAGE
- 3 FIND OUT WHICH OF YOUR PROGRAMS
TRIP POINTED PROGRAM
- 4 PASS IT TO THE END PROG POINT IT SHOULD
AN CLEAN UP
- 5 PRINT CLASS NUMBER
AND NUMBER

LABORATORY EXERCISE GUIDE #3

OBJECTIVE

To provide an exercise in using the RTE-II Class I/O Exec calls.

PROBLEM

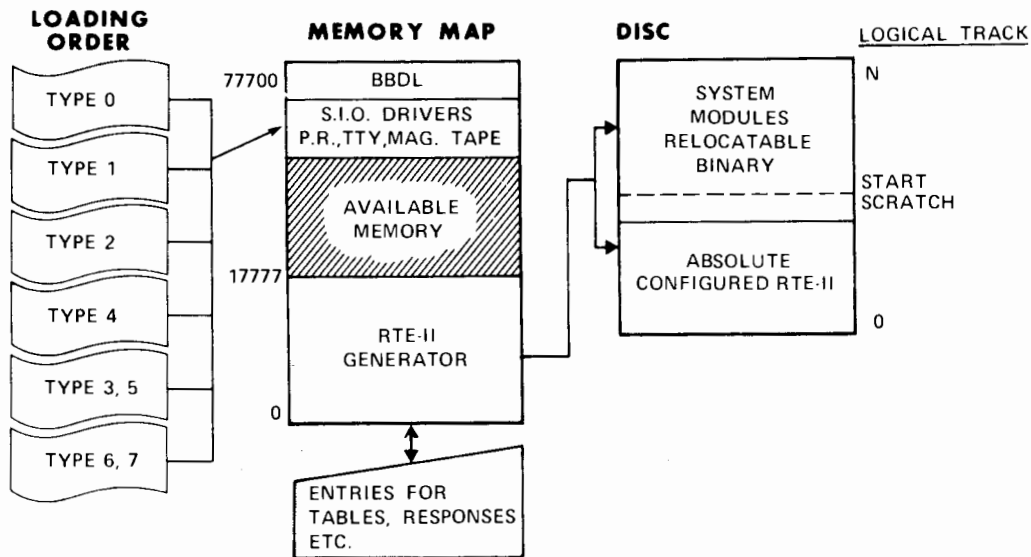
Complete the partial programs, PROGA and PROGB, listed below. PROGA is to read a record from the TTY keyboard and pass the record to PROGB. PROGB is to print the received record on the TTY.

Verify your programming efforts by compiling, loading and executing your completed PROGA and PROGB.

```
FTN,L
      PROGRAM PROGA
      DIMENSION IBFR(32), NAME (3)
      DATA NAME (1), NAME (2), NAME (3)/2HPR, 2HOG, 2HBW/
      .
      .
      .
C     DO CLASS WRITE/READ TO LU 0.
      ICLAS = 0
      CALL EXEC (20,0,IBFR,-64,IDMY,JDMY,ICLAS)
C
C     SCHEDULE RECEIVING PROGRAM, PASS IT CLASS NO.
C
      CALL EXEC (10,NAME,ICLAS)
      .
      .
      .
      END
```

```
FTN,L
      PROGRAM PROGB
      DIMENSION IBFR (32), IPRAM (5)
C
C     SAVE CLASS NO., IPRAM (1)
C
      CALL RMPAR (IPRAM)
C
C     ACCEPT DATA FROM PROG A USING CLASS GET CALL
C     AND RELEASE THE CLASS NO.
C
      CALL EXEC (21,IPRAM (1),IBFR,-64)
      .
      .
      .
      END
```


SYSTEM GENERATION (TAPE)



- THE SYSTEM GENERATOR IS LOADED INTO MEMORY USING THE BBL.
- INITIALIZATION PHASE - ESTABLISHES DISC SIZE, TYPE, SYSTEM HARDWARE INFO.
- PROGRAM INPUT PHASE - SYSTEM AND USER PROGRAMS ARE COPIED ON THE DISC.
- PARAMETER INPUT PHASE - PROGRAM PRIORITIES AND TYPE CODES MAY BE CHANGED.
- DISC LOADING PHASE - ALL TABLES ARE CONSTRUCTED AND THE ABSOLUTE SYSTEM IS CREATED ON THE SYSTEM DISC/DRUM.

RTEII/III-141

- A. ASSIGN ALL DEVICES THAT REQUIRE PRIVILEGED INTERRUPT IN ORDER OF DECREASING SPEED.
- B. AFTER THE PRIVILEGED DEVICES, ASSIGN THE PRIVILEGED INTERRUPT I/O CARD HP 12620.
- C. ASSIGN THE TBG I/O CARD HP 12539.
- D. ASSIGN ALL DEVICES THAT *DO NOT* USE DMA IN ORDER OF DECREASING SPEED.
- E. ASSIGN ALL DEVICES THAT *DO* USE DMA IN ORDER OF DECREASING SPEED.
- F. IF AN I/O EXTENDER IS REQUIRED AND THE EXTENDER DOES NOT HAVE DMA CAPABILITY, THE ORDER OF STEPS E AND F CAN BE REVERSED SO THAT ALL DMA DEVICES ARE IN THE COMPUTER MAINFRAME. IF THIS STEP IS NECESSARY, MAINTAIN THE SAME RELATIVE ORDER OF SPEED ASSIGNMENT AMONG THE DMA AND NON-DMA DEVICES.

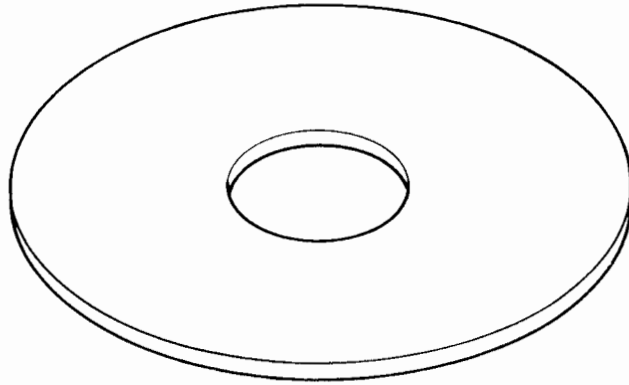
SELECT CODE ASSIGNMENTS

RTEII/III-142

HP 7900 Moving Head Disc Worksheet

SUBCHANNEL 1

REMOVABLE

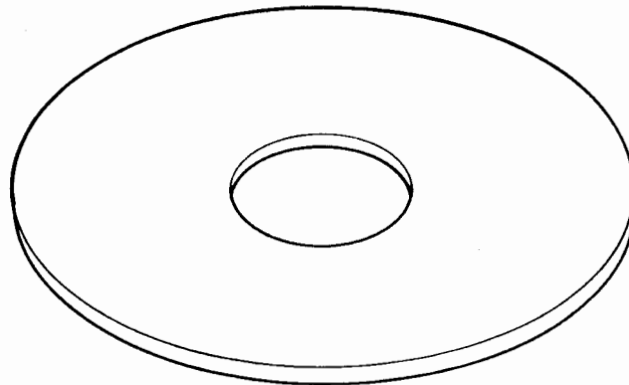


NO. OF TRACKS AVAILABLE _____

FIRST TRACK _____

SUBCHANNEL 0

FIXED



NO. OF TRACKS AVAILABLE _____

FIRST TRACK _____

SYSTEM SUBCHANNEL NUMBER _____ (LOGICAL UNIT 2)

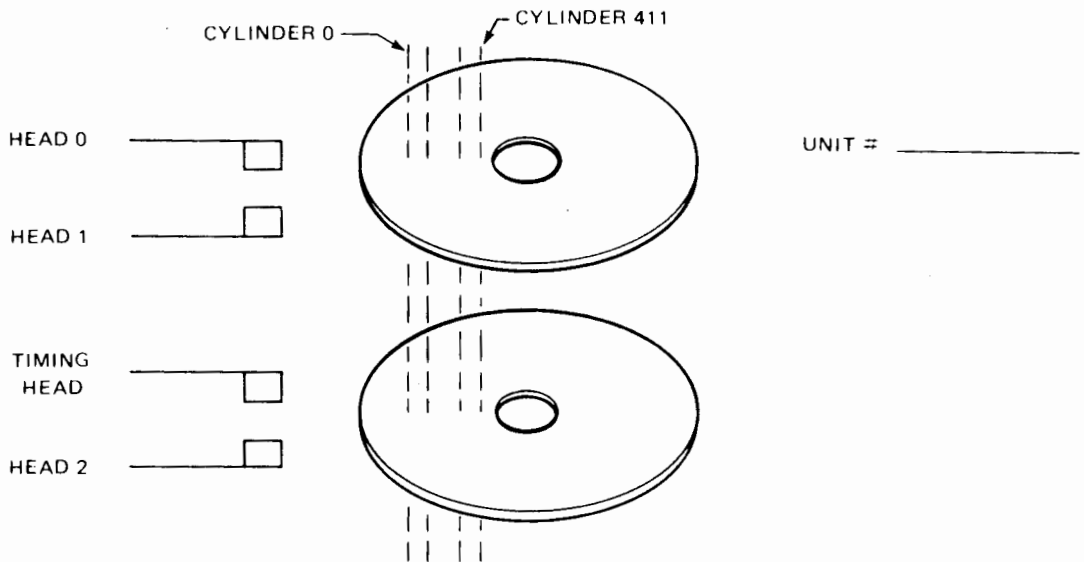
AUXILIARY SUBCHANNEL NUMBER _____ (LOGICAL UNIT 3)

SCRATCH SUBCHANNEL NUMBER _____

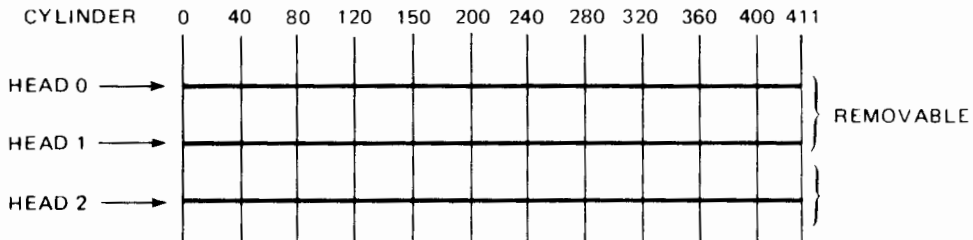
START SCRATCH (I.E. 1ST TRACK = 0) _____

HP 7905 Disc Worksheet

FILL IN UNIT NUMBER:



TRACKS SHOWN END-TO-END ON THREE SURFACES—CIRCLE SUBCHANNELS:



TRANSLATE STEP 2 TO NUMBERS:

SUBCHANNEL							
NUMBER OF TRACKS							
STARTING CYLINDER							
STARTING HEAD							
NUMBER OF SURFACES							
NUMBER OF SPARES							
SYSTEM ? (✓)							
AUXILIARY (✓)							
SCRATCH ? (✓)							

RTE-II

HP 7905 Moving Head Disc Initialization

INITIALIZATION PHASE CONTROLLER CHAN?

#TRKS, FIRST CYL#, HEAD, #SURFACES, UNIT, #SPARES
FOR SUBCHNL

_____ 0? _____

_____ 1? _____

_____ 2? _____

_____ 3? _____

_____ 4? _____

_____ 5? _____

_____ 6? _____

_____ 7? _____

/E
128 WORD SECTORS/TRACK?

SYSTEM SUBCHNL?

SCRATCH SUBCHNL?

AUX DISC (YES OR NO OR #TRKS)?

START SCRATCH?

TBG CHNL?

PRIV. INT. CARD ADDR?

FG SWAPPING?

BG SWAPPING?

FG CORE LOCK?

BG CORE LOCK?

SWAP DELAY?

LWA MEM?

PRGM INPT?

LIBR INPT?

PRAM INPT?

INITIALIZE SUBCHNL:

0?

1?

2?

3?

4?

5?

6?

7?

PUNCH BOOT?

TPRTE-20

RTE-III

Generator Input Worksheet

INITIALIZATION PHASE

7900/7901 DISC

MH DISC CHNL?

#TRKS, FIRST TRK ON SUBCHNL:

0?

1?

2?

3?

4?

5?

6?

7?

/E

#128 WORD SECTORS/TRACK?

SYSTEM SUBCHNL?

SCRATCH SUBCHNL?

AUX DISC (YES OR NO OR #TRKS)?

START SCRATCH?

TBG CHNL?

PRIV. INT. CARD ADDR?

PRIV. DRIVERS ACCESS COMMON?

7905 DISC

CONTROLLER CHAN

TRKS, FIRST CYL =, HEAD, = SURFACES, UNIT, = SPARES FOR SUBCNL

0?

1?

2?

3?

4?

5?

6?

7?

/E

FG CORE LOCK?

BG CORE LOCK?

SWAP DELAY?

MEM SIZE?

PRGM INPT?

LIBR INPT?

PRAM INPT?

INITIALIZE SUBCHNL:

0?

1?

2?

3?

4?

5?

6?

7?

PUNCH BOOT?

TPRTE-20

RTE-II

System Configuration Worksheet

PROGRAM INPUT PHASE

CORE RESIDENT SYSTEM
I/O DRIVERS
USER'S SYSTEM PROGRAMS
FOREGROUND CORE RESIDENT PROGRAMS
FOREGROUND DISC RESIDENT PROGRAMS
BACKGROUND CORE RESIDENT PROGRAMS
BACKGROUND DISC RESIDENT PROGRAMS
AND THEIR RESPECTIVE SEGMENTS
LIBRARY PROGRAMS
UTILITY PROGRAMS
SUBROUTINES

NO UNDEF EXTS

PARAMETER INPUT PHASE

NAME, TYPE [,PR [, RES [, MULT [, HR, MIN, SEC,
10'S/MS]]]]

/E

CHANGE ENTS?

/E

OF BLANK ID SEGMENTS:

OF BLANK BG SEG. ID SEGMENTS?

FWA BP LINKAGE?

TABLE GENERATION PHASE

* # OF I/O CLASSES?

* # OF LU MAPPINGS?

* # OF RESOURCE NUMBERS?

* BUFFER LIMITS (LOW, HIGH)?

* EQUIPMENT TABLE ENTRY

EQT 01?

_____, DV _____, _____, T = _____, X = _____

EQT 02?

_____, DV _____, _____, T = _____, X = _____

EQT 03?

_____, DV _____, _____, T = _____, X = _____

EQT 04?

_____, DV _____, _____, T = _____, X = _____

EQT 05?

_____, DV _____, _____, T = _____, X = _____

EQT 06?

_____, DV _____, _____, T = _____, X = _____

EQT 07?

_____, DV _____, _____, T = _____, X = _____

EQT 08?

_____, DV _____, _____, T = _____, X = _____

EQT 09?

_____, DV _____, _____, T = _____, X = _____

EQT 10?

_____, DV _____, _____, T = _____, X = _____

/E



RTE-III

Generator Input Worksheet (Continued)

PROGRAM INPUT PHASE

CORE RESIDENT SYSTEM
I/O DRIVERS
USER'S SYSTEM PROGRAMS
FOREGROUND CORE RESIDENT PROGRAMS
FOREGROUND DISC RESIDENT PROGRAMS
BACKGROUND CORE RESIDENT PROGRAMS
BACKGROUND DISC RESIDENT PROGRAMS
AND THEIR RESPECTIVE SEGMENTS
LIBRARY PROGRAMS
UTILITY PROGRAMS
SUBROUTINES

NO UNDEF EXTS

PARAMETER INPUT PHASE

NAME, TYPE [, PR [, RES [, MULT [, HR, MIN, SEC,
10'S/MS]]]]

/E

CHANGE ENTS?

/E

OF BLANK ID SEGMENTS

OF BLANK BG SEG. ID SEGMENTS?

MAX NUMBER OF PARTITIONS?

FWA BP LINKAGE?

SYSTEM LOADING PHASE
(NO USER INPUT)

TABLE GENERATION PHASE

*# OF I/O CLASSES?

* # OF LU MAPPINGS?

* # OF RESOURCE NUMBERS?

* BUFFER LIMITS (LOW, HIGH)?

* EQUIPMENT TABLE ENTRY

EQT 01?

-----, DV -----, I = -----, X = -----

EQT 02?

-----, DV -----, I = -----, X = -----

EQT 03?

-----, DV -----, I = -----, X = -----

EQT 04?

-----, DV -----, I = -----, X = -----

EQT 05?

-----, DV -----, I = -----, X = -----

EQT 06?

-----, DV -----, I = -----, X = -----

EQT 07?

-----, DV -----, I = -----, X = -----

EQT 08?

-----, DV -----, I = -----, X = -----

EQT 09?

-----, DV -----, I = -----, X = -----

EQT 10?

-----, DV -----, I = -----, X = -----

E

RTE-II/III

Table 6-6. Generator Input Worksheet (Continued)

• DEVICE REFERENCE TABLE

1 = EQT #? (SYSTEM TELEPRINTER)

2 = EQT #? (SYSTEM MASS STORAGE)

3 = EQT #? (AUXILIARY MASS STORAGE)

4 = EQT #? (STANDARD PUNCH UNIT)

5 = EQT #? (STANDARD INPUT UNIT)

6 = EQT #? (STANDARD LIST UNIT)

7 = EQT #?

8 = EQT #? (MAG TAPE RECOMMENDED)

9 = EQT #?

10 = EQT #?

11 = EQT #?

12 = EQT #?

13 = EQT #?

14 = EQT #?

15 = EQT #?

/E

• INTERRUPT TABLE

/E

RTE-II

System Configuration Worksheet (continued)

SYSTEM BOUNDARIES PHASE

LIB ADDRS _____ (A)

CHANGE LIB ADDRS?

FG COMMON _____ (B)

CHANGE FG COMMON?

FG RES ADD _____ (C)

CHANGE FG RES ADD?

FG DSC ADD _____ (D)

CHANGE FG DSC ADD?

BP LINKAGE _____ (X)

CHANGE BP LINKAGE?

SYS AVMEM _____ (E)

CHANGE SYS AVMEM?

BG BOUNDRY _____ (F)

CHANGE BG BOUNDRY?

BG COMMON _____ (G)

CHANGE BG COMMON?

BG RES ADD _____ (H)

CHANGE BG RES ADD?

BG DSC ADD _____ (I)

CHANGE BG DSC ADD?

SYSTEM STORED ON DISC

SYS SIZE: _____ TRKS, _____ SECS (10)

RTE-III

Generator Input Worksheet (Continued)

RT COMMON XXXXX
CHANGE RT COMMON?

MODIFY PROGRAM PAGE REQUIREMENTS?

_____ / _____

RT COMMON XXXXX
BG COMMON XXXXX
CHANGE BG COMMON?

_____ / _____

_____ / _____

BG COMMON XXXXX
LWA BG COMMON XXXXX
ALIGN AT NEXT PAGE?

_____ / _____

_____ / _____

LWA BG COMMON XXXXX

_____ / _____

PARTITION DEFINITION PHASE

LWA MEM RESIDENT PROG AREA XXXXX
ALIGN AT NEXT PAGE?

_____ / _____

_____ / _____

/E

LWA MEM RESIDENT PROG AREA XXXXX
SYS AV MEM; XXXXX WORDS
1ST DSK PG XXXXX
CHANGE 1ST DSK PG?

ASSIGN PROGRAM PARTITIONS?

_____ / _____

_____ / _____

SYS AV MEM; XXXXX WORDS
PAGE REMAINING: XXXXX

_____ / _____

_____ / _____

DEFINE PARTITIONS

_____ / _____ / _____ / _____

_____ / _____

_____ / _____ / _____ / _____

_____ / _____

/E

_____ / _____ / _____ / _____

SYSTEM STORED ON DISC

SYS SIZE: _____ TRKS; _____ SECS (10)

_____ / _____ / _____ / _____

_____ / _____ / _____ / _____

_____ / _____ / _____ / _____

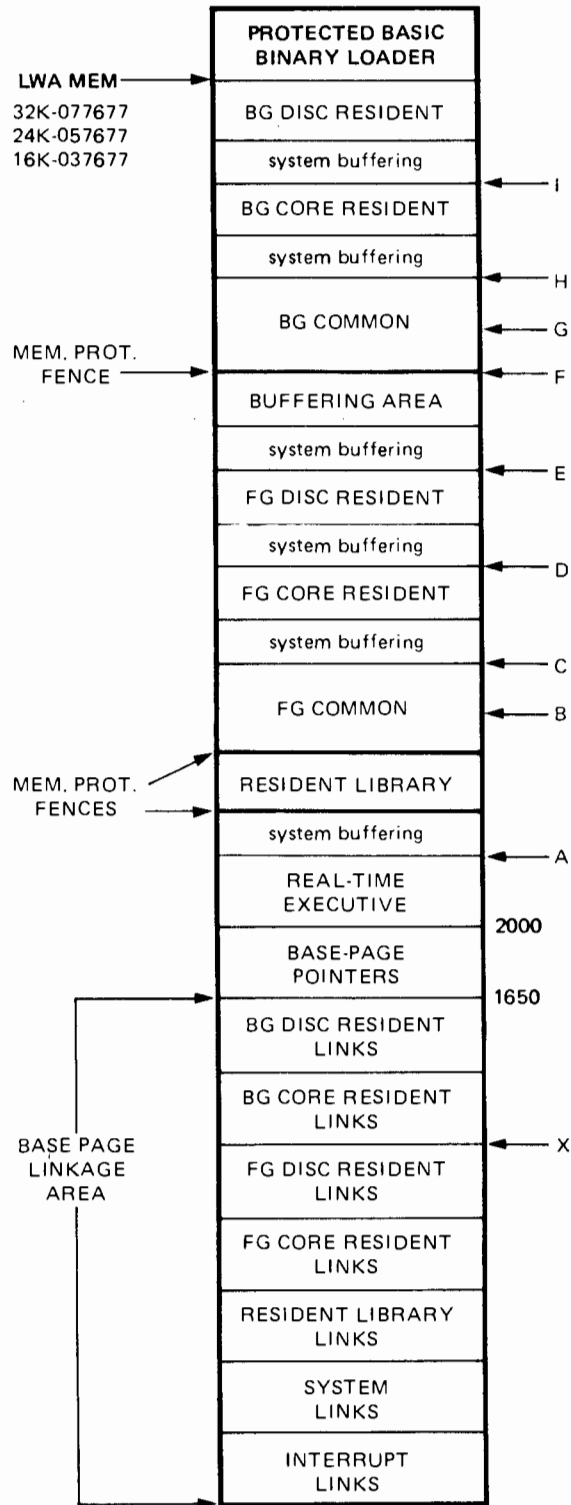
_____ / _____ / _____ / _____

_____ / _____ / _____ / _____

_____ / _____ / _____ / _____

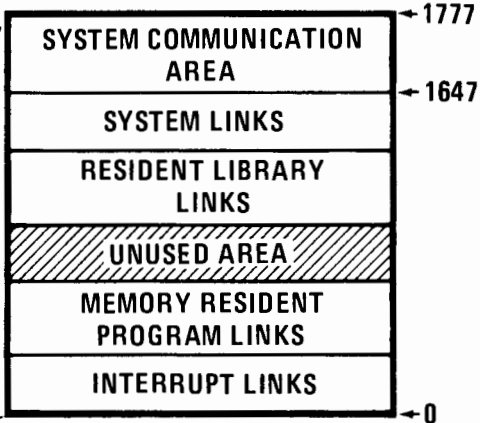
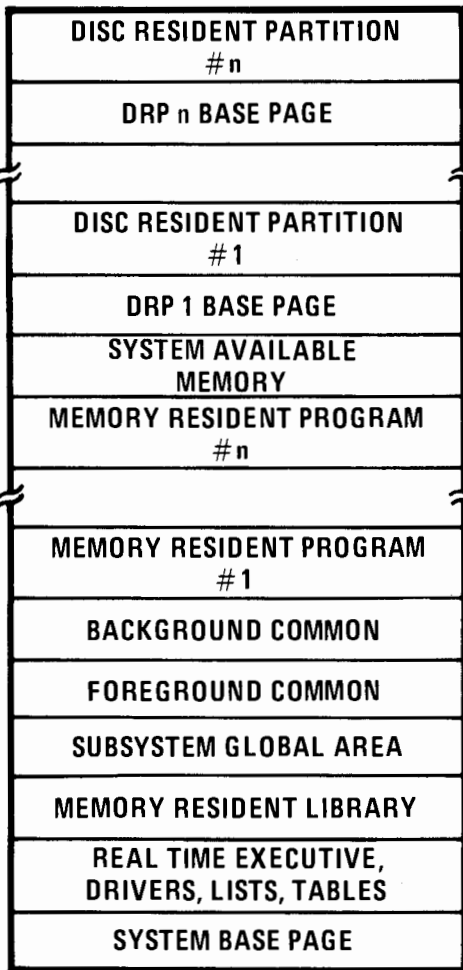
/E

RTE-II



RTE-III PHYSICAL SYSTEM MEMORY MAP

256K MAX →



RTEII/III-143

RESOURCE MANAGEMENT (RESOURCE NUMBERING)

· ALLOWS COOPERATING PROGRAMS A METHOD OF SHARING RESOURCES

CALL RNRQ (ICODE, IRN, ISTAT)

15	14	5	4	3	2	1	0
WAIT OPTION		ALLOCATE OPTION				SET OPTION	
NO W A I T	NO A B O R T	C L E A R	G L O B A L	L O C A L	C L E A R	G L O B A L	L O C A L

ICODE =

IRN = RESOURCE NUMBER. RETURNED ON ALLOCATE;
REQUIRED OTHERWISE.

ISTAT = (RETURNED) = 0 NORMAL DEALLOCATE
1 RN IS CLEAR (UNLOCKED)
2 RN IS LOCKED LOCALLY TO CALLER
3 RN IS LOCKED GLOBALLY
4 NO RN AVAILABLE NOW
5 —
6 RN IS LOCKED LOCALLY TO ANOTHER PROGRAM
7 RN IS LOCKED GLOBALLY

