Figure E8

LOADING OF AN EDITED TASK ON LINE

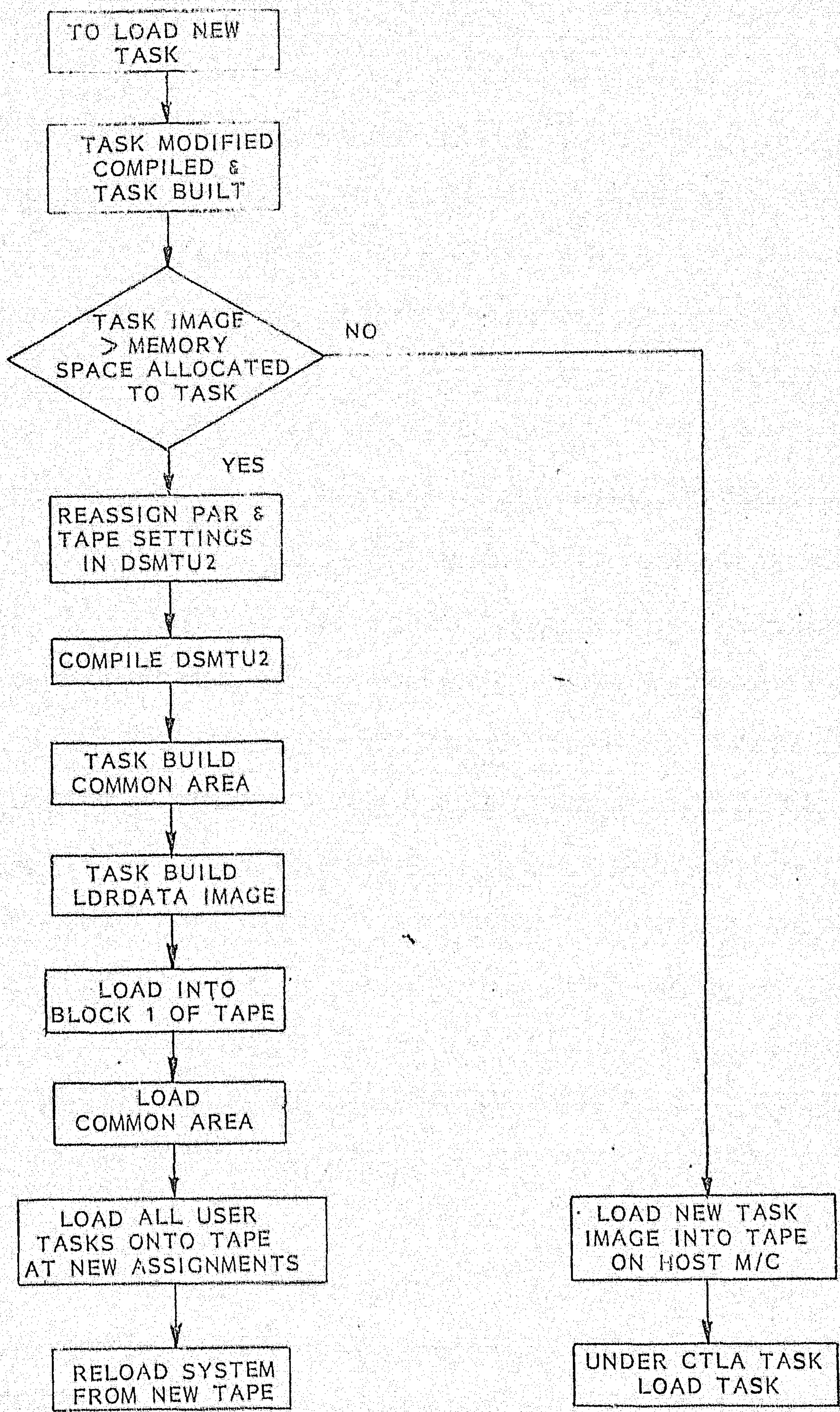
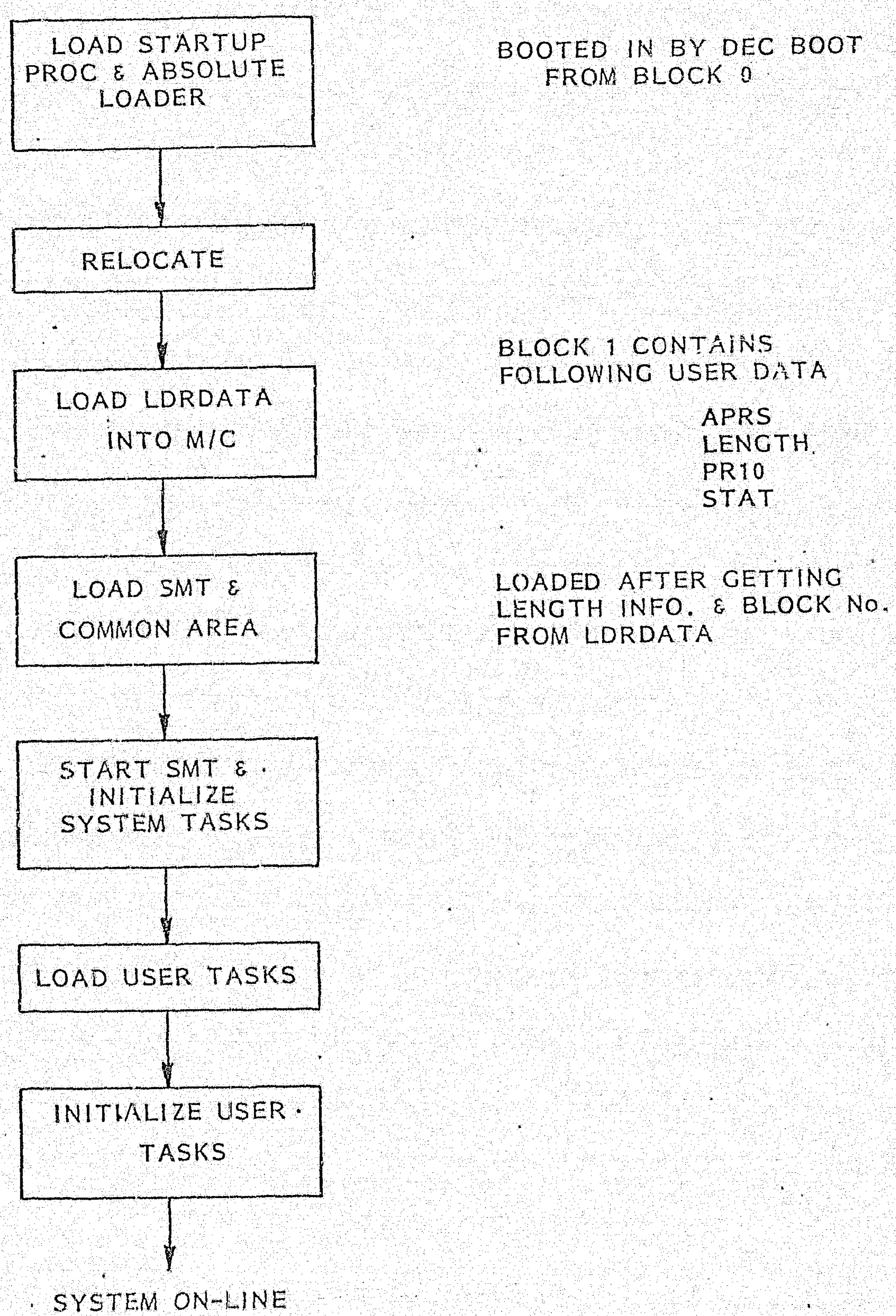


Figure E9

STARTUP PROCEDURE OF NEW SMT



WHAT STARTUP PROGRAM DOES

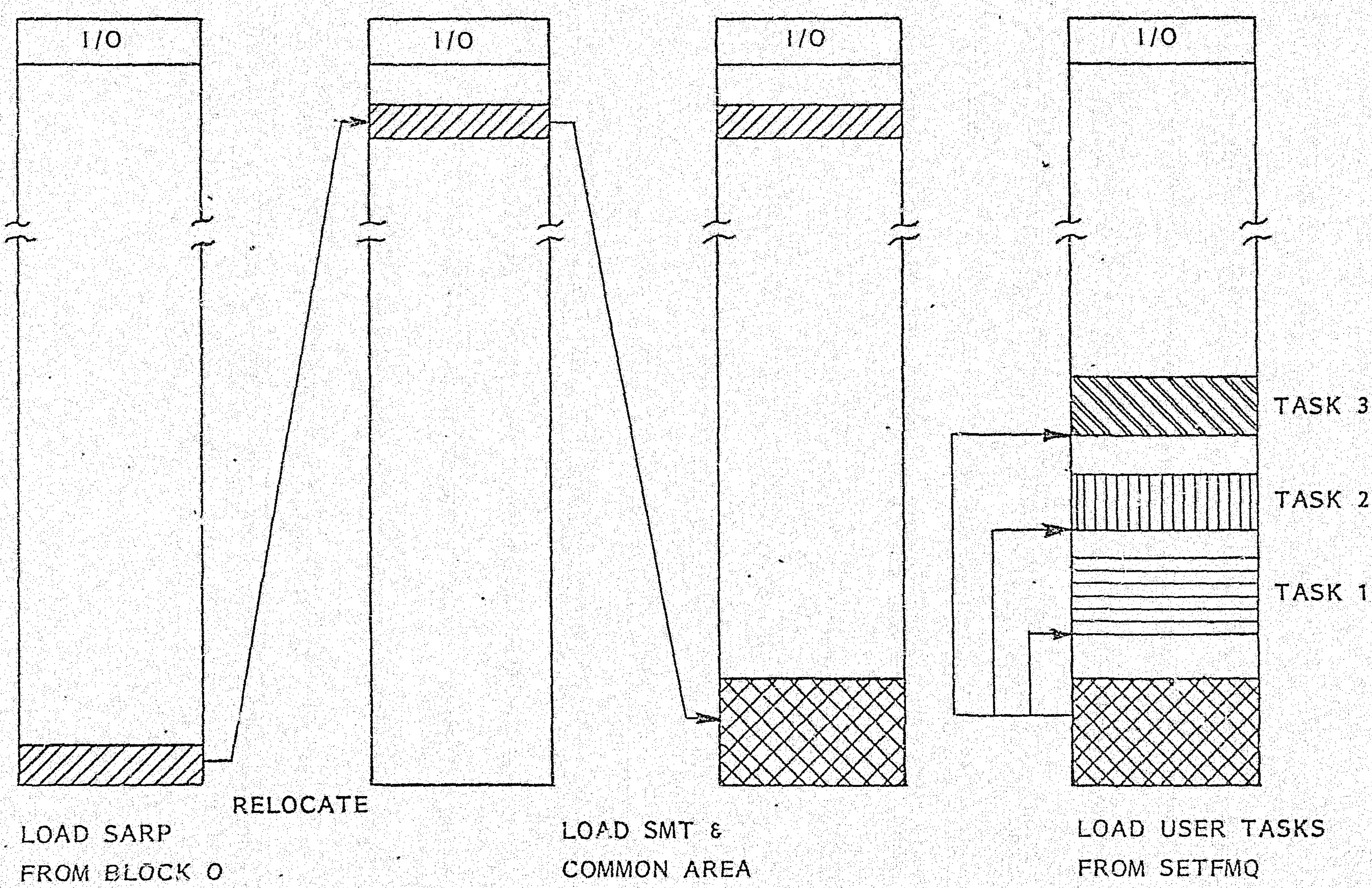


Figure E10

The startup proc extracts these 2 pieces of information and loads in the common area (either 12k or 16k words).

After the common area has been loaded, the machine is told to proceed to execute. At this stage no user tasks have been loaded. The o/s tasks are initialized and the STARTUP task (SETFMQ) loads all the user tasks utilizing the interrupt driven loader within SMT.

As each task is loaded the stack is initialized and the task is ready to start executing once all the user tasks are loaded. As each task is loaded the task byte in the array BADLOAD is reset indicating a sucessful load. If one task was not loaded sucessfully the task STATUS is made STOPPED and the task cannot be started because of the extra check on BADLOAD put in the proc START.

When all user tasks are loaded the flag LOADEDFLAG is made 1 indicating that the tasks have been loaded the first time. If the user wishes to reinitialize the system by doing a OG the tasks are not reloaded.

The system is now fully loaded onto the machine, the user task stacks have been initialized and the o/s arrays have been set up. The system then starts to execute.

APPENDIX 1

EVENTS AND FACILITIES

1.1 SYSTEM S-TASKS

S-TASK NO	USE	BASE PROCEDURE	PRIORITY	STATUS (ON STARTUP)
-5	Fall back task	FBPROC	1	GO
-4	Loader task	TU58	5	GO
-3	Set task	SETFMQ	2	GO
-2	Clock task	CLOCK	250	GO
-1	Error print task	ERPRIN	200	GO
0	Spare system task slot			

1.2 EVENTS

	-15	System device CTLA	- CTLAEV
	-14	System device CTLB	- CTLBEV
	-13	System device keyboard	- KBEV
	-12	System device printer	- PREV
	-11	Secure/release mechanism	- FACEV
	-10	Power fail event	- PWFEV
SYSTEM	- 9		
TASKS	- 8		
	- 7		
	- 6		
	- 5		
	- 4		
	- 3	TU58 Loader	- TU58EV
	- 2		
	- 1		
	0		
	1		
USER	2		
TASKS	3		

1.3 FACILITIES

	-15	System device	- TTYFAC
	-14		
	-13		
	-12		
	-11		
	-10		
	- 9		
	- 8		
SYSTEM	- 7		
TASKS	- 6		
	- 5		
	- 4		
	- 3	TU58 Loader	- TU58FAC
	- 2		
	- 1		
	0		
	1		
USER	2		
TASKS	3		

1.3 FACILITIES

	-15	System device	- TTYFAC
	-14		
	-13		
	-12		
	-11		
	-10		
	- 9		
	- 8		
SYSTEM	- 7		
TASKS	- 6		
	- 5		
	- 4	TU58 Loader	- TU58FAC
	- 3		
	- 2		
	- 1		
	0		
	1		
USER	2		
TASKS	3		

1.3 FACILITIES

	-15	System device	- TTYFAC
	-14		
	-13		
	-12		
	-11		
	-10		
	- 9		
	- 8		
SYSTEM	- 7		
TASKS	- 6		
	- 5		
	- 4	TU58 Loader	- TU58FAC
	- 3		
	- 2		
	- 1		
	0		
	1		
USER	2		
TASKS	3		

APPENDIX 2

UNRECOVERABLE ERROR NUMBERS

Refer to Reference 1 for all the unrecoverable error numbers and their explanation.

Two further additions to the error list should be made namely:

UNREC ERROR NUMBER	REPORTED BY PROC/CONTROL ROUTINE	REFERENCE	CAUSE OF ERROR
36	TRAP 250		Trap generated by mmu protection hardware.
37	START		Attempt to START a task not loaded.

Interpretation of Unrecoverable Error Message

36 The task attempted to access a non existant physical address. A PAR constant used which maps the V.A into a non existant P.A.

37 An attempt was made to START a task that either does not exist, or which did not load successfully. Check stack dump - task number to be STARTED after last link cell. Does the task exist. If so, reload task.

NOTE ON ERROR 33

If an unrecoverable error number 33 is reported by RRGEL, one possible cause is that a modification was made in the common area modules and not all the tasks were recompiled and task built with the new common area image.

APPENDIX 3

RRGEL PRINTOUT EXPLAINED

A printout by RRGEL takes the form as shown in Figure 3.1. This printout will give an exact indication where the fault occured in the task.

With a task map of the common area and the task at fault, as well as the object code listing of the task and the object code listing of all the common area modules can the fault be tracked down quickly.

Points to note the contents of R7 points to the instuction after the instruction causing the error. This address could be either in the task space or the common area.

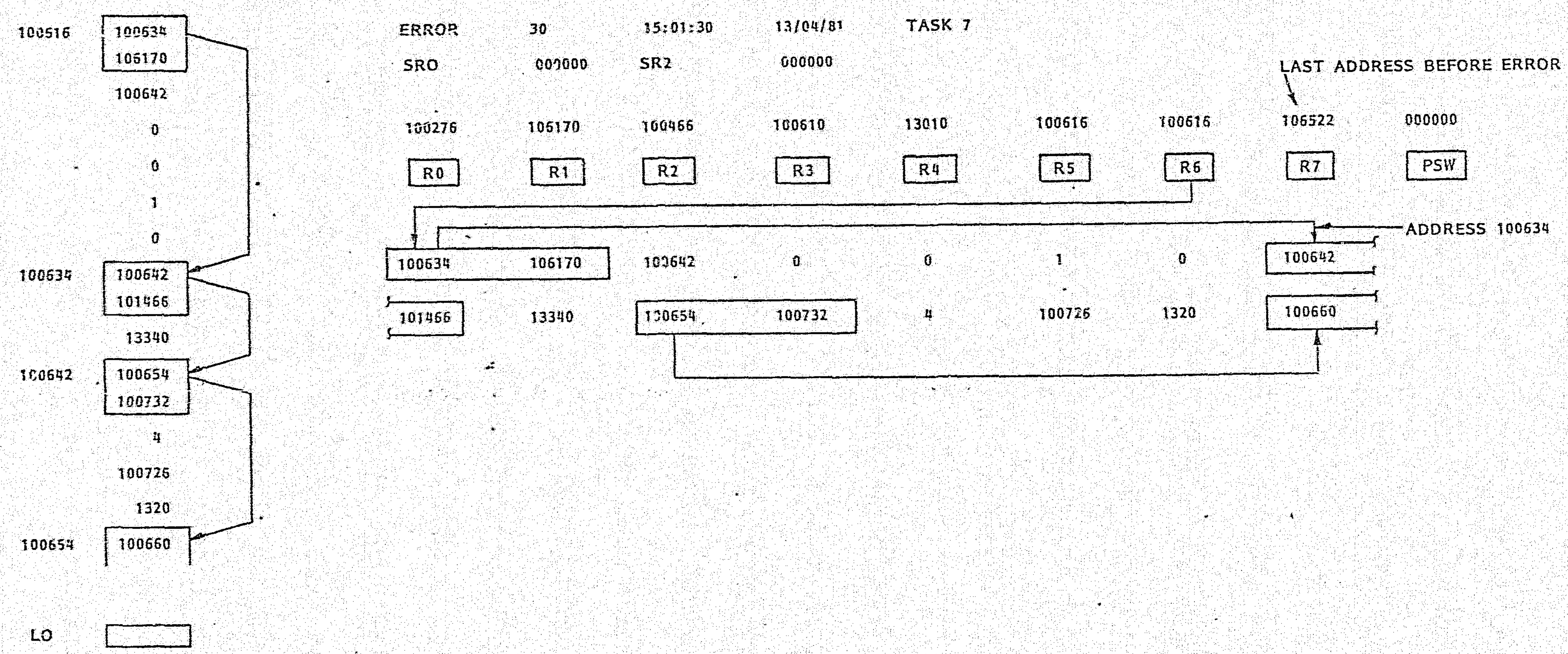
R5 is the address of the last link cell on the stack.

R6 points to the first address of the stack dump.

If R5 = R6 it implies that the first two numbers on the stack dump is a link cell.

Figure 3.1

RRGEL PRINTOUT



APPENDIX 4

PAR AND TAPE LAYOUT CONSTANTS ASSIGNMENT TABLES

When developing an SMT operating under memory management, tape layout constants as well as PAR constants have to be assigned to tasks.

The two tables below will assist in assigning constants for a given task length. The constants given are the number of tape blocks required and the number of PAR blocks required for the task. The absolute values of the constants are cumulative and the tasks should be assigned absolute values when entering the constants in the table in DSMTU2.

An example on how to layout these constants is shown in Section E1.3.

BLOCKS NEEDED ON TAPE FOR GIVEN TASK LENGTH

NO OF BYTES - TASK LENGTH		NO OF BLOCKS NEEDED
DECIMAL	OCTAL	
512	1000	1
1024	2000	2
1536	3000	3
2048	4000	4
2560	5000	5
3072	6000	6
3584	7000	7
4096	10000	8
4608	11000	9
5120	12000	10
7680	17000	15
10240	24000	20
12800	31000	25
15360	36000	30
17920	43000	35
20480	50000	40

RELOCATION CONSTANTS CALCULATION

Relocation constant given as an octal number.

Physical address given as an octal word

Relocation Const	Physical Address
7600	760 000
7400	740 000
7200	720 000
7000	700 000
6600	660 000
6400	640 000
6200	620 000
6000	600 000
5600	560 000
5400	540 000
5200	520 000
5000	500 000
4600	460 000
4400	440 000
4200	420 000
4000	400 000
3600	360 000
3400	340 000
3200	320 000
3000	300 000
2600	260 000
2400	240 000
2200	220 000
2000	200 000
1600	160 000
1400	140 000
1200	120 000
1000	100 000
0600	60 000
0400	40 000
0200	20 000
0000	00 000

					Blocks on tape
EVERY	200	4 096 Words	or	8192 Bytes	16
	100	2 048		4096	8
	70	1 792		3584	7
	60	1 536		3072	6
	50	1 280		2560	5
	40	1 024		2048	4
	30	768		1536	3
	20	512		1024	2
	10	256		512	1
	4	128		256	
	2	64		128	
	1	32		64	

The smallest we can ever go down to is 32 words.

APPENDIX 5

TASK NUMBERING

This appendix briefly describes the numbering system employed within DSMT.

CURTASK — This number is the number which is passed between the user tasks and the operating system. This number lies in the range -5 to 16. The system tasks are negative while the user tasks are numbered from 1 to 16.

CURTEX — This number is used by the operating system to address the system tables. There is no negative CURTEX number. This number lies in the range 1 to 22.

NUSERTASKNO — This number is used by the memory management facilities to access the data arrays used to achieve memory management. This number only exists for user tasks and lies in the range 1 to 16.

See table below - each number has a 1 to 1 correspondence i.e. if CURTASK = 1 it implies CURTEX = 7 and NUSERTASKNO = 1.

TASK NUMBERING

	CURTASK	CURTEX	NUSERTASKNO
USER TASKS	16	22	16
	15	21	15
	14	20	14
	.	.	.
	.	.	.
	.	.	.
	2	8	2
	1	7	1
SYSTEM TASKS	0	6	
	-1	5	
	-2	4	
	-3	3	
	-4	2	
	-5	1	

APPENDIX 5

TASK NUMBERING

This appendix briefly describes the numbering system employed within DSMT.

CURTASK — This number is the number which is passed between the user tasks and the operating system. This number lies in the range -5 to 16. The system tasks are negative while the user tasks are numbered from 1 to 16.

CURTEX — This number is used by the operating system to address the system tables. There is no negative CURTEX number. This number lies in the range 1 to 22.

NUSERTASKNO — This number is used by the memory management facilities to access the data arrays used to achieve memory management. This number only exists for user tasks and lies in the range 1 to 16.

See table below - each number has a 1 to 1 correspondence i.e. if CURTASK = 1 it implies CURTEX = 7 and NUSERTASKNO = 1.

TASK NUMBERING

	CURTASK	CURTEX	NUSERTASKNO
USER TASKS	16	22	16
	15	21	15
	14	20	14
	.	.	.
	.	.	.
	.	.	.
	2	8	2
	1	7	1
	0	6	
SYSTEM TASKS	- 1	5	
	- 2	4	
	- 3	3	
	- 4	2	
	- 5	1	

APPENDIX 6

EXTENDING THE COMMON AREA

If the common area of the system being designed needs to be enlarged from 12k to 16k word a number of changes have to be made to the DSMT modules. These are listed below and the replacement value is given as well.

1. In DSMTB1

AREA OF CHANGE	ACTION	ACTUAL CHANGE
MODE DECLARATION	FROM	(INT PAR3,PAR4,PAR5,PAR6)
	TO	(INT PAR4,PAR5,PAR6)
PROC GETSTEP	FROM	60 000 TO 100 000
	FROM	60 002 TO 100 002
PROC RETFIN STPAR	REMOVE	MOV * CURAPSET 3 ___,@#*PARA3
PROC RETFIN STPAR	FROM	CODE 32,8
	TO	CODE 38,8
PROC RETFIN STPAR	FROM	JSR 2, R22 .WORD 10
	TO	JSR 2, R22 .WORD 6
PROC SETFMQ	FROM	IF CURAPSET ().PAR3 #0
	TO	IF CURAPSET ().PAR4 #0
PROC BEGIN	INCLUDE	MOV #*PARD3,@#*PARA3
PROC BEGIN	FROM	CODE 140,0
	TO .	CODE 146,0
LET DEFINITIONS	INCLUDE	LET PARD3 = 600

2. <u>In DSMTB5</u>

AREA OF CHANGE	ACTION	ACTUAL CHANGE
MODE DECLARATION	FROM	(INT PAR,PAR4,PAR5,PAR6)
	TO	(INT PAR4,PAR5,PAR6)
PROC LOADER	FROM	CURAPSET (LTREQ).PAR3
(MOD APPEARS TWICE)	TO	CURAPSET (LTREQ).PAR4

3. <u>In DSMTU2</u>

AREA OF CHANGE	ACTION	ACTUAL CHANGE
MODE DECLARATION	FROM	(INT PAR3,PAR4,PAR5,PAR6)
	TO	(INT PAR4,PAR5,PAR6)
DATA DUMMY	FROM	ARRAY (272) BYTE DUMMYX:=(0(272))
	TO	ARRAY (304) BYTE DUMMYX:=(0(304))
DATA SMTDATA	FROM	SMTLENGTH := OCT 60 000
	TO	SMTLENGTH := OCT 100 000

4. <u>In SMTTKB</u> (command file)

AREA OF CHANGE	ACTION	ACTUAL CHANGE
	FROM	.DATA PAR ='SMTASK':60 000:100 000
	TO	.DATA PAR ='SMTASK':100 000:60 000