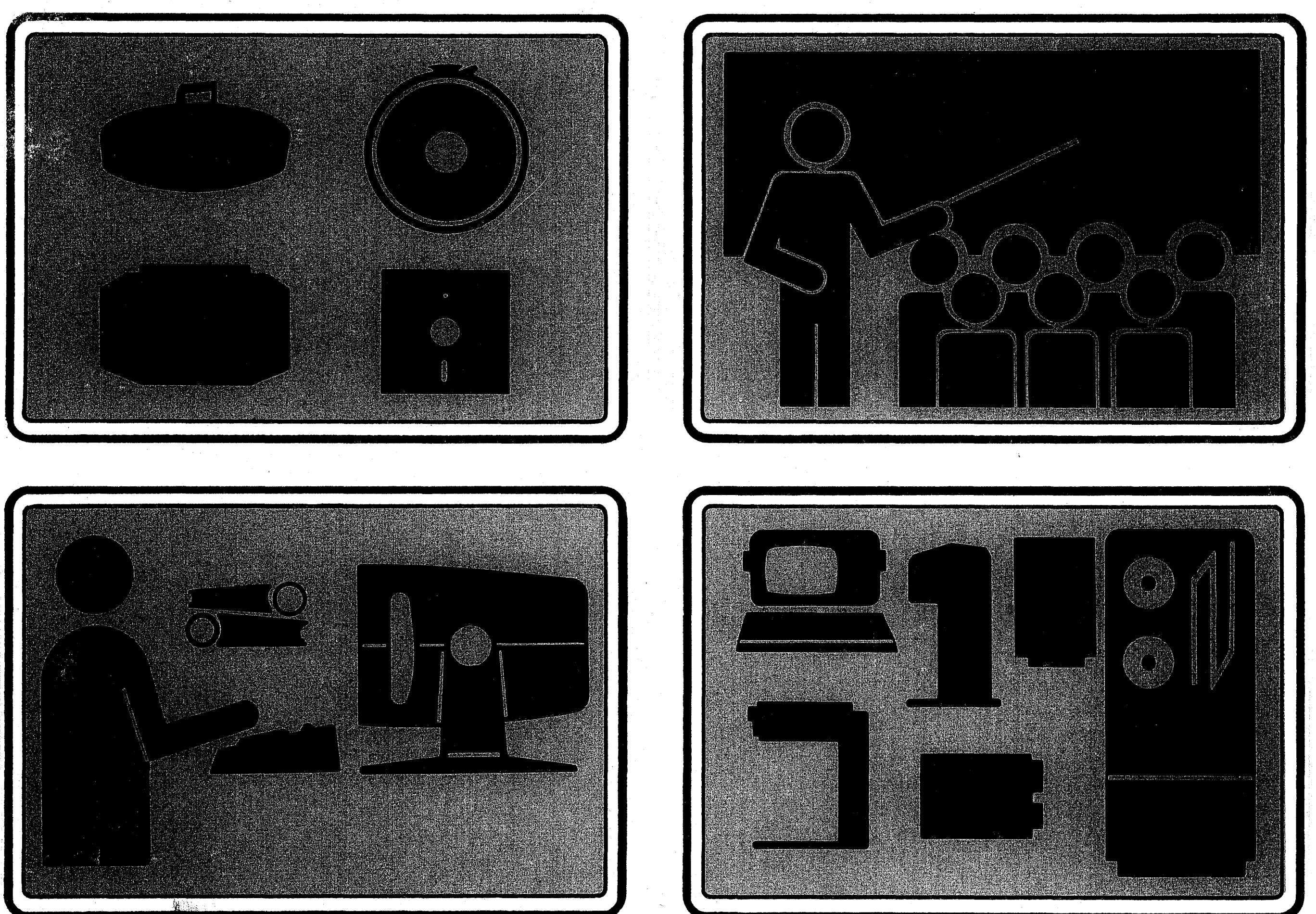
Data General

Educational Services

S200
RDOS USER

STUDENT HANDOUT



019-000048-04

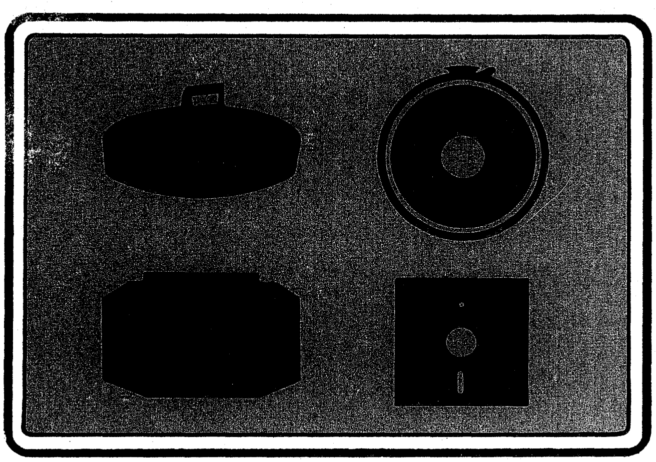
NOTICE

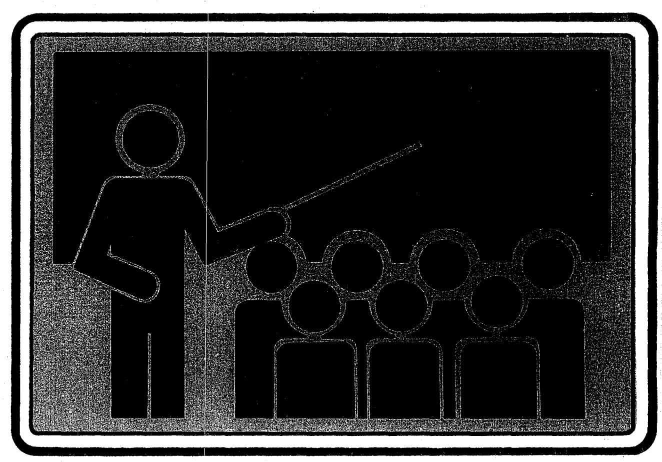
Data General

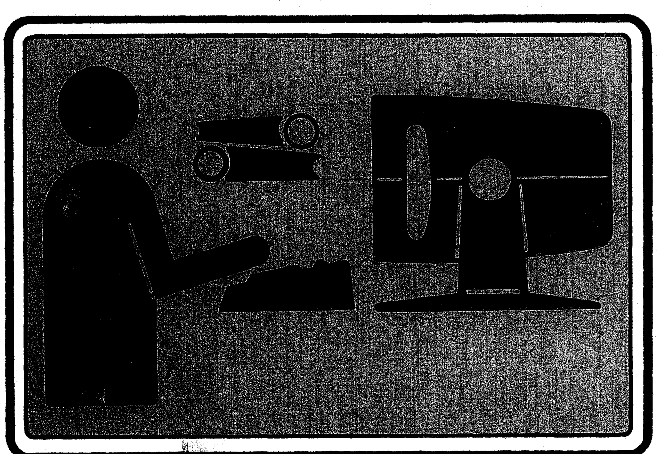
Educational Services

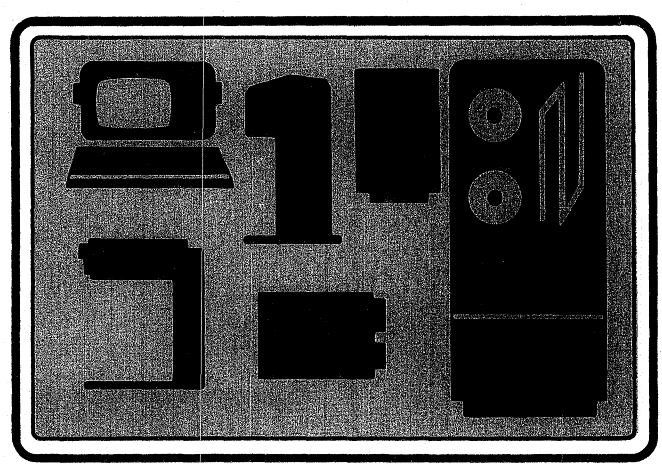
S200
RDOS USER

STUDENT HANDOUT









019-000048-04

S200

RDOS USER

COURSE OUTLINE

I. ARCHITECTURAL OVERVIEW/FRONT PANEL OPERATION

A. MEMORY

1. KINDS
2. UNITS
3. SIZES OF PHYSICAL MEMORY
4. LOGICAL SUBDIVISIONS

B. CENTRAL PROCESSING UNIT (CPU)

1. REGISTERS AND FLIP FLOPS

- PC
- IR
- DECODE LOGIC
- ALC UNIT
- ACCUMULATORS
- CARRY
- ION
- MAP

2. OVERVIEW OF CAPABILITIES

- ARITHMETIC
- CHANGE PC
- DEVICE I/O
- MEMORY I/O

C. PERIPHERALS

1. CONTROLLERS

- I/O BUSS
- DEVICE CODES
- REGISTERS
- BUSY/DONE

2. DEVICES

- RUDIMENTARY DEVICES ($TTI $TTO $LPT DP0)

D. FRONT PANEL OPERATION

1. ADDRESS/DATA LIGHTS
2. DATA SWITCHES
3. OFF/ON/LOCK
4. START/CONTINUE
5. RESET/STOP
6. EXAMINE/EXAMINE NEXT
7. DEPOSIT/DEPOSIT NEXT
8. EXAMINE/DEPOSIT ACCUMULATORS
9. PROGRAM LOAD
10. FUNCTION LIGHTS: (ROM FETCH INDIRECT MAP ION . . .)

II. BOOTSTRAPPING, STARTUP, SHUTDOWN *

A. ORIENTATION & PROGRAMS

1. INSTALLATION OF SUCCESSIVELY MORE POWERFUL PROGRAMS
2. PGM LD/BOOT.SV (HIPBOOT)

B. POSITIONING TO THE BOOTSTRAP PROGRAM

1. DISK: POWER & LINE SWITCHES, TOGGLING
2. TAPE: LOAD SWITCH OR RESET REWIND ON-LINE

C. HOW TO PERFORM THE BOOTSTRAP

1. DISK
2. TAPE

D. STARTUP FROM DISK

1. FILENAME: "ENTER SYSTEM NAME"
2. DATE: "ENTER CURRENT DATE"
3. TIME: "ENTER CURRENT TIME"
4. PROGRAMS SOUGHT ON DISK: CLI. <SV,ER,OL>, BOOT.SV

E. SHUTDOWN FROM DISK

1. HALT VARIOUS SYSTEM PROCESSES: SPOOLING, FOREGROUND, LOG, . . .
2. GET ACCESS TO THE COMMAND LINE INTERPRETER
3. RELEASE MASTER DIRECTORY

III. INTRODUCTION TO OPERATING SYSTEMS — RDOS *

A. THE NEED FOR OPERATING SYSTEMS: A HISTORIC OVERVIEW

1. ORIGINAL PROGRAM DEVELOPMENT
2. THE FIRST UTILITIES - - EDITOR & ASSEMBLER
3. THE SIMPLE MONITOR SYSTEM
4. MODERN OPERATING SYSTEM TECHNIQUES FOR EFFICIENCY

B. ELEMENTS OF MODERN OPERATING SYSTEMS

1. PRIMARY GOAL: HELP USER MANAGE RESOURCES
2. I/O AND DEVICE MANAGEMENT
3. FILE MANAGEMENT
4. MEMORY MANAGEMENT
5. PROCESS MANAGEMENT

C. RDOS ANALOGUES TO THE MODEL OPERATING SYSTEM

1. PROCESS
2. MEMORY
3. FILE AND I/O MANAGEMENT

IV. INTRODUCTION TO THE COMMAND LINE INTERPRETER (CLI) *

A. DEFINITION OF CLI

B. THE CLI COMMAND STRUCTURE

1. COMMANDS AND ARGUMENTS
2. GLOBAL & LOCAL SWITCHES
3. CLI PUNCTUATION
4. EXPANDERS: IN-LINE, MULTI-LINE
5. RDOS FILE NAME TEMPLATES
6. SPECIAL SYMBOLS

C. CLI PERCENT VARIABLES

D. INDIRECT FILES

E. MACRO FILES

**************************** CLI VOCABULARY ****************************

BOOT	TYPE	GSYS	STOD	LOG	ENDLOG	PUNCH	GTOD
LIST	REV	MESSAGE	PRINT	XFER	SDAY		

****

LAB EXERCISE: BOOTSTRAPPING & CLI

V. DISK BASICS/RDOS & INFOS FILE STRUCTURES *

A. DISK BASICS

1. A PHYSICAL DISK BLOCK

- SECTOR X SURFACE X TRACK (CYLINDER)
- SYNC BITS, ADDRESS, DATA, CYCLIC CHECK SUM
- DTOS/DDOS WRITES FORMATTING INFO

2. PRELIMINARY DISK BLOCKS

• HIPBOOT (BOOT.SV)	BLOCK 0 & 1
• PHYSICAL DISK MANAGEMENT INFORMATION	BLOCK 3
• REMAP AREA (BAD DISK BLOCKS)	BLOCK 4
• SOME UNUSED INITIAL BLOCKS	BLOCK 2 & 5
• RDOS REFERS TO THESE PHYSICALLY	

B. RDOS FILE STRUCTURES

1. SEQUENTIAL

- 255 DATA WORDS/BLOCK AND A LINK ACCESS WORD
- EXPANDABLE
- MEDIUM OVERHEAD, SLOWEST ACCESS
- SEQUENTIAL ACCESS, NO DMA

2. CONTIGUOUS

- 256 DATA WORDS/BLOCK AND GARENTEED PROXIMITY
- NONEXPANDABLE, FIXED ALLOCATION (EOF ON WRITE)
- MINIMUM OVERHEAD, FASTEST ACCESS
- RANDOM ACCESS, DIRECT MEMORY ACCESS (DMA)
- BEST FOR VERY LARGE FILES

3. RANDOM

- 256 WORDS/DATA BLOCK AND A FILE INDEX BLOCK
- EXPANDABLE
- MAXIMUM OVERHEAD, MEDIUM SPEED
- RANDOM ACCESS, DMA

C. INFOS FILE STRUCTURES OVERVIEW

1. SAM:
   - SEQUENTIAL ACCESS METHOD
   - CONTIGUOUS ALLOCATION & RANDOM OVERFLOW

2. RAM:
- RELATIVE ACCESS METHOD
- ACCESS VIA RELATIVE RECORD NUMBER

3. ISAM:
- INDEXED SEQUENTIAL ACCESS METHOD
- SINGLE DEYED ACCESS VIA INDEX FILE (.1X)
- DATA.IX → (DATA.VL INDEX.VL) ⇒ (DATA INDEX)
- MULTIPLE DATA & INDEX FILES

4. DBAM:
- DATA BASED ACCESS METHOD
- MULTI KEYED ACCESS
- MULTI LEVEL KEYED ACCESS
- DATA.IX → (DATA.VL INDEXN.VL) ⇒ (DATA INDEXN)

VI. RDOS DIRECTORY STRUCTURE

A. PRIMARY PARTITION : CONTROLS ENTIRE DISK PLATTER

- PRIMARY PARTITION SYSTEM DIRECTORY FILE : SYS.DR
- SYS.DR – A FILE INDEX BLOCK
- SYSTEM DATA BLOCKS – DATA ENTRY BLOCKS
- MAP.DR – BIT MAP DISK BLOCK ALLOCATION CONTROL

1. DATA ENTRY BLOCKS

- CONTENTS : USER FILE DESCRIPTIONS (UFD'S, 14 MAX)
- CURRENT UFD'S IN DATA ENTRY BLOCK (FIRST WORD IN BLOCK)
- TOTAL UFD'S IN DATA ENTRY BLOCK (NEXT TO LAST WORD)

2. USER FILE DESCRIPTIONS (UFD)

- FILENAME & EXTENSION
- ATTRIBUTES & CHARACTERISTICS
- LINK ATTRIBUTES & CHARACTERISTICS
- 512 * RELATIVE BLOCKS + LAST BLOCK BYTES = TOTAL BYTES
- WORD 12 : POINTER TO DATA BLOCKS
- DATE & TIME CREATED
- TIME LAST ACCESSED
- USE COUNT
- DCT LINK

3. FILENAME RESOLUTION

- FRAME SIZE (FS) & HASH VALUE OFFSET (HVD)
- FILENAME SEARCH WITHIN DATA ENTRY BLOCK
- NOT FOUND, TOTAL = 14; HVO = HVO + FS, SEARCH AGAIN
- NOT FOUND, TOTAL < 14; FILE NOT FOUND ERROR

4. FILENAME DELETION

- HASH & SEARCH DATA ENTRY BLOCKS
- DECREMENT CURRENT UFD'S COUNT
- NULL FIRST TWO CHARACTERS IN FILENAME
- ZERO BITS ALLOCATED IN MAP.DR

B. SECONDARY PARTITION STRUCTURE

- USER NAMED UFD IN PRIMARY → SYS.DR FOR SECONDARY
- SYS.DR : FILE INDEX BLOCK FOR SECONDARY PARTITION DATA ENTRY BLOCKS
- DATA ENTRY BLOCKS HOLD SECONDARY PARTITION'S UFD'S
- MAP.DR : ALLOCATION CONTROLL (INTERNALLY & EXTERNALLY)

C. SUBDIRECTORY STRUCTURE

- USER NAMED UFD IN CURRENT DIRECTORY → SYS.DR FOR DIRECTORY
- SYS.DR : FILE INDEX BLOCK FOR DIRECTORY'S DATA ENTRY BLOCKS
- MAP.DR – POINTS TO PARENT MAP.DR FOR SHARED ACCESS

D. REFERENCES WITHIN/BETWEEN PARTITIONS & DIRECTORIES

- DCB'S – INITIALIZATION & RELEASE
- LINKS – REFERENCES ACROSS DIRECTORY/PARTITION BOUNDARIES
- LINK UFD'S, LINK ATTRIBUTES & CHARACTERISTICS

VII. ALTERING THE INFORMATION ON DISK – DSKED.SV

A. DSKED.SV – STAND ALONE DISK EDITOR (BOOT'ED)

1. ADDRESS SPECIFICATION (BLOCK:OFFSET/CONTENTS)
2. HASHING (FRAMESIZE; FILENAME=)
3. HALT DSKED, UPDATE DISK (ESC Z)
4. LOCAL COMMANDS:

RIGHTSLASH	ASTERISK	LINE FEED	LEFT ARROW
APOSTROPHY	EQUAL SIGN	UP ARROW	CARRIAGE RETURN

******************************** VOCABULARY ******************************

BUILD	CRAND	RENAME	GDIR	CCONT	CREATE	FPRINT	INIT	CHATR	FILCOM
CPART	LDIR	CHLAT	DIR	MDIR	CLEAR	LINK	DISK		

****

LAB : MORE CLI AND DSKED LAB

VIII. PROGRAM DEVELOPMENT *

A. SOURCE CREATION : EDIT.SV

1. OPERATING PRINCIPLES
2. EDIT COMMANDS

- FILE ASSOCIATION
- INPUT / OUTPUT
- DELETE
- MACRO IMPLEMENTATION
- CP POSITIONING
- SEARCH
- DISPLAY

3. COMMON PROBLEMS

B. COMPILATION

1. LANGUAGE TRANSLATOR (HIGH LEVEL → ASSEMBLY)
2. EXTERNAL REFERENCES & RUNTIME SUPPORT LIBRARIES
3. SYNTACTICAL ERRORS – LISTING FILES

C. ASSEMBLY

1. SYMBOLOGY TRANSLATOR (ASSEMBLY → BINARY)
2. PASS 1 : TRANSLATION & SYMBOL TABLE CONSTRUCTION
3. PASS 2 : INTERNAL RESOLUTION – FORWARD REFERENCES
4. THE RELOCATABLE BINARY FILE (.RB)

D. RELOCATABLE LOADING

1. RELOCATABLE BINARY (.RB) → CORE IMAGE SAVE FILE (.SV)
2. CODE PLACEMENT IN .ZREL AND .NREL
3. ENTRY POINTS LOGICALLY CONNECTED TO EXTERNAL REFERENCES
4. UNRESOLVED EXTERNALS AND THE LOAD MAP

E. EXECUTION & TEST

1. LOGICAL ERRORS – LOCATION WITHIN LOAD MAP
2. DEBUGGER OVERVIEW

IX. OTHER RDOS EDITORS

C SEDIT / OEDIT / MEDIT / SPEED / LFE OVERVIEWS

1. SEDIT – SINGLE USER, SINGLE LOCATION SYMBOLIC EDITOR
2. OEDIT – SINGLE USER, SINGLE LOCATION EDITOR
3. MEDIT – MULTIUSER, TEXT EDITOR (ESSENTIALLY EDIT)
4. SPEED – SINGLE USER, MULTIBUFFER SUPER TEXT EDITOR
5. LFE – SINGLE USER, SINGLE SCAN, LIBRARY FILE EDITOR

X. PROGRAMMING TECHNIQUES TO MANAGE MEMORY

1. CHAIN EXECUTION OF PROGRAMS

   - MANIPULATION OF ENTIRE PROGRAMS
   - DESTRUCTIVE MEMORY LOADS
   - EXECUTION AT SINGLE PROGRAM LEVEL

2. SWAP EXECUTION OF PROGRAMS

   - SUBORDINATE EXECUTION OF ENTIRE PROGRAMS
   - NONDESTRUCTIVE MEMORY LOADS
   - PROGRAM LEVELS & SWAP FILE INDEX BLOCKS

3. OVERLAY

   - PORTION OF USER SPACE LOADED WITH CODE FROM DISK
   - ROOT CODE & OVERLAY AREAS WITHIN .SV
   - SEGMENTS & OVERLAYS WITHIN .OL
   - SIZES & ASSOCIATIONS (SEGMENT VS. OVLYAREA)
   - RLDR CONFIGURATIONS

******************************** VOCABULARY ********************************

PROGNAME	NSPEED.SV	MAC.<SV.PS>	SEDIT.SV
ASM.SV	SPEED.<SV,ER>	CLG.SV	OVLDR.SV
XREN.SV	ALGOL.SV	LFE.SV	FORT.SV
EDIT.SV	RLDR.<SV,OL>	FIV.SV	MEDIT.RB
OEDIT.SV	CHAIN	POP	

****

LAB : TEXT EDIT, PROGRAM DEVELOPMENT, SEDIT

XI. SYSTEM INSTALLATION ON A FORMATTED DISK *

A. THE RDOS STARTER TAPE [SYSGEN MANUAL 3–1]

1.	MT 0:0	TBOOT.SV	XFER FORMAT
2.	MT0:1	CLI.<SV,ER,OL> BOOT.SV, BOOTSYS.SV	DUMP
3.	MT0:2	BOOTSYS.SV	XFER
4.	MT0:3	BOOTSYS.OL	DUMP/A
5.	MT0:4	DKINIT.SV	XFER
6.	MT0:5	BOOT.SV	XFER
7.	MT0:6	RDOS UTILITIES	DUMP
8.	MT0:7	RDOS LIBRARIES	DUMP

B. DISK INITIALIZER [SYSGEN CH 9]

1. FUNCTIONS

- DISK TYPE, FRAME SIZE, BAD BLOCK TABLE SIZE
- TEST FOR BAD BLOCKS, BUILD REMAP AREA

2. COMMANDS

- FULL – FULLY INITIALIZES THE DISK
- PARTIAL – INSPECTS FOR BAD BLOCKS
- ENTER – UPDATE OLD REMAP AREA WITH NEW BAD BLOCKS
- LIST – DISPLAY DISK STATUS, REMAP & FRAME SIZES, BAD BLOCKS
- STOP – HALTS DKINIT, REHOMES DISK HEADS

C. INSTALLATION OF RDOS SOFTWARE [SYSGEN CH 3]

1. INITIALIZE THE DISK
2. INSTALL HIPBOOT (BOOT.SV)
3. INSTALL STARTER SYSTEM & ASSOCIATED SOFTWARE
4. THE UTILITIES DIRECTORY
5. THE SYSGEN DIRECTORY (ALSO: EDIT, SYSGEN, RLDR.<SV,OL>)

XII. SYSTEM GENERATION * [SYSGEN CH 5]

A. PROGRAMS AND FILES

1.	SYSGEN.SV, RLDR.<SV,OL>	THOSE EXECUTING
2.	CLI.CM, SYS000.RB	THE TEMPORARIES
3.	*RDOS<A,B,C,I,O> .LB	THOSE REFERENCED FOR CODE
4.	SYSNAME.<SV,OL,MP,SG>	THOSE CREATED

B. THE PROCESS/MECHANICS OF SYSTEM GENERATION

1. SYSGEN INVOKATION : *SYSGEN SYSNAME.</S SG/V MP/L>
2. CURRENT DIALOGUE QUESTIONS – ANSWERS
3. SYS000.RB LOADING WITH *RDOS<A,B,C,I.O>.LB
4. GENERATION OF SYSNAME.<SV,OL,MP,SG>

C. SYSTEM COMPONENTS, THEIR FUNCTIONS & SIZES

1. STACKS
   – TOTAL CONCURRENT SYSTEM PROCESSIES
   – 310 OCTAL WORDS (OW), WAIT STATE WHEN INSUFFICIENT

2. CELLS
   – TOTAL CONCURRENT SYSTEM CALLS (FG & BG)
   – 20 OW, TCB WAIT STATE WHEN INSUFFICIENT

3. BUFFERS
   – SYSTEMS CAPACITY TO HOLD DATA IN CORE
   – 416 OW, SLOWER OVERALL MORE DISK ACCESSES

4. UFT
   – SYSTEM CAPACITY FOR DISTINCT I/O TRANSPORT
   – 50 OW/CHANNEL, ERROR REJECT WHEN INSUFFICIENT

5. DCB
   – TOTAL CONCURRENTLY ACCESSABLE DISK DIRECTORIES
   – 416 OW, ERROR REJECT WHEN INSUFFICIENT

6. OTHER CORE RESIDENT COMPONENTS

   - SCHEDULER
   - SYSTEM CALL PROCESSOR
   - DRIVERS & SERVICE ROUTINES
   - INTERRUPT HANDLING
   - 77 OVERLAYS (400 OW /)

XIII. SYSTEM UPDATES / PATCH FACILITIES

A. THE STANDARD UPDATE TAPE

1. MAJOR / MINOR REVISION NUMBERS & UPDATE NUMBER (RDOS 19.84)
2. UPDATE FILE, PATCH FILES, PATCH MACROS.

B. ENPAT UTILITY

1. PATCH = A ONE WORD CHANGE TO A .SV OR .OL FILE
2. ENPAT ALLOWS CODING OF PATCH DATA (CONDITIONALLY/ UNCONDITIONALLY)

C. PATCH UTILITY

1. PATCH INSTALLS PATCH DATA CREATED VIA ENPAT
2. COMMAND STRUCTURE :

- PATCH SAVEFILENAME/S PATCHFILE.PF/P LOADMAP/L

3. GLOBALS: /I SUPRESS COMMENTS
   /N NO ACCOMPANYING LOAD MAP FILE

XIV. MONITORING AN RDOS OPERATING SYSTEM

A. ASPECTS OF TUNING

1. REQUESTED DURING SYSGEN
2. RESOURCE ALLOCATION RECORDS VS. ACCEPTABLE RESULTS
3. CLI MECHANICS

- TUON SYSNAME
- TUOFF
- TPRINT/L/O SYSNAME

XV. SYSTEM BACKUP : STARTER TAPE EMULATION

A. MECHANICS OF TRANSFERS (TAPE & DISK→DISK)

1. XFER
   - FILE CONTENTS ONLY TRANSPORTED
   - ONE DISK FILE/COMMAND: SOURCE DESTINATION
   - CONTENTS BOOT'ABLE FROM TAPE

2. DUMP/LOAD
   - UFD & CONTENTS TRANSFERRED TO TAPE
   - MANY DISK FILES/MAG TAPE FILE
   - DIRECTORY STRUCTURE MAINTAINED
   - NOT BOOT'ABLE

3. FDUMP/FLOAD
   - THREE MAG TAPE FILES/COMMAND
   - ALL FILES IN CURRENT DIRECTORY TRANSPORTED
   - FASTEST & MOST CONDENSED
   - NEW TAPE VOLUME CONTROLS

4. MOVE
   - DIRECTORY TO DIRECTORY TRANSPORT
   - UFD & FILE CONTENTS TRANSFERRED
   - DIRECTORY SPECIFIER OR FILENAME TEMPLATES
   - GLOBALS : /A /D /K /L /R /V
   - LOCALS : MM–DD–YY/A, MM–DD–YY/B, NAME/N

B. A BACKUP TAPE MACRO

1. MESSAGE ANNOUNCEMENT – BACKUP IN PROGRESS
2. EMULATION OF THE RDOS STARTER TAPE (FILES 0 – 5)
3. DUMP ALL SOFTWARE TO MT0: (6,7) [BELTS & SUSPENDERS]
4. OR FDUMP ALL SOFTWARE TO MT0: (0,3) [ON A NEW TAPE]
5. TERMINATION MESSAGE

******************************* VOCABULARY *******************************

DUMP	FDUMP	MOVE	TUOFF	ENPAT	EQUIV	LOAD	FLOAD	TUON
TPRINT	PATCH	SYSGEN						

****

LAB: SYSTEM BACKUP MACRO & SYSTEM INSTALLATION

XVI. RDOS SPOOLING

A. THE OUTPUT PROCESS

1. USER DATA BUFFER OR SOURCE
2. RDOS DATA BUFFER
3. DEVICE DATA BUFFER
4. SPOOL FILES / OPTIMUM CPU TIMING
5. INTERRUPT DRIVEN DATA REQUESTS

B. CLI CONTROL COMMANDS & SPOOL'ABLE DEVICES

- SPDIS DEVICENAME
- SPKILL DEVICENAME
- SPEBL DEVICENAME

$DP0 $LPT(1) $PTP(1) $TTO(1)
$TTP(1)

C. SPOOL FILE LOSS & RECOVERY

1. UNUSED DISK ALLOCATION
2. UNDELETABLE WITHOUT FILENAMES
3. FULL INITIALIZATION / BACKUP RECOVERY
4. SECONDARY PARTITION BOOTSTRAP REMEDY

XVII. RDOS PROCESS MANAGEMENT : FOREGROUND / BACKGROUND

A. MAP UNIT ADDRESS TRANSLATION

1. 5 + 10 BIT PROGRAM COUNTER
2. 32 SLOT TRANSLATION TABLE CONTAINING 7 BIT PHYSICAL ADDRESSES
3. 7 + 10 BIT TO ACCESS PHYSICAL MEMORY
4. 10 BIT OFFSET INTO PHYSICAL PAGE (1K)
5. EACH PAGE PROTECTED: VALIDITY, READ, WRITE, I/O, DATA CHANNEL

B. RDOS TRANSLATION TABLE MANIPULATION

1. PROGRAM TABLES LOAD MAPA, MAPB, DATA CHANNEL MAP
2. FOREGROUND: ACTIVATE MAPA
3. BACKGROUND: ACTIVATE MAPB
4. EVENTUAL 1K PAGE SHUFFLE VS. LOGICALLY SEQUENTIAL PAGES

C. RDOS DUAL PROCESSES : FOREGROUND / BACKGROUND

1. CORE CONFIGURATION (MAPPED / UNMAPPED)

- USER SPACE / RDOS / PAGE ZERO
- USER STATUS TABLE & TASK CONTROL BLOCKS
- UNMAPPED LOADING CONSIDERATIONS (/Z /F)

2. PROCESS PRIORITIES

- EQUAL – ROUND ROBIN
- DEFAULT – FOREGROUND HIGHER (REAL-TIME APPLICATIONS)

3. CLI CONTROL

- EXFG/E
- GMEM
- FGND
- SMEM
- CNTRL F

XVIII. RDOS EXTENDED MEMORY : VIRTUAL TECHNIQUES

A. DEFINITION OF EXTENDED MEMORY / HYPERSPACE

1. MEMORY IN GROUND BEYOND USER ADDRESS SPACE
2. HYPERSPACE MAY HAVE: DATA – WINDOW MAPPING CODE – VIRTUAL OVERLAYS

B. WINDOW MAPPING & VIRTUAL OVERLAYS

1. WINDOW DEFINITION IN USER SPACE (BOTH)
2. EXTENDED MEMORY HOLDS CODE OR DATA
3. REMAP – PHYSICAL PAGE ADDRESS TRADE IN TRANSLATION TABLE

XIX. RDOS MALFUNCTIONS & RECOVERY

A. TRAP – A MAP VIOLATION

1. $TTO(1) ⇒ ACCUMULATORS, PROGRAM COUNTER
2. PROGRAM COUNTER = LOCATION OF INSTRUCTION IN ERROR
3. (F)BREAK.SV CREATED

B. PANIC – EXCEPTIONAL SYSTEM DATA [RDOS REF MAN.APPENDIX G]

1. $TTO ⇒ ACCUMULATORS, PANIC CODE

******************************* VOCABULARY *******************************

SPDIS SPKILL GMEM EXFG (F)BREAK.SV PANIC TRAP SPEBL
FGND SMEM SAVE

****

LAB : SYSGEN, TUNNING FILE REPORT, CARRY OVER

S200

RDOS USER

MODULE 1

ARCHITECTURAL OVERVIEW/FRONT PANEL OPERATION

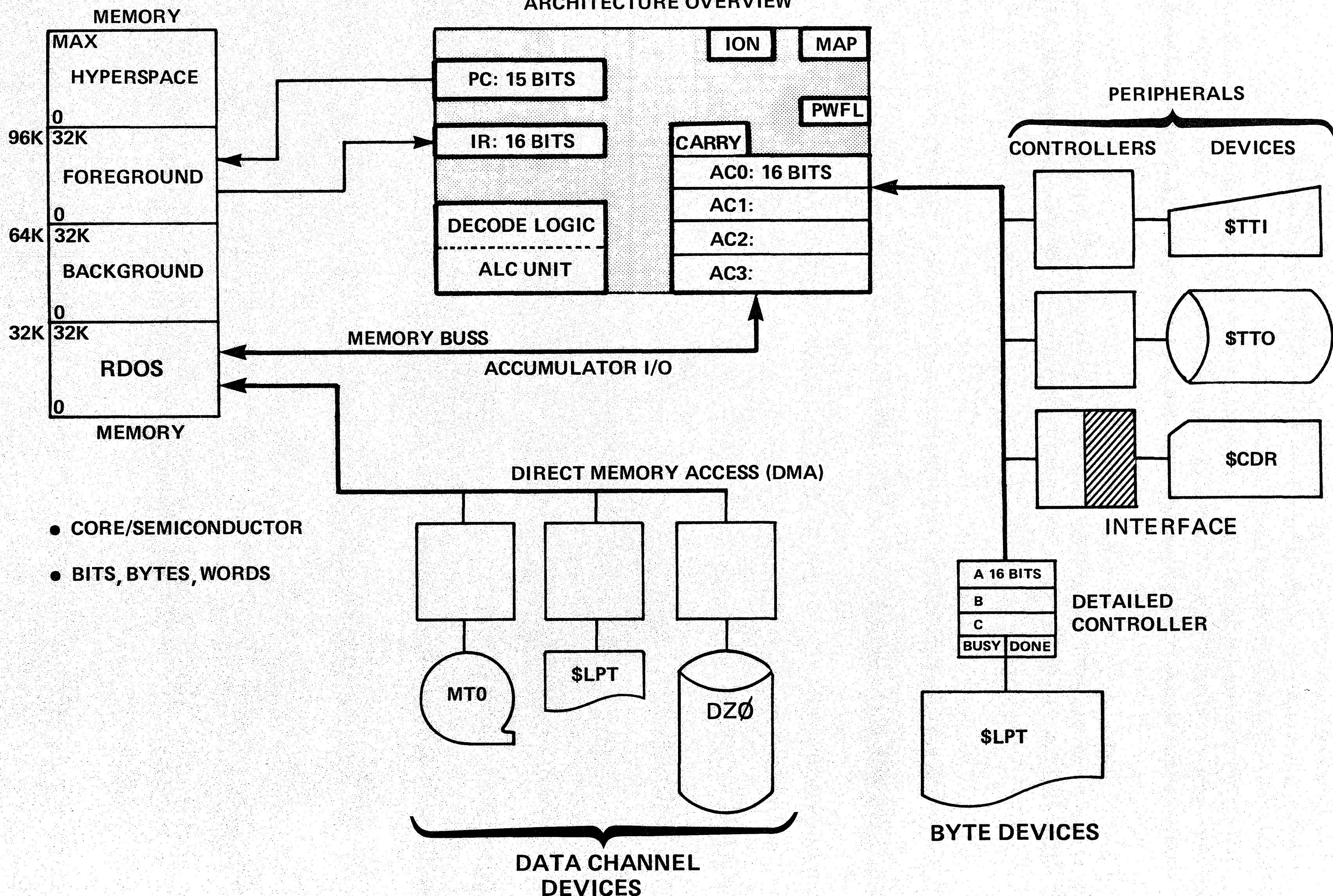
Module 1

OBJECTIVES

Upon successful completion of this module you will be able to:

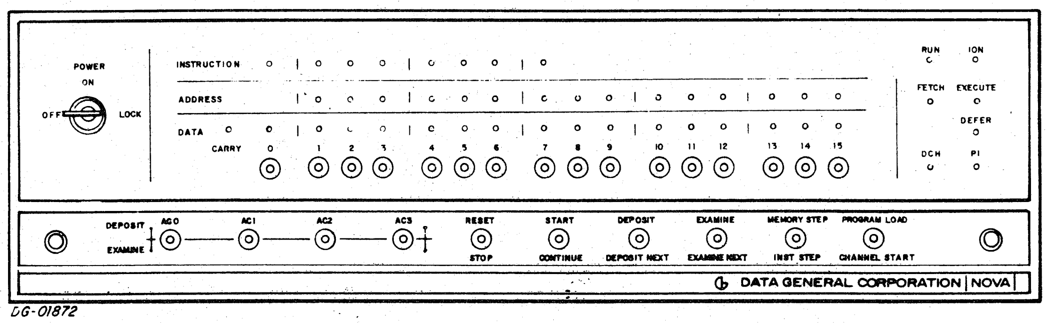
* DEFINE THE UNITS OF INFORMATION WITHIN MEMORY
* DEFINE THE PHYSICAL SIZES OF MEMORY WHICH RDOS CAN MANAGE
* DEFINE THE LOGICAL SUBDIVISIONS WHERE RDOS RESIDES, WHERE THE FOREGROUND/BACKGROUND PROGRAMS RESIDE
* DESCRIBE THE INTERNAL REGISTERS IN THE CENTRAL PROCESSING UNIT WHICH ARE CONTROLLED VIA THE FRONT PANEL
* ENUMERATE THE CENTRAL PROCESSING UNIT CAPABILITIES WHICH RDOS BOTH RELIES UPON AS A PROGRAM AND MANAGES AS AN OPERATING SYSTEM
* EXPLAIN THE SIZE OF USER ADDRESS SPACE USING CENTRAL PROCESSING UNIT REGISTERS
* EXPLAIN RDOS'S REAL TIME DEVICE CONTROL VIA DEVICE CODES, INTERRUPTS, THE I/O BUSS, AND DEVICE CONTROLLER BOARDS
* DISTINGUISH BETWEEN RDOS SINGLE AND MULTIPLE FILE DEVICE CONTROL
* LIST THE RDOS DEVICE NAMES

ARCHITECTURE OVERVIEW

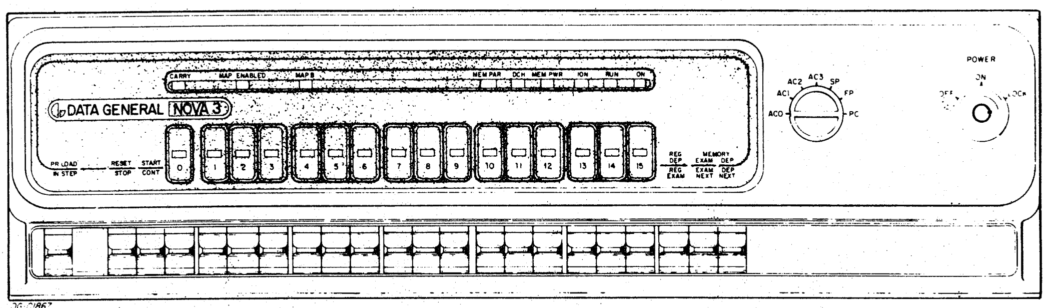


- CORE/SEMICONDUCTOR
- BITS, BYTES, WORDS

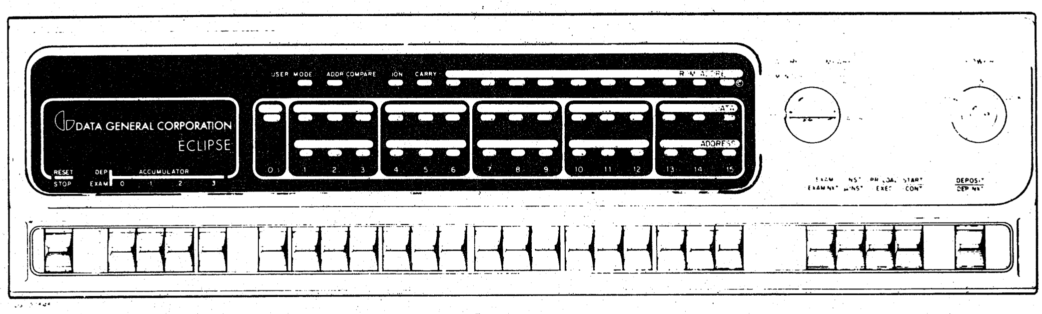
FRONT PANEL OPERATION



NOVA



NOVA 3



ECLIPSE

- ADDRESS/DATA LIGHTS
- DATA SWITCHES
- OFF/ON/LOCK
- START/CONTINUE
- RESET/STOP
- EXAMINE/EXAMINE NEXT
- DEPOSIT/DEPOSIT NEXT
- EXAMINE/DEPOSIT ACCUMULATORS
- PROGRAM LOAD
- FUNCTION LIGHTS

S200

RDOS USER

MODULE 2

BOOTSTRAPPING/STARTUP/SHUTDOWN

MODULE 2

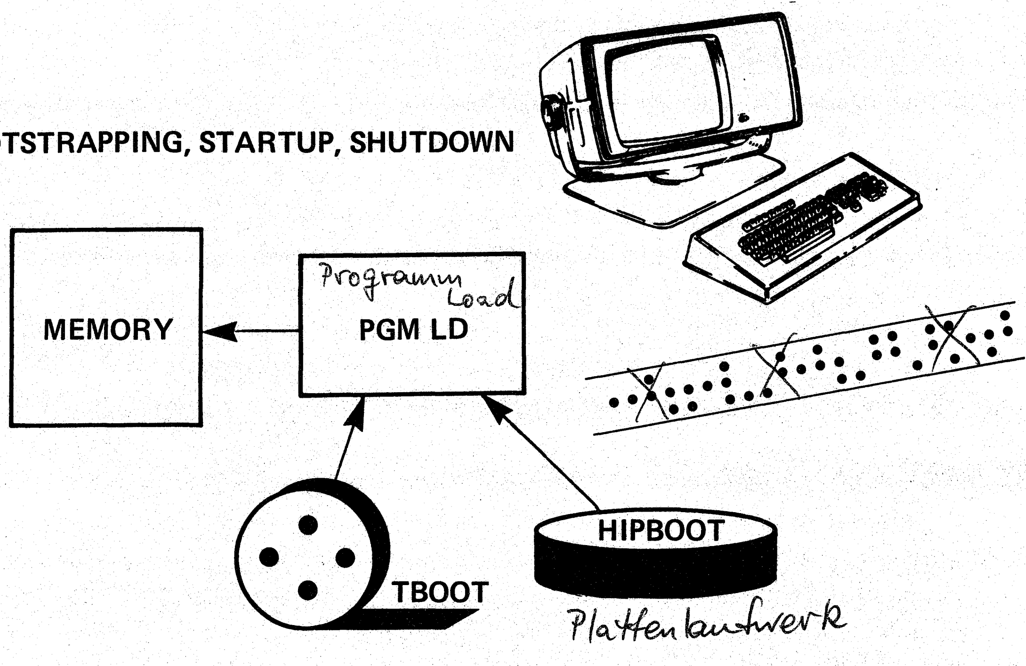
OBJECTIVES

BOOTSTRAPPING STARTUP, SHUTDOWN

Upon successful completion of this module you will be able to:

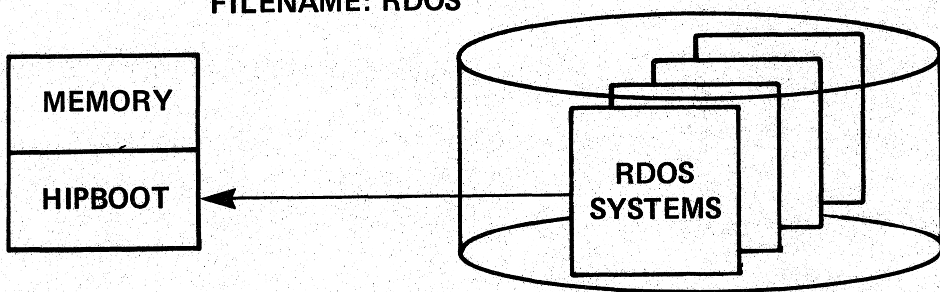
* POWER UP & DOWN ALL DGC COMPUTING EQUIPMENT
* START RDOS RUNNING, THE REQUIRED SOFTWARE, AND THE PROCESS

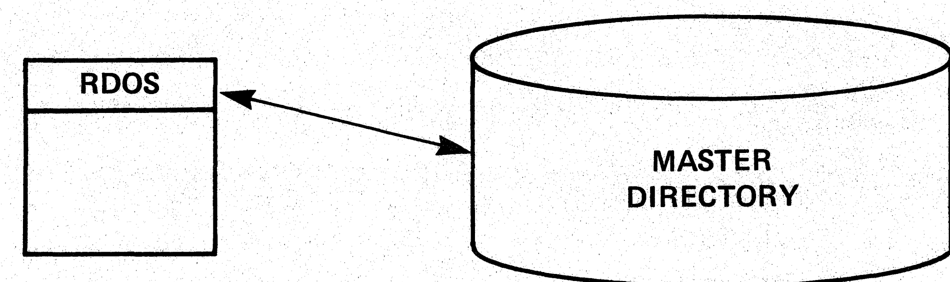
BOOTSTRAPPING, STARTUP, SHUTDOWN



When HIPBOOT is loaded, it may be directed to load any RDOS System

FILENAME: RDOS





Once loaded, RDOS is associated to the disk until shutdown.

BOOTSTRAPPING, STARTUP, SHUTDOWN

Reposition to a Bootstrap Program

Disk (HIPBOOT)	Power Switch OFF → ON (ouch!)	Line Switch OFF → ON (sloooow)	Toggling Stop, Reset AC0 set 1400 061333, Deposit Inst step
Tape (TBOOT)	Reset LOAD ON-LINE	Reset REWIND ON—LINE	

How to perform the Bootstrap

Disk	Set 100033 PGM LD	Tape	Set 100022 PGM LD
Zebra	Set 100027 PGM LD	Paper Tape	Set 000012 PGM LD

NOTICE Bit 0 & Device Codes

BOOTSTRAPPING, STARTUP, SHUTDOWN

Start Up

FILENAME?
DATE (MM/DD/YY)
TIME (HH:MM:SS)

ACCESS CLI.SV
CLI.OL

R

SHUTDOWN:

R	
FG TERM	(HALT FOREGROUND)
R	
ENDLOG	(HALT LOG)
R	
RELEASE %MDIR%	
MASTER DIRECTORY RELEASED	

S200

RDOS USER

MODULE 3

INTRODUCTION TO OPERATING SYSTEMS

MODULE 3

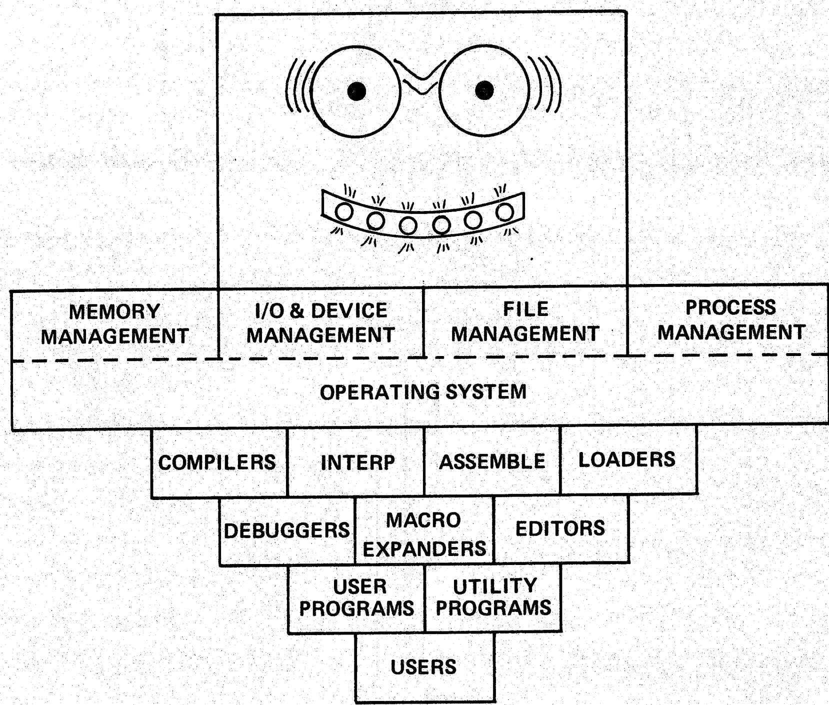
OBJECTIVES

INTRODUCTION TO OPERATING SYSTEMS

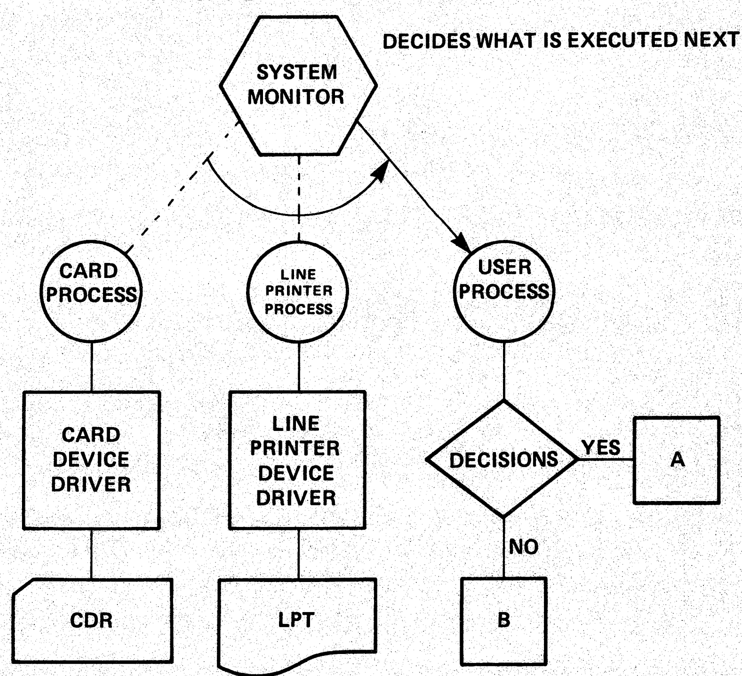
Upon successful completion of this module you will be able to:

* EXPLAIN THE NEED FOR OPERATING SYSTEMS AND THE NECESSARY DEVELOPMENT OF SOFTWARE TO ACCOMODATE OPERATING SYSTEM CONSTRUCTION
* DESCRIBE THE ELEMENTS OF MODERN OPERATING SYSTEMS AND HOW THEY AID THE USER IN MANAGING A COMPUTING ENVIRONMENT
* DEMONSTRATE THE RDOS ANALOGY TO THE MODEL OPERATING SYSTEM

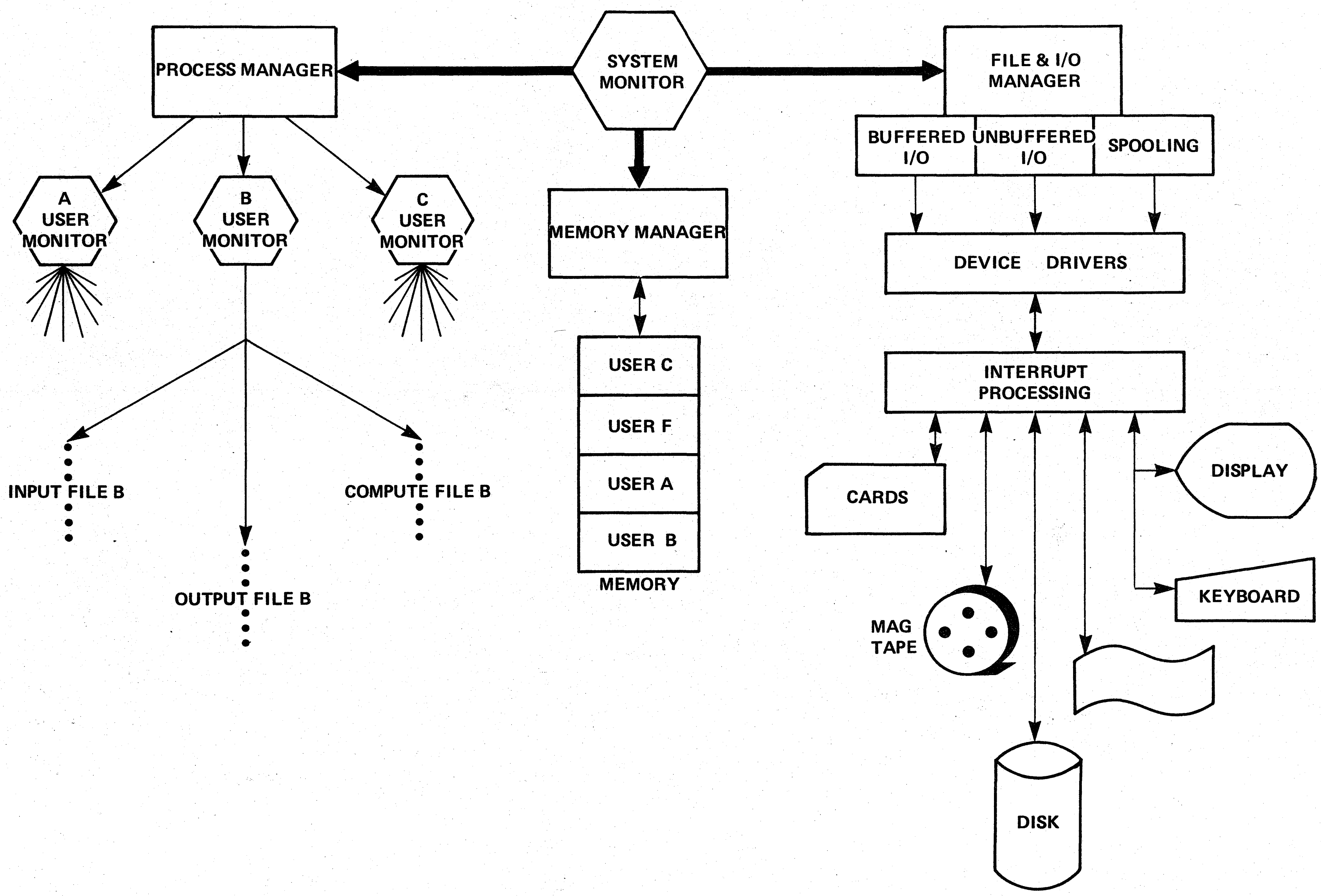
OPERATING SYSTEMS



THE SIMPLE MONITOR SYSTEM



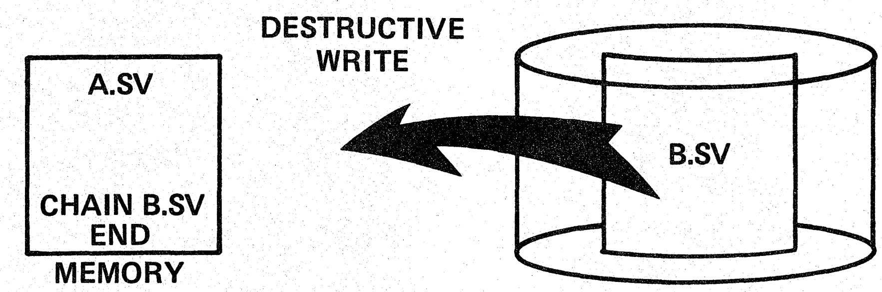
ELEMENTS OF A MODERN OPERATING SYSTEM



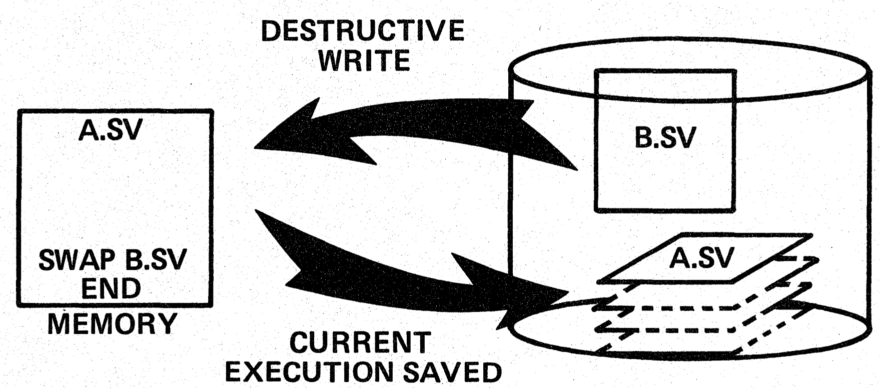
NOTES

RDOS MEMORY MANAGEMENT

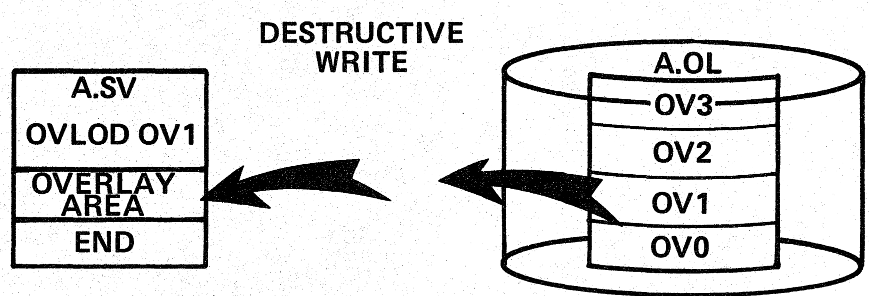
CHAIN: Program invoking via destructive write to memory from disk.



SWAP: Program invoking with previous program execution "pushed" to disk



OVERLAY: Code destructively written to local memory area.



NOTES

RDOS FILE & I/O MANAGEMENT

- SINGLE FILE DEVICES

  DEVICES CONTAINING INDIVISABLE INFORMATION

  EX:
  - CARD READER READS 80 COLUMNS
  - LINE PRINTER WRITES 80 or 132 COLUMNS
  - PAPER TAPE PUNCH WRITES ONE CHARACTER

- MULTIPLE FILE DEVICES

  DEVICES CONTAINING DIVISABLE INFORMATION

  EX:
  - MAG TAPE REELS CONTAIN MANY FILES
  - COMMUNICATION MULTIPLEXOR HAS MANY LINES TO TERMINALS
  - DISK DRIVES CONTAIN MANY DISK FILES
  - MULTICOMMUNICATIONS ADAPTER HAS MANY LINES TO CPU'S.

RDOS SINGLE FILE DEVICE NAMES

$CDR(1)	Card Reader	
$DPI+	Dual Processor (Input)	IPB
$DPO +	Dual Processor (Output)	
$LPT(1)	Line Printer (80 or 132 Columns)	
$PLT(1)	Incremental Plotter	
$PTR(1)	Paper Tape Reader	
$PTP(1)	Paper Tape Punch	
$TTI(1)	Master Console (Input)	
$TTO(1)	Master Console (Output)	
$TTR(1)	Teletype Paper Tape Reader	
$TTP(1)	Teletype Paper Tape Punch	

RDOS allows a primary and secondary controller board for each of the above devices and distinguishes between the two by names having a "1" appended. The hardware is distinguished by adding an octal 40 to the device code (i.e., $LPT - 17, $LPT1-57).

+ Dual processor communications are supported for a primary controller only.

RDOS MULTIPLE FILE DEVICES

Terminals:	Asynchronous Data Communications Multiplexor Terminal identified by line number n, $0 \leqslant n \leqslant 63$ QTY: n
Mag Tape:	Cassette (CT) or Seven/Nine Track Magnetic Tape (MT) Drive identified by unit number, n, $0 \leqslant n \leqslant 7$/controller File identified by file number m, $0 \leqslant m \leqslant 99$/reel.

MTn:M	CTn:M	
MT0:0	CT7:99	primary controller
MT10:0	CT17:99	secondary controller

Disks:

Fixed Head Disk (DK), Unit number n $0 \leqslant n \leqslant 1$
Any RDOS filename (FN)

DKn:FN	
DK0:DIL	primary controller
DK1:C3PO	secondary controller

Moving Head Disk (DP), unit number n, $0 \leqslant n \leqslant 7$

DPn:FN	
DP0:1A	primary: $0 \leqslant n \leqslant 3$
DP7:LAST	secondary: $4 \leqslant n \leqslant 7$

Fixed Plotter Portion on Moving Head Drive (5 Meg.W)

DPnF:FN	
DP0F:YUP	DP7F:DISKCOPY

MORE MULTIPLE FILE DEVICES

Disks:

Floppy Disks are Moving Head Disks
Zebra (DZ) Multiplattered Moving Head Disk, unit number n $0 \leqslant n \leqslant 7$

DZN:FN	
DZ0:BLAH	primary
DZ7:FLAT	secondary

Multiprocessor:

Multiprocessor Communications Adapter (MCA)
Transmit Section (MCAT), Receive Section (MCAR) to/from CPU number n, $0 \leqslant n \leqslant 15$

MCAR:n	MCAT:n	
MCAR:0	MCAT:15	primary
MCAR1:0	MCAT1:15	secondary

Any CPU may communicate to any other CPU or to itself for foreground/background communications.

RDOS DISK FILE NAMING RULES

FILENAM.EX

FILENAME	–	1 to 10 Characters A – Z, 0 – 9, $ ANY ORDER, AT LEAST ONE
EX	–	EXTENSION 0 to 2 Characters A – Z, 0 – 9, $ ANY ORDER
Examples:		ZILON DECOM.15 R2D2 C3PO RDOS6.41

Examples with Directory Specifiers:

DP0:DSKED.SV
UPDATE: BRDOS.PF
UTIL: EDIT.SV
DP0F:SECPART:SUBDIR DATA

RDOS DISK FILE EXTENSIONS

Source File Extensions — Anticipated by Compilers

.SR	—	Assembly	.CB	—	Cobol
.FR	—	Fortran (IV or 5)		—	Basic
.AL	—	Algol	.JB	—	Batch Job File

Those derived from editing

.BU	—	Back Up File	.SC	—	Scratch File

Those derived through program development

.LS	—	Listing File	.RB	—	Relocatable Binary File
.OL	—	Overlay File	.OR	—	Overlay Replacement File
.LB	—	Library File	.LM	—	Relocatable Load Map File
			oder .MP		

Executable File Extensions

.SV	—	Save File	.AB	—	Absolute Binary File

System Utility or Informatory File Extensions

.DR	—	Partition/Directory File	.CM	—	Command File
.KS	—	Data General Keysheet	.MC	—	Macro File
			.PF	—	Patch File
			.TU	-	Tuning File

Those needed by the BASIC program

.SW	=	Swap File	.ID	—	Valid LOGON ID
.AF	—	Accounting File			

Commercial File Extensions

.VL	—	Volume File	.IX	—	Index File

S200

RDOS USER

MODULE 4

INTRODUCTION TO THE COMMAND LINE INTERPRETER

MODULE 4

OBJECTIVES

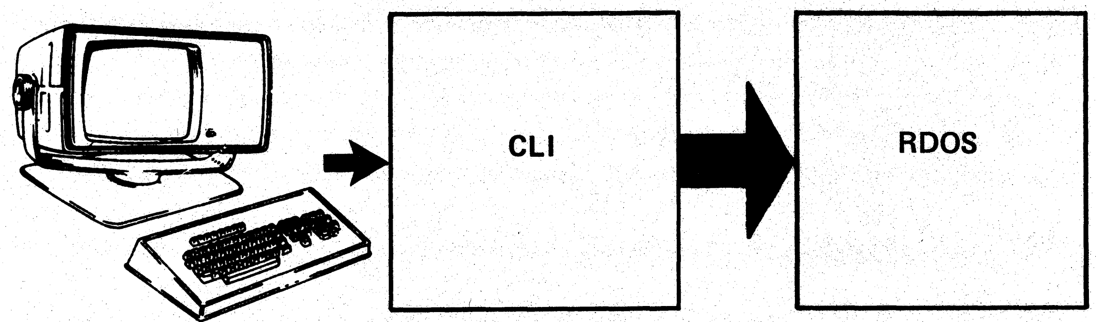
INTRODUCTION TO THE COMMAND
LINE INTERPRETER

Upon successful completion of this module you will be able to:

* IMPLEMENT VALID CLI COMMANDS
* DEMONSTRATE PROPER USE OF LOCAL AND GLOBAL COMMAND SWITCHES
* USE IN–LINE & MULTI–LINE COMMAND EXPANSIONS
* USE SPECIAL SYMBOLS TO CONTROL RDOS VIA CLI
* USE CLI PERCENT VARIABLES, INDIRECT, AND MACRO FILES

COMMAND LINE INTERPRETER

CLI translates a human oriented language into assembly language system calls.



Command/Globals   Arg1/Locals   Arg2/Locals . . . Argn/Locals

Internal Command ?
Macro File on Disk ?
RDOS Program File ?

File not found !

Some Commands take no arguments

```
R                        R                             R
GTOD                     DISK                          MDIR
10/31/79  13:00:00       LEFT:3127  USER:149870        DP0
R                        R                             R
```

```
R
LIST
BULLFROG        22      C
ITCH            259     D
ITCH.SC         259     P
R
```

Some Commands take one argument

```
R              R                        R
STOD 17 45     SDAY 1 1 68              LIST YO
R              R                        YO              0
                                        R
```

More Commands use Multiple Arguments

```
R                      R                      R
DELETE 1 2 3 4         XFER/A FILE $LPT       FILECOM    F1   F2
R                      R                      R
```

Global Switches Alter the Command

LIST	list a filename on disk
LIST/A	list permanent files on disk
LIST/E/A	list everything about permanent files
LIST/S/L	sorted list of files to the line printer

XFER	binary transfer command
XFER/A	ASCII transfer command
XFER/A/B	append this ASCII transfer to a file

```
XFER/A  TEXT  $LPT             XFER/A    $TTI     QTY:0
XFER/A  $TTI   NEWFILE         XFER      TBOOT.SV  MT0:0
```

INTRODUCTION TO THE COMMAND LINE INTERPRETER

Arguments are separated with Spaces or Commas

```
        TYPE  EZRA                TYPE,EZRA
```

Semicolon separates Multiple Commands
Up Caret Ignores Next Character

```
PRINT BZ; GTOD; XFER/A 259 $LPT; PUNCH CHARLES; ^
STOD 17 45; SAVE TOM.S
```

Angles Allow Arguments to Share Common Characters

```
MESSAGE F < 1, 2, 3, 4, 5 > ILE
F1LE    F2LE    F3ILE    F4ILE    F5ILE
```

Parenthesis Generate Multiple Command Lines

```
MESSAGE F(1,2,3,4,5)ILE
F1ILE
F2ILE
F3ILE
F4ILE
F5ILE
```

```
(LIST, DELETE/V) MYFILE
MYFILE    0    D
MYFILE
```

RDOS FILENAME TEMPLATES

— Substitute any number of characters, any value
* Substitute one character, any value

LIST F —		
FFILE1.	88	
FFILE2.	88	

LIST F —·—		
FFILE1.	88	
FFILE2.	88	
FCOM.CM	18	

LIST *		
1.	0	
7.	13	D
Y.	410	SD
Z.	26	D

LIST CLI. —		
CLI.OL	43008	C
CLI.SO	0	D
CLI.TO	0	D
CLI.ER	8704	D
CLI.SV	10752	SD
~~CLI.SO~~	~~0~~	~~D~~

CLI.<SV,OL> ab Rev.7.00 nur noch RCLI.SV

Special Symbols

- RUBOUT BACKSLASH — Deletion
- ↑S ↑Q — Scrolling
- ↑A ↑C — Interrupt
- ↑Z — END OF DATA INPUT

CLI Percent Variable

RDOS will substitute a value for the variable when enclosed between percent signs

MESSAGE "DATE:", %DATE%, "TIME", %TIME%
DATE: 5/2/79 TIME 10:42:30

MESSAGE "MASTER DIRECTORY", %MDIR%
MASTER DIRECTORY DZ0

MESSAGE "CURRENT DIRECTORY", %GDIR%
CURRENT DIRECTORY WORK

MESSAGE "LAST DIRECTORY", %LDIR%
LAST DIRECTORY UTIL

XFER/A %GCIN% TESTFILE/R

XFER/A TESTFILE %GCOUT%

INDIRECT FILES

RDOS will substitute the contents of a disk file if the disk file name is enclosed in "@" signs

TYPE DIRS
FORT4.DR, FORT5.DR, UTIL.DR, WORK.DR, GEN.DR

INIT (@DIRS@)
==⇒ INIT (FOR4.DR,FORT5.DR, UTIL.DR, WORK.DR, GEN.DR)
==⇒ INIT FORT4.DR
==⇒ INIT FORT5.DR
==⇒ INIT UTIL.DR
==⇒ INIT WORK.DR
==⇒ INIT GEN.DR

INDIRECT FILES

```
TYPE COMMANDS
MESSAGE "DATE", %DATE%, "TIME", %TIME%
INIT MT0; DUMP/A/L MT0:0
MESSAGE "ALL FILES IN THIS DIRECTORY DUMPED TO MT0:0"
R
@ COMMANDS@
DATE 5/2/79 TIME 10:50:47
ALL FILES IN THIS DIRECTORY DUMPED TO MT0:0
R
```

MACRO FILES

Files having the .MC extension will have their contents executed as CLI commands.

```
TYPE DIRNIT.MC
DELETE DIRS
BUILD DIRS -.DR
INIT (@DIRS@)
MESSAGE "INITIALIZED", @DIRS@
R
DIRNIT
FILE DOES NOT EXIST DIRS
INITIALIZED 4.DR, 5.DR, UTIL.DR, GEN.DR
```

VOCABULARY

BOOT	–	A bootstrap program used to invoke operating systems or stand-alone programs
TYPE	–	Displays the contents of an argument filename
GSYS	–	Get the system name
STOD	–	Set the time of day
LOG	–	Record CLI communication in (F)LOG.CM
ENDLOG	–	Halt recording of CLI communications
APPEND	–	Join two or more files together.
GTOD	–	Get system time of day
LIST	–	Display bookkeeping of a file
REV	–	Get a program's revision number.
MESSAGE	–	Display a message
PRINT	–	Transfer an argument file to the line printer.
XFER	–	Transfer a source file into a destination file.
SDAY	–	Set the system's date.
.	–	Report time and date with every prompt.

DELETE/C/V
BUILD

S200

RDOS USER

MODULE 5

DISK BASICS

MODULE 5

OBJECTIVES

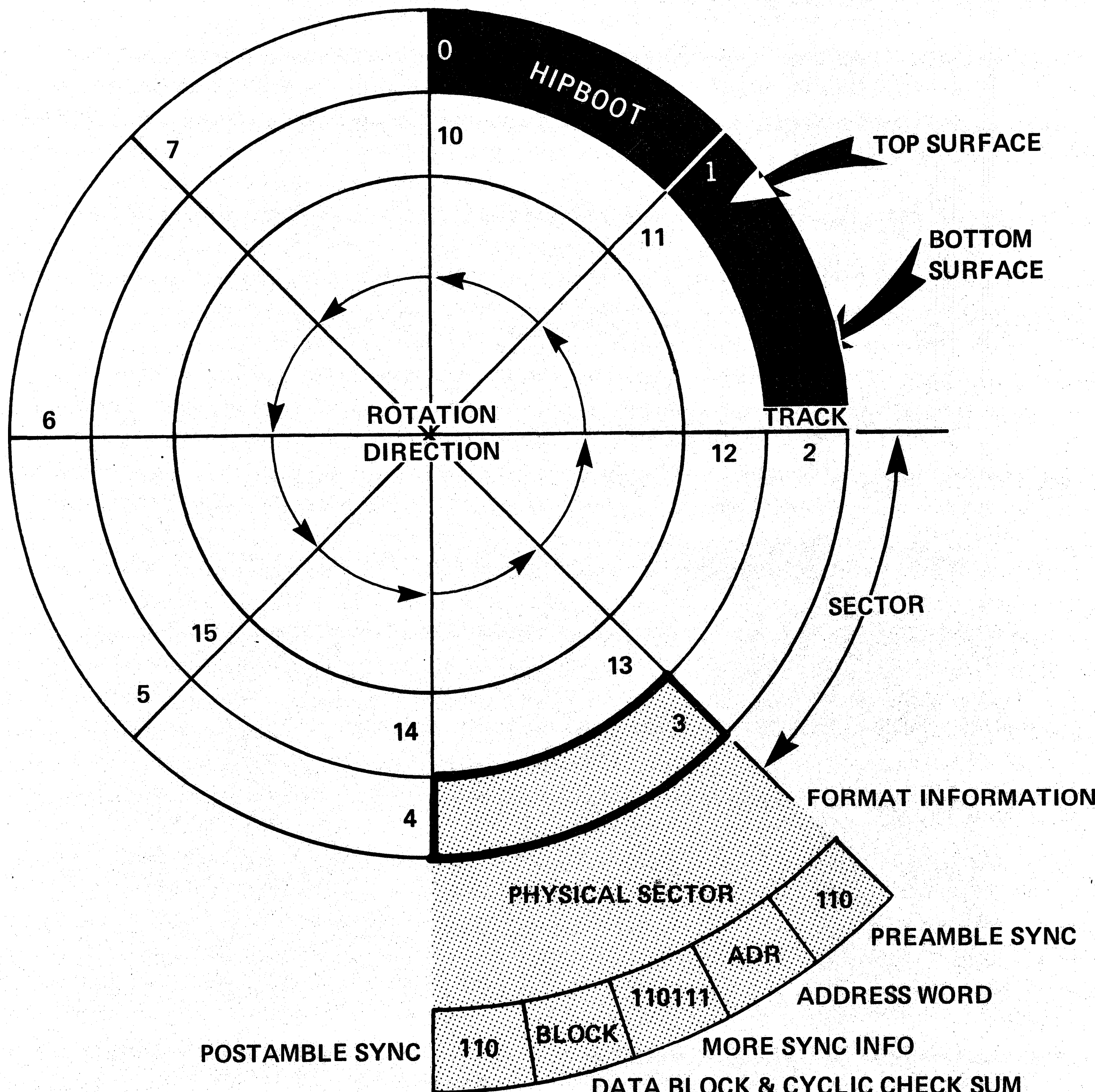
DISK BASICS

Upon successful completion of this module you will be able to:

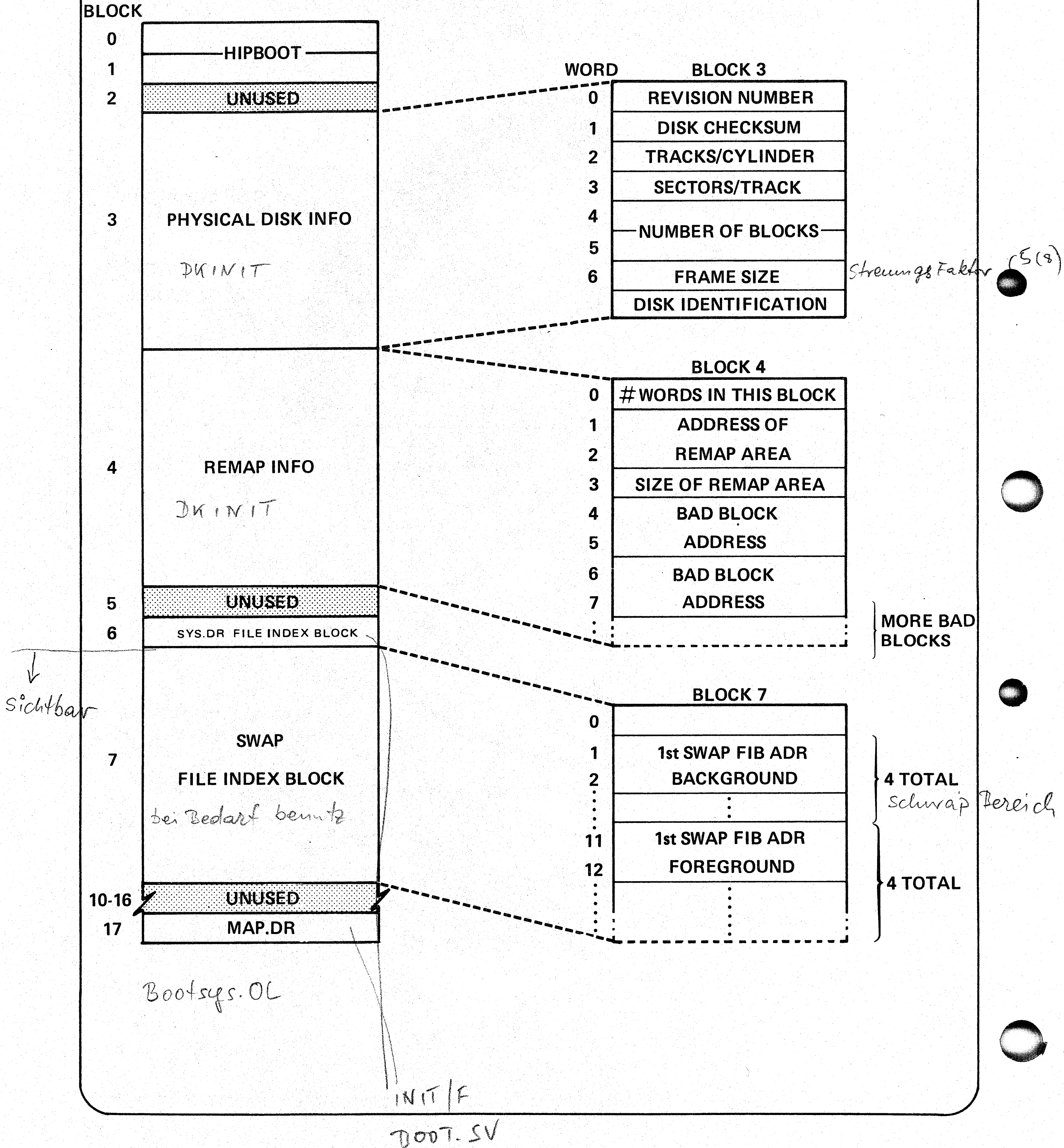
* DESCRIBE RDOS'S DISK BLOCK NUMBERING SCHEME
* DETERMINE WHEN TO FORMAT A DISK, IDENTIFY THE NECESSARY PROGRAMS AND DESCRIBE THE INITIAL STEPS OF DISK FORMATTING.
* DESCRIBE THE LOCATION AND USAGE OF PRELIMINARY DISK BLOCKS WHICH RDOS MUST USE TO INITIALIZE OPERATIONS
* DESCRIBE THE LOGICAL STRUCTURE OF RDOS FILES AND REALIZE THEIR FUNCTIONAL TRADEOFFS

DISK BASICS

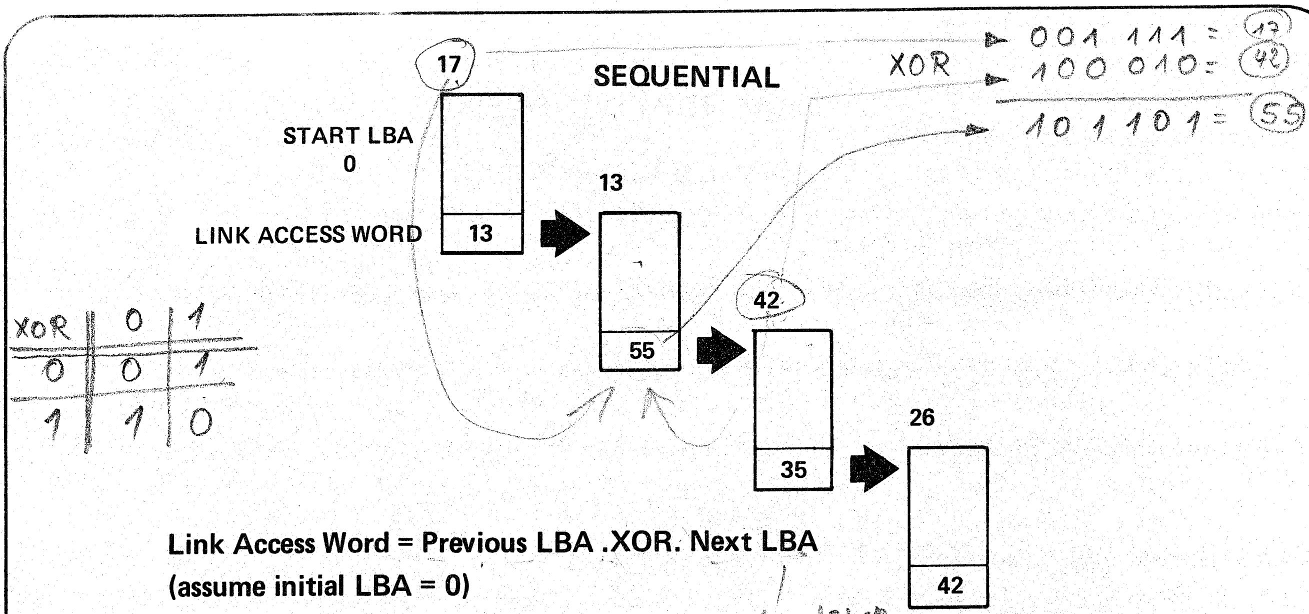
LOGICAL BLOCK ADDRESS = SECTOR, SURFACE, TRACK
1 BLOCK = SECTOR X SURFACE X TRACK



PRELIMINARY DISK BLOCKS



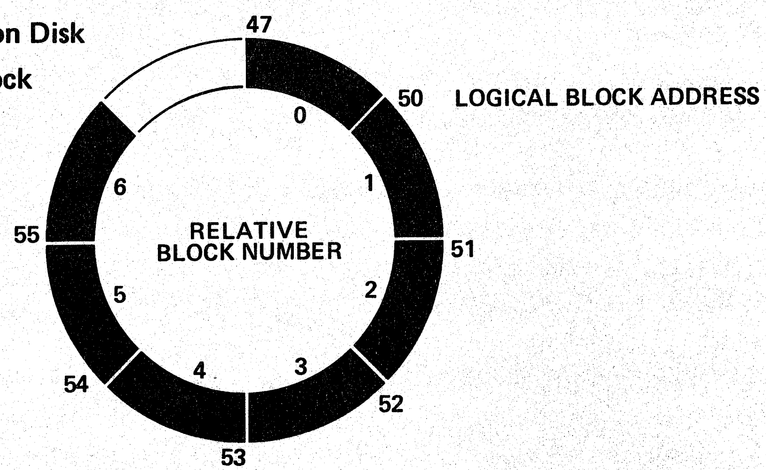
SEQUENTIAL



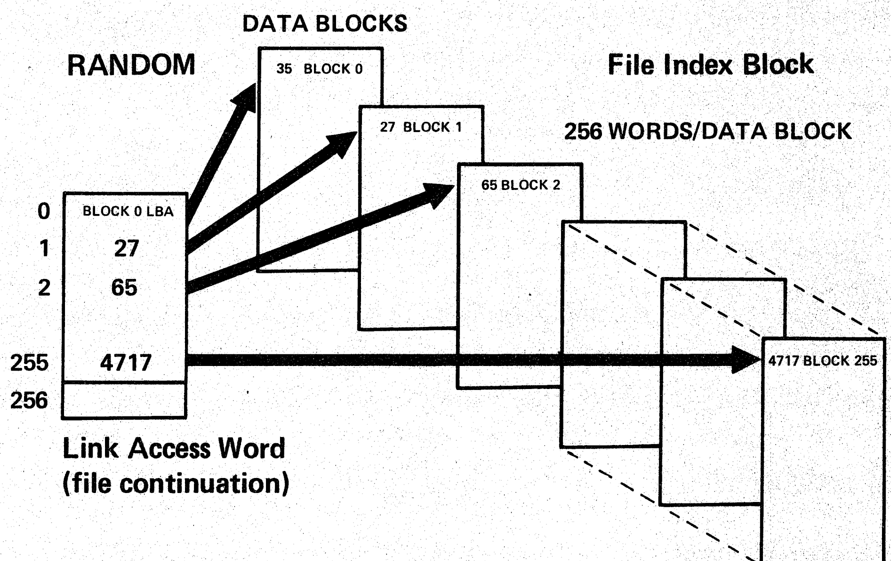
Link Access Word = Previous LBA .XOR. Next LBA
(assume initial LBA = 0)

CONTIGUOUS

Blocks Contiguous on Disk
256 Data Words/Block



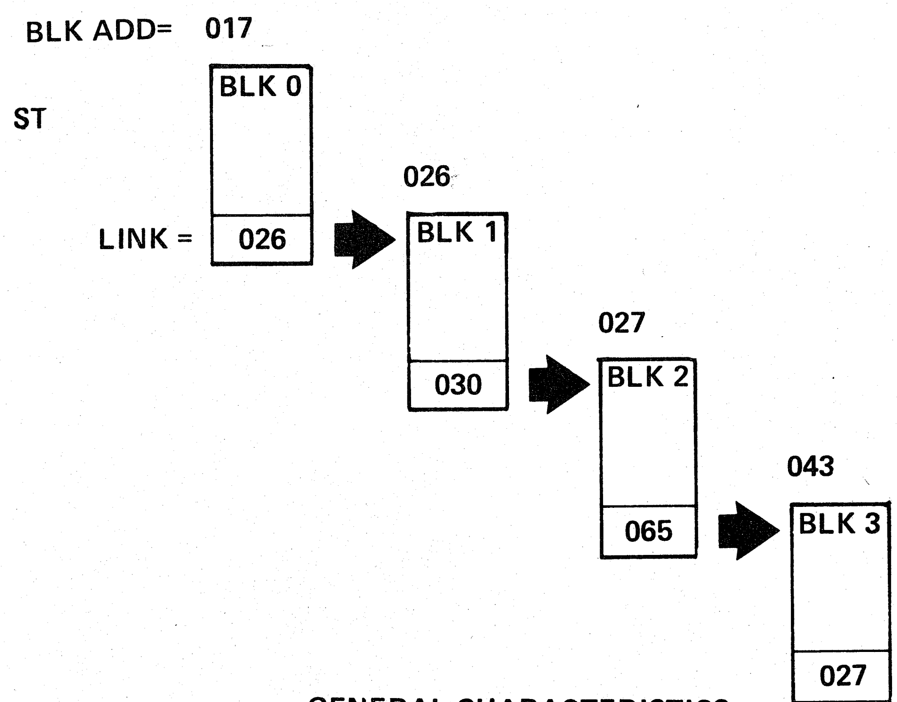
RANDOM



NOTE: All link access word pointers and logical block addresses may require double words to access all blocks on the larger zebra disk pack.

SEQUENTIAL DISC FILES

PHYSICAL ORGANIZATION



GENERAL CHARACTERISTICS

- UP TO 255 DATA WORDS PER BLOCK
- LAST BLOCK PADDED WITH NULLS AS REQUIRED
- LAST WORD IN EACH BLOCK IS A LINK FOR COMPUTING THE NEXT BLOCK ADDRESS.
- SEQUENTIAL ACCESS ONLY.
- CANNOT USE DIRECT BLOCK I/O
- EXPANDABLE
- TO ACCESS A BLOCK, RDOS MUST ACCESS ALL INTERVENING BLOCKS
- MEDIUM SYSTEM OVERHEAD

bis 33 MByte

bei XFER/A Sequential Files

CREATE

CONTIGUOUS DISK FILES

PHYSICAL ORGANIZATION

LOGICAL BLOCK ADDRESS	DATA BLOCKS
1000	BLK 0
1001	BLK 1
1002	BLK 2

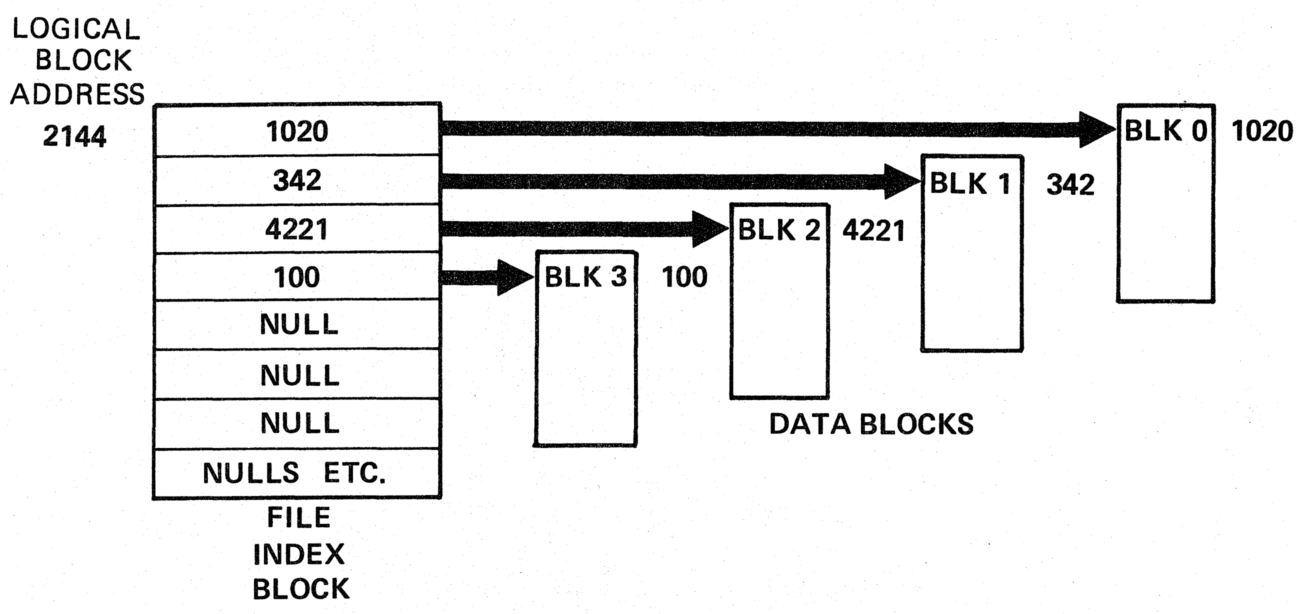
GENERAL CHARACTERISTICS

- UP TO 256 DATA WORDS PER DATA BLOCK
- DATA BLOCKS PADDED WITH NULLS AS REQUIRED
- DATA BLOCKS ASSIGNED AT TIME OF FILE CREATION
- DATA BLOCKS PHYSICALLY CONTIGUOUS ON THE DISC
- CREATION POSSIBLE ONLY IF SUFFICIENT CONTIGUOUS DISC BLOCKS ARE AVAILABLE
- CAN USE ALL FILE ACCESSING METHODS
- FILE SIZE IS FIXED AT TIME OF FILE CREATION AND CANNOT BE ALTERED SUBSEQUENTLY
- FASTEST ACCESSIBLE DATA
  MINIMUM SYSTEM OVERHEAD

C CONT ABC 20 = Blocke (hintereinander)

RANDOM DISC FILES

PHYSICAL ORGANIZATION



GENERAL CHARACTERISTICS

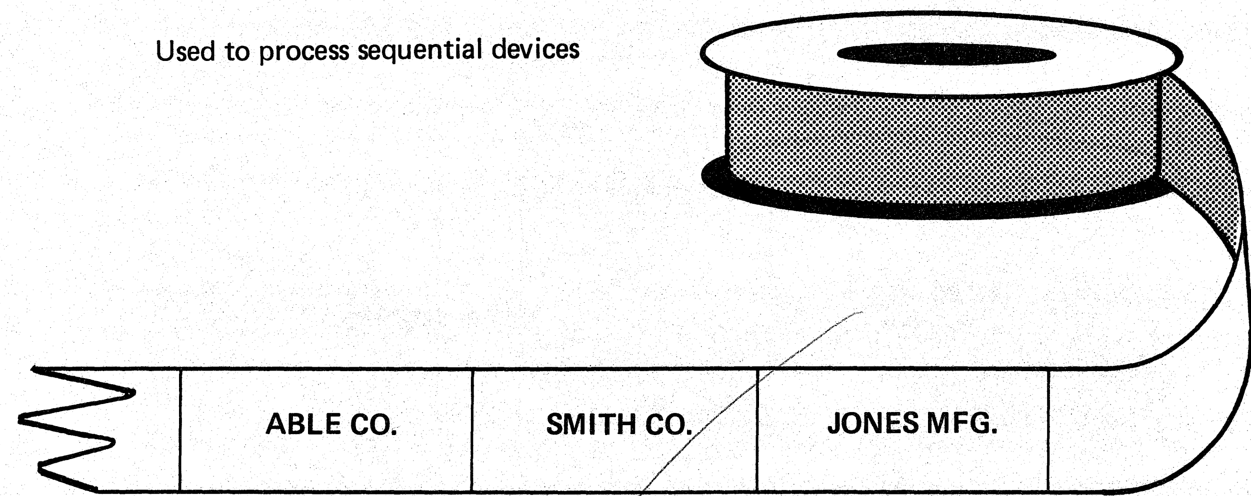
- UP TO 256 DATA WORDS PER DATA BLOCK
- DATA BLOCKS PADDED WITH NULLS AS REQUIRED
- UP TO 255 DATA BLOCK ADDRESSES PER FILE INDEX BLOCK
- FILE INDEX BLOCKS PADDED WITH NULLS TO INDICATE NO DATA BLOCK ASSIGNED
- LAST ENTRY IN FILE INDEX BLOCK IS A LINK TO THE NEXT FILE INDEX BLOCK (IF REQUIRED); LINKING SAME AS SEQ FILE.
- CAN USE ALL FILE ACCESSING METHODS
- EXPANDABLE
- TO ACCESS A DATA BLOCK, RDOS NEEDS ONLY THE APPROPRIATE FILE INDEX BLOCK TO BE CORE RESIDENT

INFOS FILE STRUCTURES OVERVIEW

SAM: Sequential Access Method

SAM retrieves a record from a physically ordered sequence, after examining all preceding records.

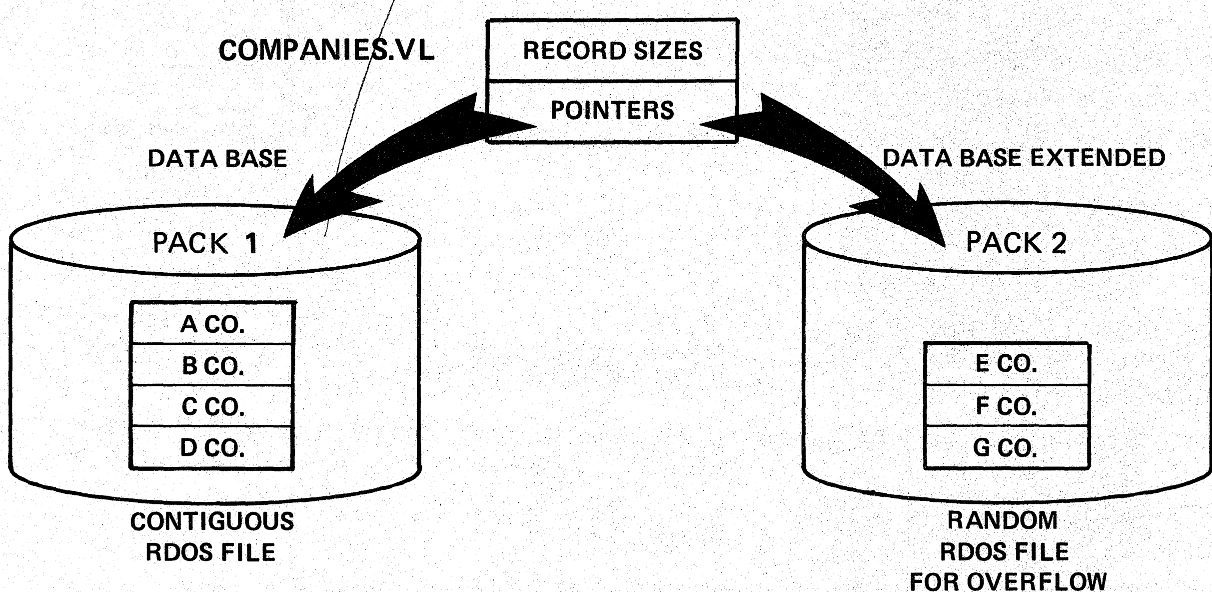
Used to process sequential devices



ABLE & Smith Co. must be read prior to Jones Mfg.

VOLUME DEFINITION FILE .VL

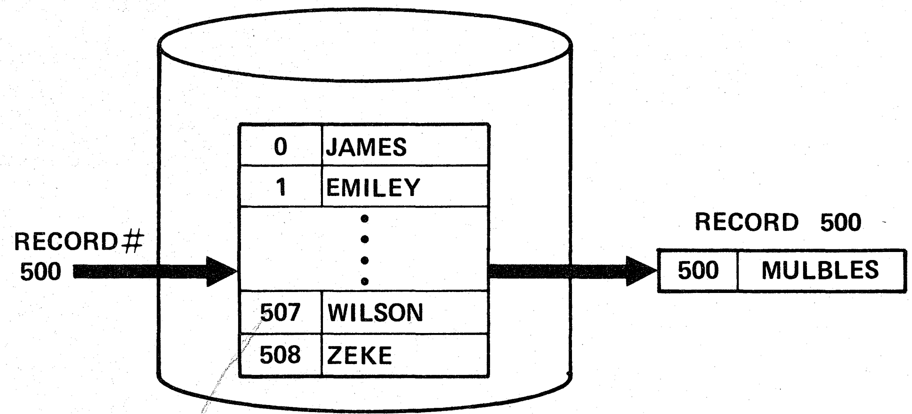
Disk Files have a .VL file for bookkeeping & allow physical device spanning, multiple constructions accomodating overflow



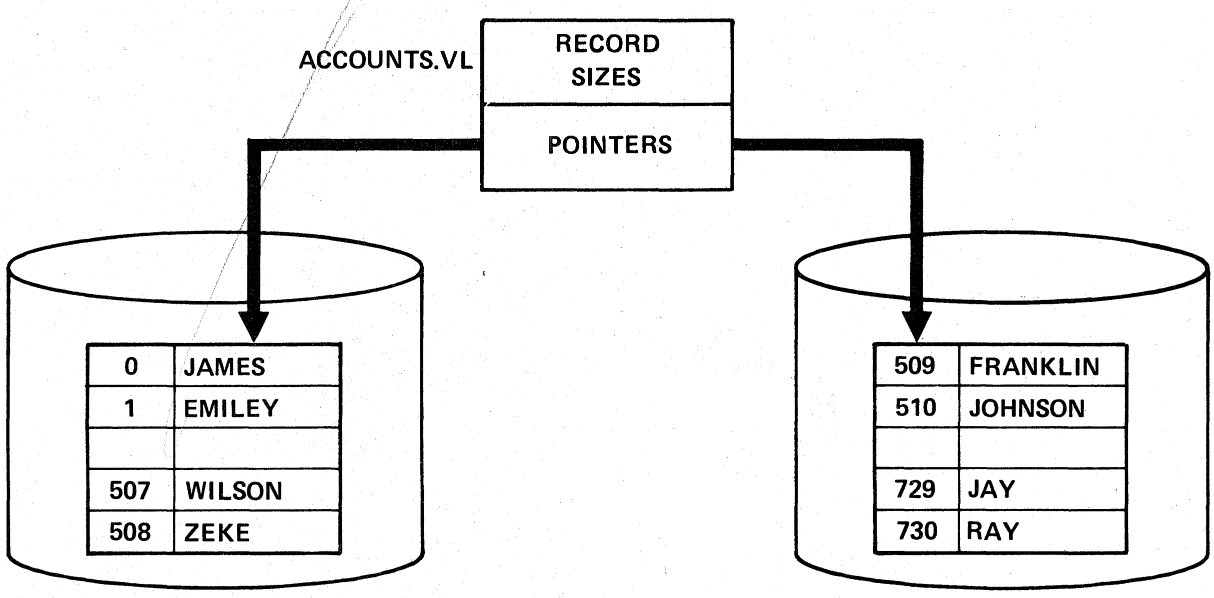
INFOS FILE STRUCTURES OVERVIEW

RAM: Random Access Method

Records Accessible via Relative Record Number



All files use .VL Files to expand physical devices

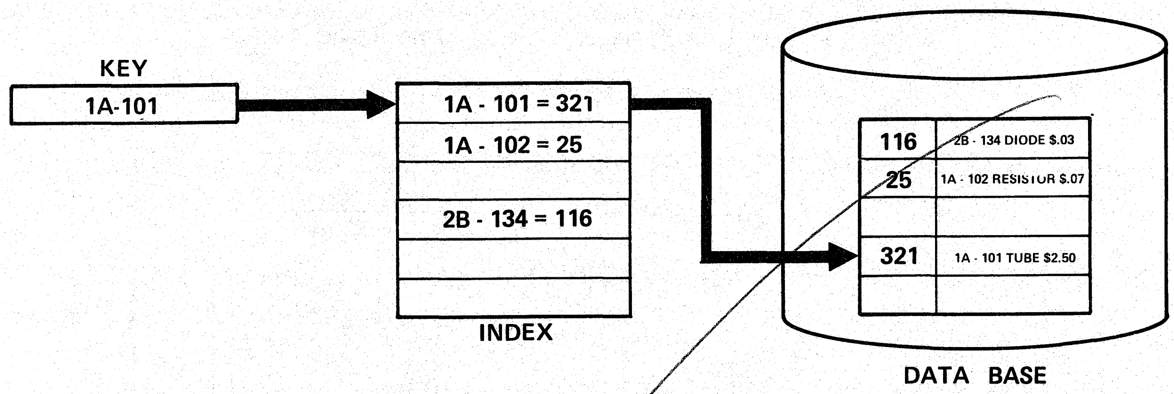


FASTEST RETRIEVAL METHOD

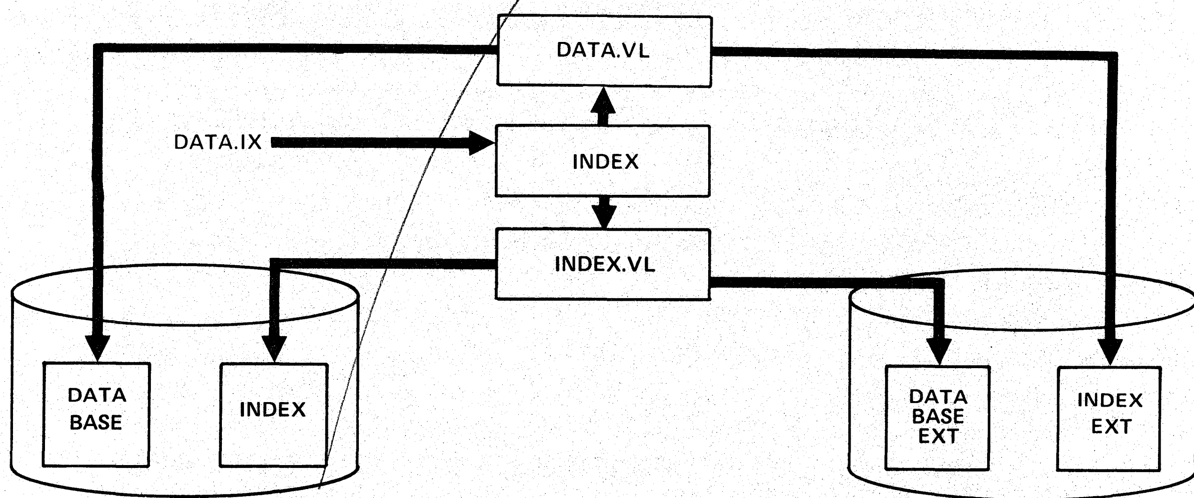
INFOS FILE STRUCTURES OVERVIEW

ISAM: Indexed Sequential Access Method

Records Accessable Via Alphanumeric Keys



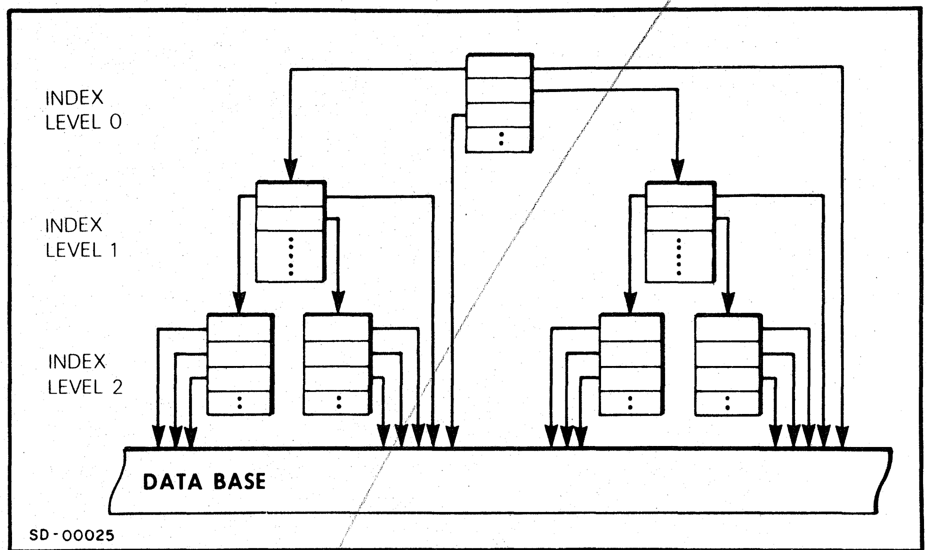
Likewise, .VL will allow expansion



INFOS FILE STRUCTURES OVERVIEW

DBAM: Data Based Access Method

Multiple Indices & Multi-level Indicies, Data Based Access



Again, the .VL file will allow file expansion.

FOR MORE DETAILED INFORMATION SEE THE INFOS STORY BOOK

S200

RDOS USER

MODULE 6

RDOS DIRECTORY STRUCTURE

MODULE 6

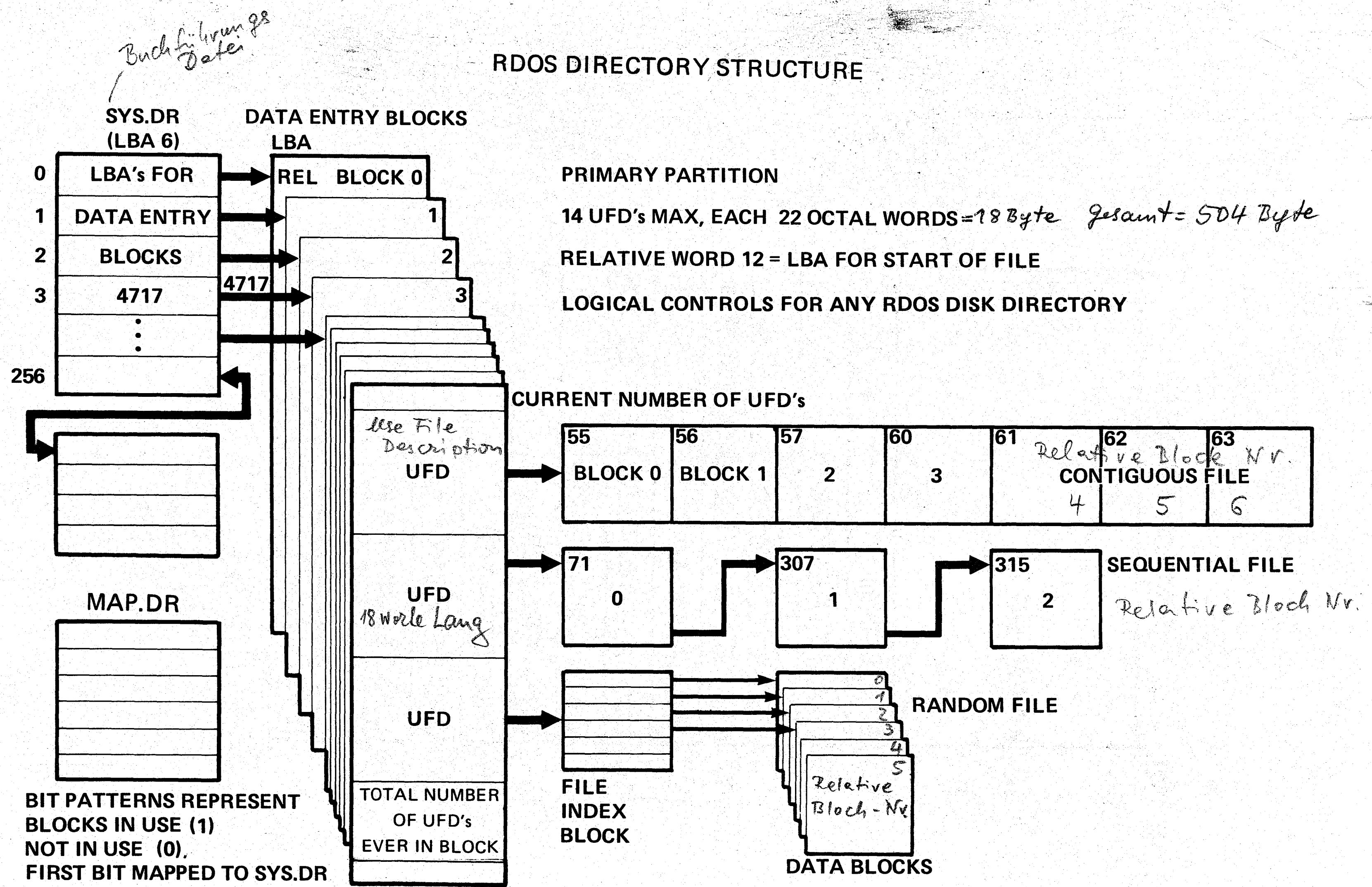
OBJECTIVES

RDOS DIRECTORY STRUCTURE

Upon successful completion of this module you will be able to:

* RESTORE ACCIDENTALLY DELETED FILES
* MANIPULATE AN RDOS DIRECTORY STRUCTURE WITH A DISK EDITOR
* DESCRIBE THE MODULAR ORGANIZATION PARTITIONS AND DIRECTORIES OFFER
* SPAN THE RDOS DIRECTORY STRUCTURE USING LINKS
* DESCRIBE THE DISTINCTIONS BETWEEN PARTITIONS & DIRECTORIES AND IMPLEMENT TECHNIQUES OPTIMIZING THEIR FUNCTIONAL TRADEOFFS

RDOS DIRECTORY STRUCTURE



pro Block (256 Worte) = 14 UFD

pro UFD = 18 Worte Lang

Im Wort 0 sind die Anzahl der UFD's gespeichert

DECIMAL	OCTAL	RESOLUTION UFD
1	0	
2	1	2x5=10 Zeichen
3	2	FILENAME
4	3	
5	4	
6	5	EXTENSION 2 Zeichen
7	6	FILE ATTRIBUTES
8	7	LINK ATTRIBUTES L
9	10	LAST RELATIVE BLOCK NUMBER
10	11	#BYTES IN LAST BLOCK
11	12	LOGICAL ADDRESS FIRST BLOCK
12	13	DATA LAST ACCESSED
13	14	DATA CREATED — 1.1.68 = 0 + T (Kalender)
14	15	TIME CREATED
15	16	UFD TEMPORARY Systeme
16	17	UFD TEMPORARY
17	20	FILE USE COUNT
18	21	DCT LINK Divice Controll Table

Nur für Platten bis 25 MByte

ab 25 MByte ist der 21_8 geteilt Linke helfte für 12_8 und Rechte helfte für 21(8)

FILE CHARACTERISTICS

PHYSICAL CHARACTERISTICS OF A FILE CATALOGUED IN THE RESOLUTION FILE ATTRIBUTE WORD OF THE FILE'S UFD.

C	:	CONTIGUOUS
D	:	RANDOM
	:	DEFAULT IS SEQUENTIAL
T	:	PARTITION
Y	:	DIRECTORY
~~I~~	~~:~~	~~DIRECT BLOCK I/O ONLY~~
~~L~~	~~:~~	~~LINK ENTRY~~

FILE PROTECTION

DIRECT PROTECTION THROUGH USE OF RESOLUTION FILE ATTRIBUTES

P : PERMANENT

W : WRITE PROTECT

R : READ PROTECT

S : SAVE FILE

N : CANNOT BE LINKED TO

? : USER DEFINED

& : USER DEFINED

A : ATTRIBUTE PROTECT

veränderbar nur über DISK EDITOR

$.... SYS.DR BOOTSYS.1

LINK PROTECTION THROUGH THE USE OF LINK ACCESS ATTRIBUTES

USERS OF A FILE THROUGH LINKS SEE A SET OF ATTRIBUTES THAT IS THE OR'ING OF RESOLUTION FILE ATTRIBUTES AND LINK ACCESS ATTRIBUTES

SAME SET OF ATTRIBUTES AS ABOVE

C

USER FILE ATTRIBUTES/ CHARACTERISTICS WORD

ATTRIBUTES BIT POSITIONS			CHARACTERISTICS BIT POSITIONS
0	READ PROTECTED	R	
1	ATTRIBUTE PROTECTED	A	
2	SAVED FILE	S	
	LINK ENTRY	L	3
	PARTITION ENTRY	T	4
	DIRECTORY ENTRY	Y	5
7	NO RESOLUTION	N	
	DIRECT I/O ONLY	I	8
9	USER ATTRIBUTE 1	?	
10	USER ATTRIBUTE 2	&	
	CONTIGUOUS FILE	C	12
	RANDOM FILE	D	13
14	PERMANENT FILE	P	
15	WRITE PROTECTED	W	

RDOS DIRECTORY STRUCTURE

FILENAME RESOLUTION

A. Filename given to system

TYPE FILENAME.EX

B. Filename is Hashed

F	I	43111	2
L	E	46105	Character
N	A	47101	Octal
M	E	46505	Equivalent
0	0	00000	
E	X	42530	
		~~2~~71554	

overflow

soll 1/10 Alle Dateien in der DIR.

```
                 105 5
Frame Size:  65 | 71554   : 65  1055
  (FS)            65
   71554          455
   71521          411
   -----          ---
      33           33    = Hash Value   Rest
```

(Octal Arithmetic)

C Hash Value Offset is Applied to SYS.DR

SYS.DR File Index Block

	offset		
	31	153	
Hash Value	32	154	
Offset →	33	155	→ The Data Entry Block is LBA 155
(HVO)	34	156	
	35	157	
	36	160	

D. Data Entry Block (155) is Searched for

FILENAME.EX

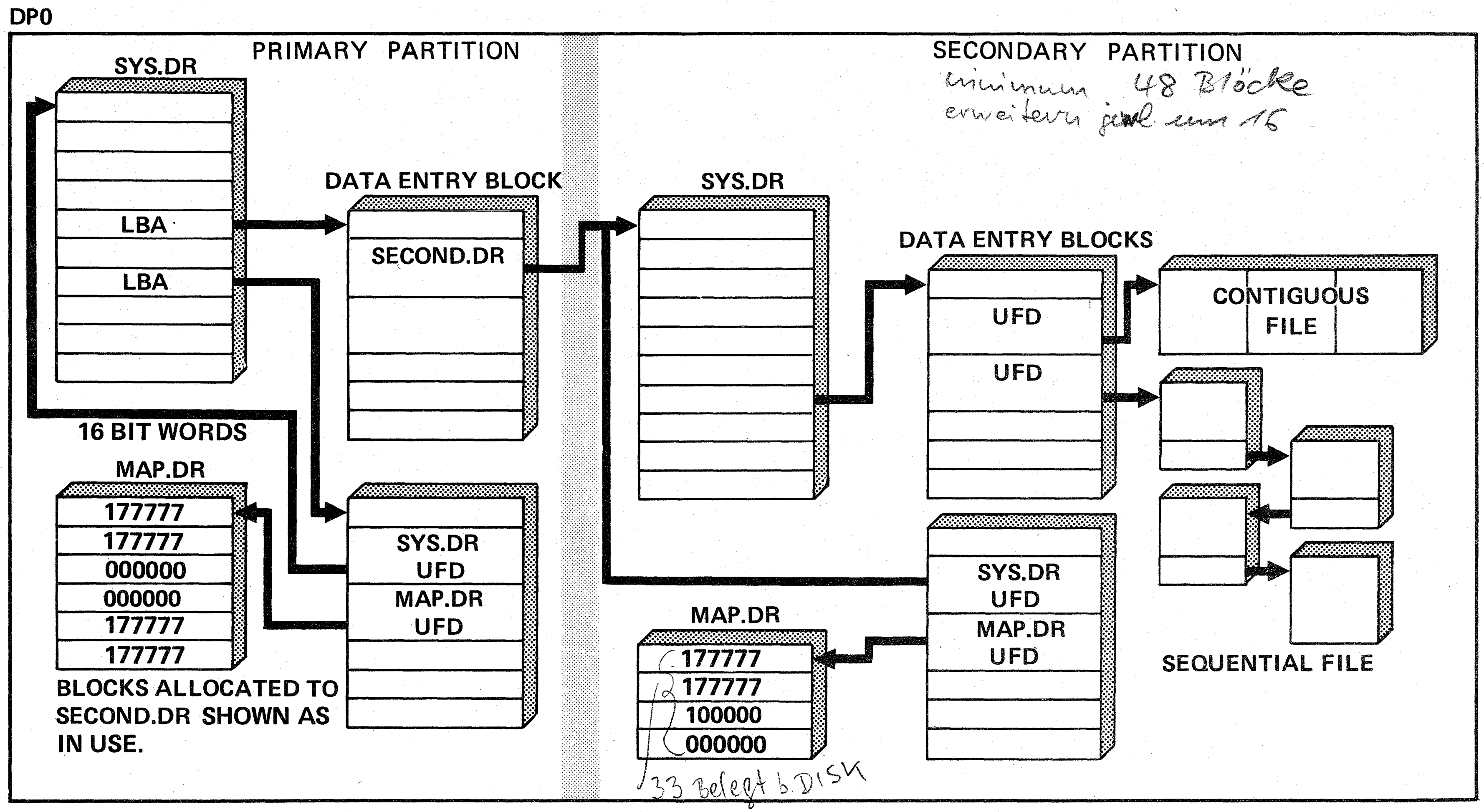
E. Overflow Condition

If Data Entry Block (155) was once filled
(i.e., Total UFD's Ever = 14)

Hash Value Offset = 33 + 65 = 120
(Next)

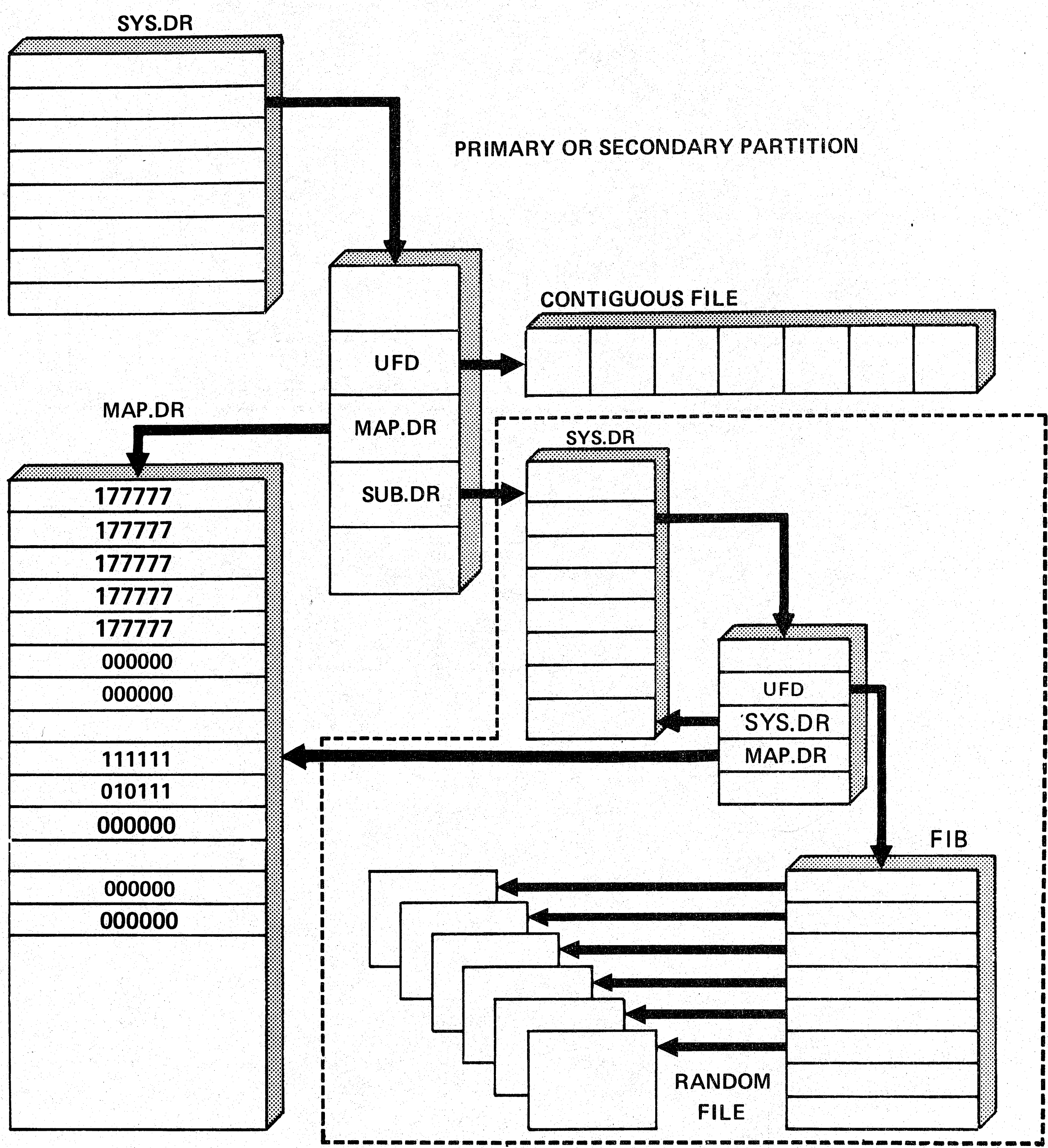
——— Look Again ———

RDOS DIRECTORY STRUCTURE
SECONDARY PARTITION STRUCTURE



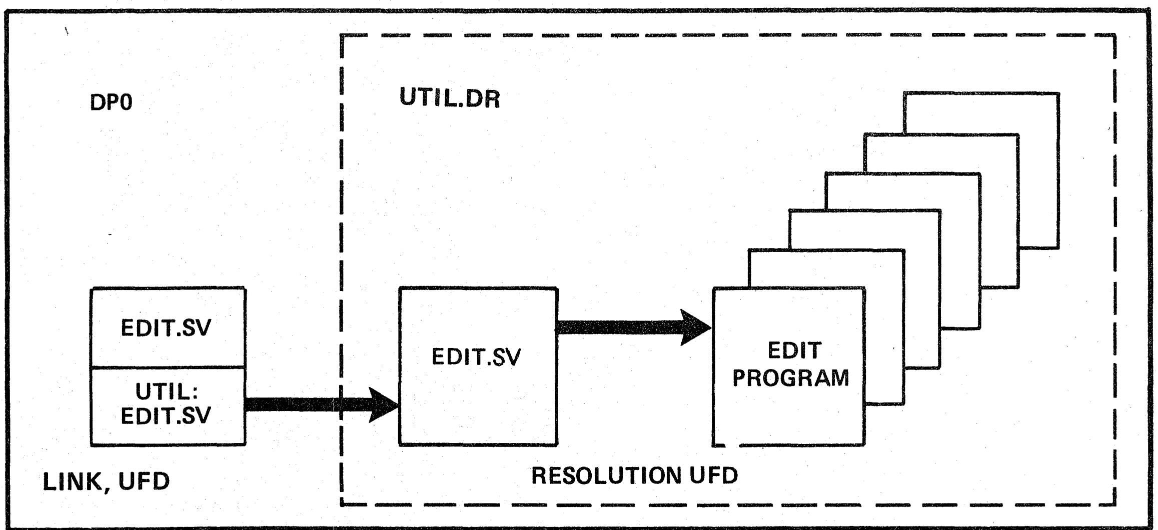
BLOCK EXCLUSION MAINTAINED VIA INDEPENDENT MAP.DR FILES

RDOS DIRECTORY STRUCTURE
SUBDIRECTORY STRUCTURE



A SHARED MAP DIRECTORY ALLOWS SHARED DISK SPACE USAGE
WHILE SEPARATE SYS.DR FILES ALLOW DISTINCT FILE NAME RESOLUTION

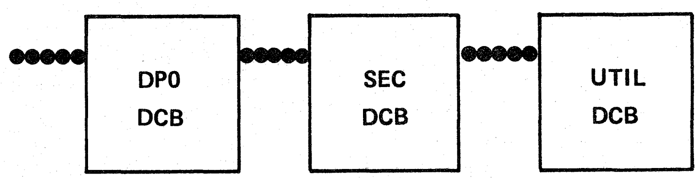
LINKS - REFERENCES ACROSS DIRECTORY/PARTITION BOUNDARIES



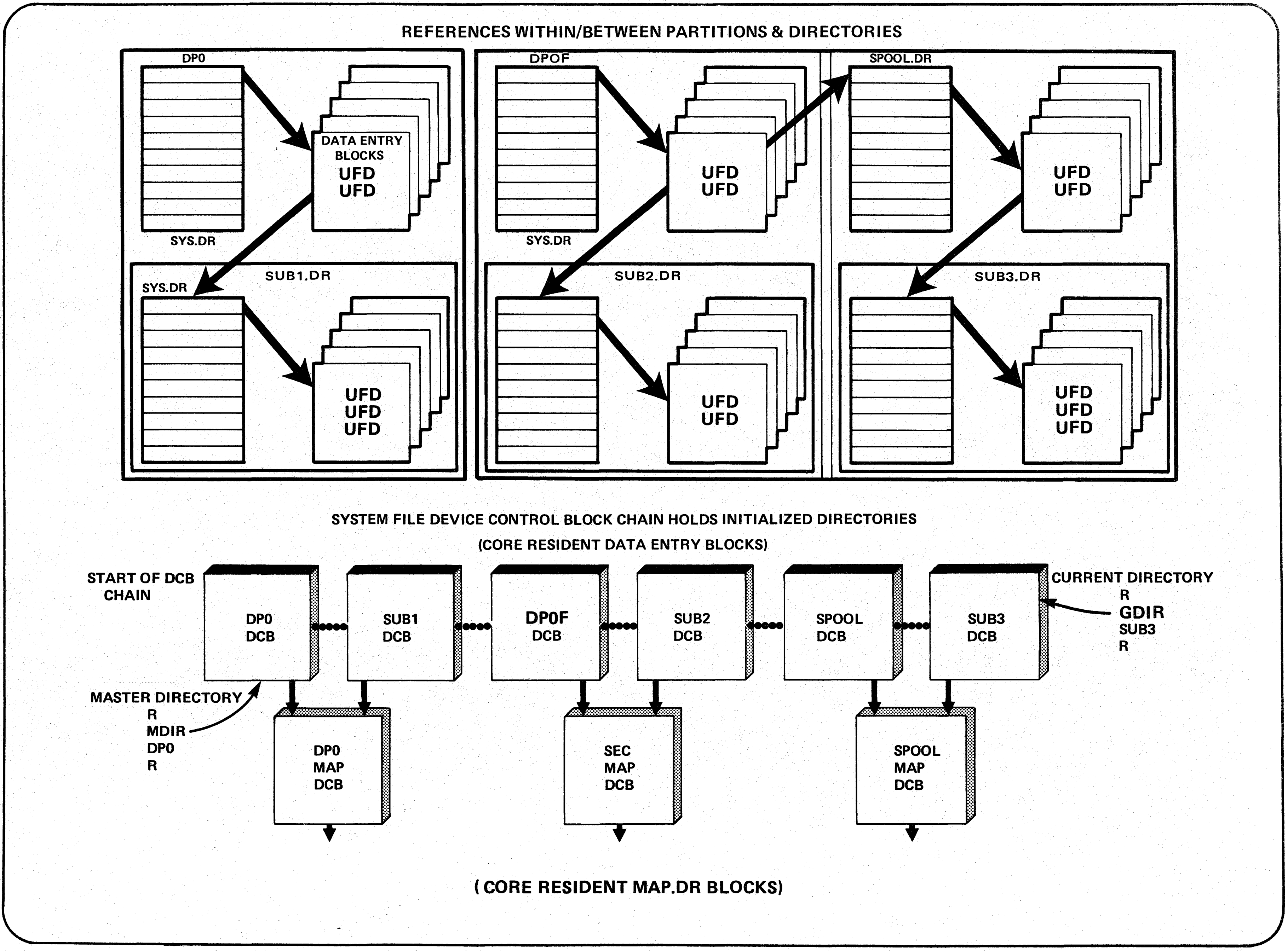
SYSTEM FILE DEVICE CONTROL BLOCKS

SCAN DCB

DIR:



	R
Link Creation:	LINK EDIT.SV UTIL:EDIT.SV)
	R
Link Resolution:	R
	INIT UTIL
	R
	EDIT
	*
	R
Link Removal:	UNLINK EDIT.SV ** Never Delete Links **
	R
	UNLINK —·—
	R
	R
Link Access Exclusion:	DIR UTIL
	R
	CHLAT EDIT.SV PW **LINK USERS CANNOT DELETE EDIT VIA THE LINK**
	R


REFERENCES WITHIN/BETWEEN PARTITIONS & DIRECTORIES
DP0
DATA ENTRY BLOCKS
UFD
UFD
SYS.DR
SUB1.DR
SYS.DR
UFD
UFD
UFD
DPOF
UFD
UFD
SYS.DR
SUB2.DR
UFD
UFD
SPOOL.DR
UFD
UFD
SUB3.DR
UFD
UFD
UFD
SYSTEM FILE DEVICE CONTROL BLOCK CHAIN HOLDS INITIALIZED DIRECTORIES
(CORE RESIDENT DATA ENTRY BLOCKS)
START OF DCB CHAIN
DP0 DCB
SUB1 DCB
DP0F DCB
SUB2 DCB
SPOOL DCB
SUB3 DCB
CURRENT DIRECTORY R GDIR SUB3 R
MASTER DIRECTORY R MDIR DP0 R
DP0 MAP DCB
SEC MAP DCB
SPOOL MAP DCB
(CORE RESIDENT MAP.DR BLOCKS)

S200

RDOS USER

MODULE 7

ALTERING INFORMATION ON THE DISK

MODULE 7

OBJECTIVES

ALTERING INFORMATION ON THE DISK

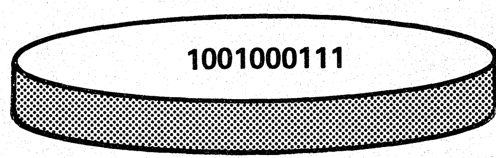
Upon successful completion of this module you will be able to:

* PERFORM EDITING OF AN RDOS DISK WITH THE DISK EDITOR

ALTERING THE INFORMATION ON DISK

STAND ALONE DSKED.SV

Informationen v. Platte holen od. ... auf d. Platte bringen Bitweise



R

BOOT DSKED ↲

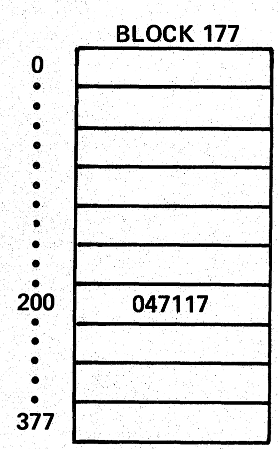
DISK MODEL NUMBER______ ↲

DISK UNIT NUMBER______ ↲

Display Disk Addresses

- Block Number: Word Number/Contents
- 177:200/047117

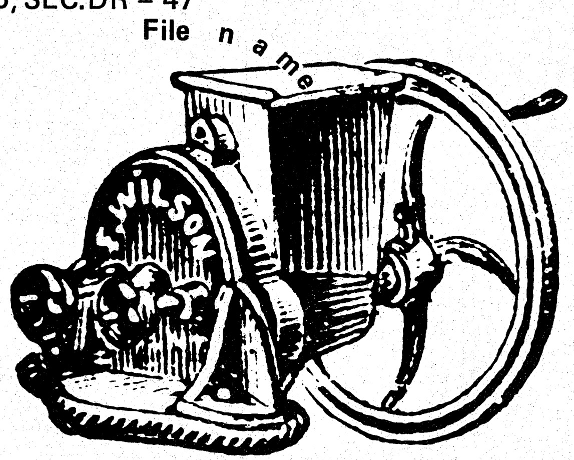
Block Nr. Wort Inhalt (Oktal)



3:6/ FRAME SIZE

Filename Hashing:

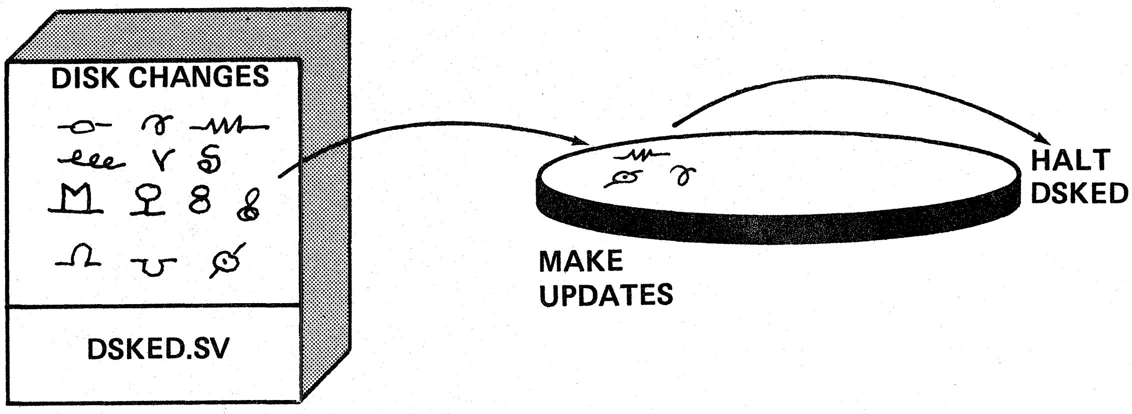
- Frame Size; Filename = Hash Offset (Rest)
- 65; SEC.DR = 47



HVO

STANDALONE DSKED.SV

Disk is Updated at Termination: $Z



Local Commands: Output Formats: .$Command

Apostrophy:	$'	ASCII Output / AB
Asterisk:	$*	Octal Numeric Output /075177

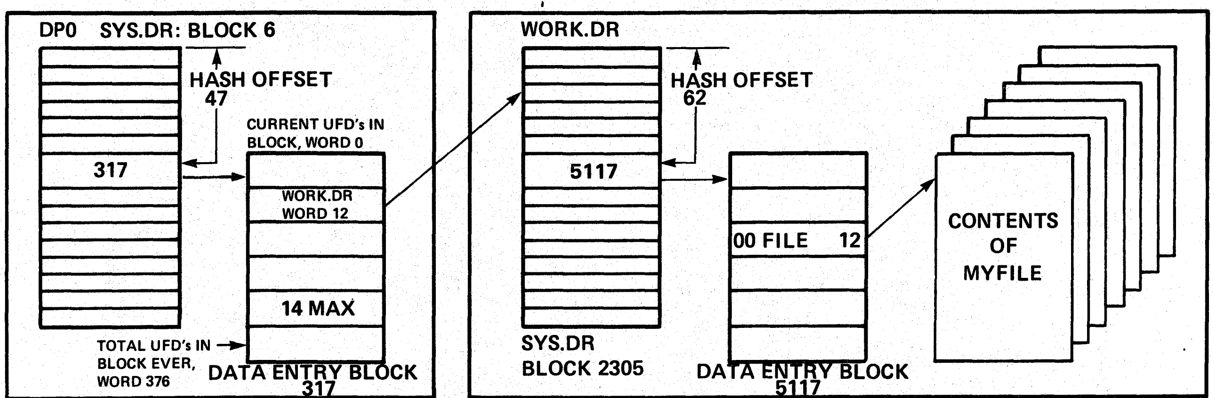
Disk Control Commands: .Command

- Line Feed — Open & Display Next Location
- Up Caret — Open & Display Last Location
- Carriage Return — Close Current Location

NOTE: $ is ESC

ALTERING THE INFORMATION IN DISK

DSKED AND THE DIRECTORY STRUCTURE



Restoration of a Deleted File

R
BOOT DSKED

Disk Model Number ~~4234~~ 6070 (Nova)

Disk Unit Number DP0

- 3:6/65 (Frame Size) 123 siehe Seite 5-2
- 65;WORK.DR = 47 (Hash offset) 122
- 65;MYFILE = 62 (Hash offset) Nova 51
- 6: 47/317 1 (Entry Block)
- 317:13/2305 (SYS.DR for WORK.DR)

(May have to line feed through Entry Block to find SYS.DR LBA)

Wort 13 v Date Entry Block = Wort 12 von UFD Logical Adresse First Block sieh. Seite 6-2

- 2305:62/5117 (Entry Block) MYFILE
- 5117:23/000000 (Deleted Filename)
- 5117:23/000000 "MY (Filename Restored)
- 5117:0/000001 (Single UFD in Entry Block)
- 5117:0/000001 : 2 (2 UFD's in Entry Block)
- $Z (Update Disk - File Restored)

FILENAME? (Reboot System)

(MAP.DR bits in error; move file away from partition, redelete file, move file back to partition)

S200

RDOS USER

MODULE 8

PROGRAM DEVELOPMENT

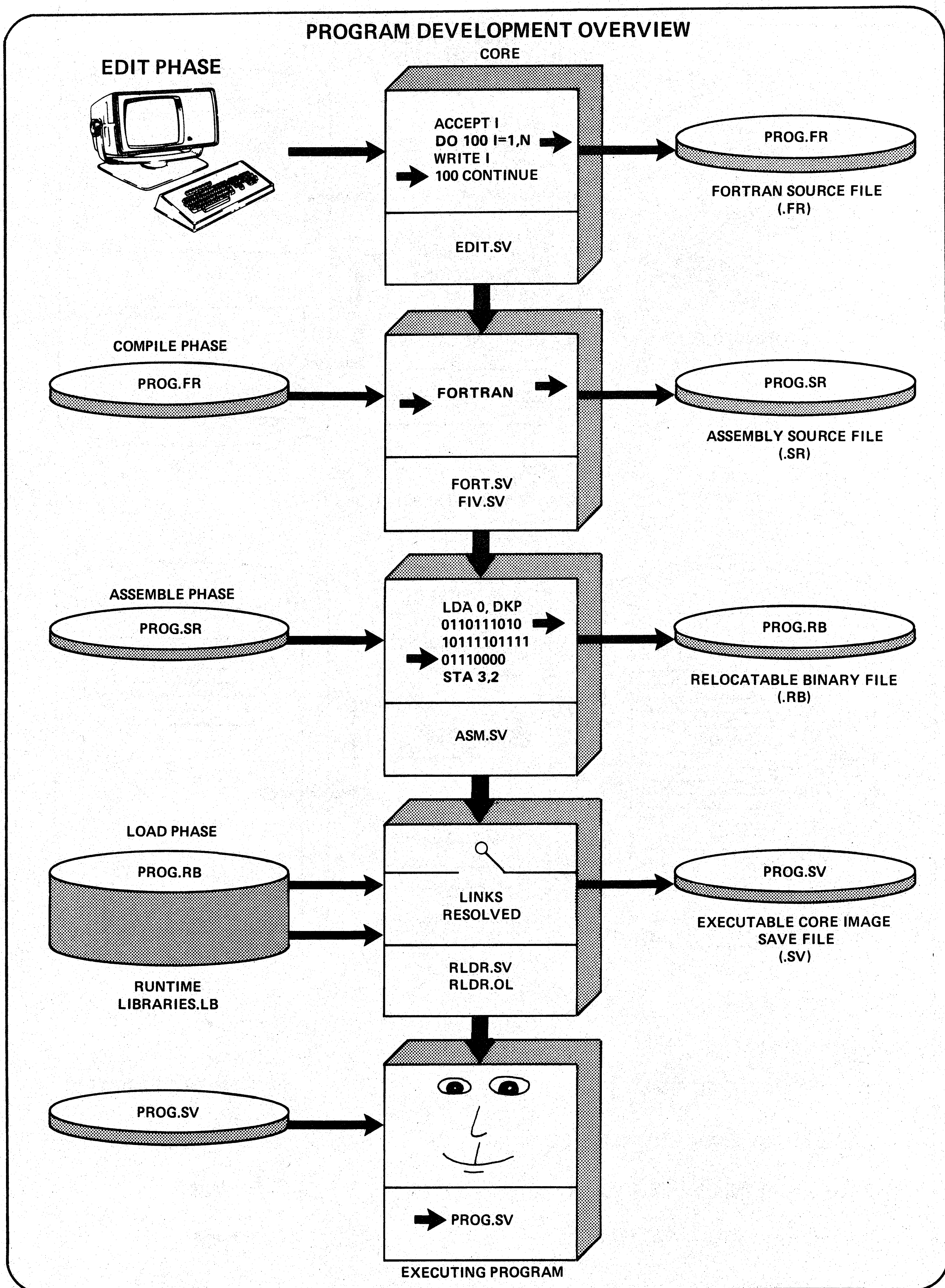
MODULE 8

OBJECTIVES

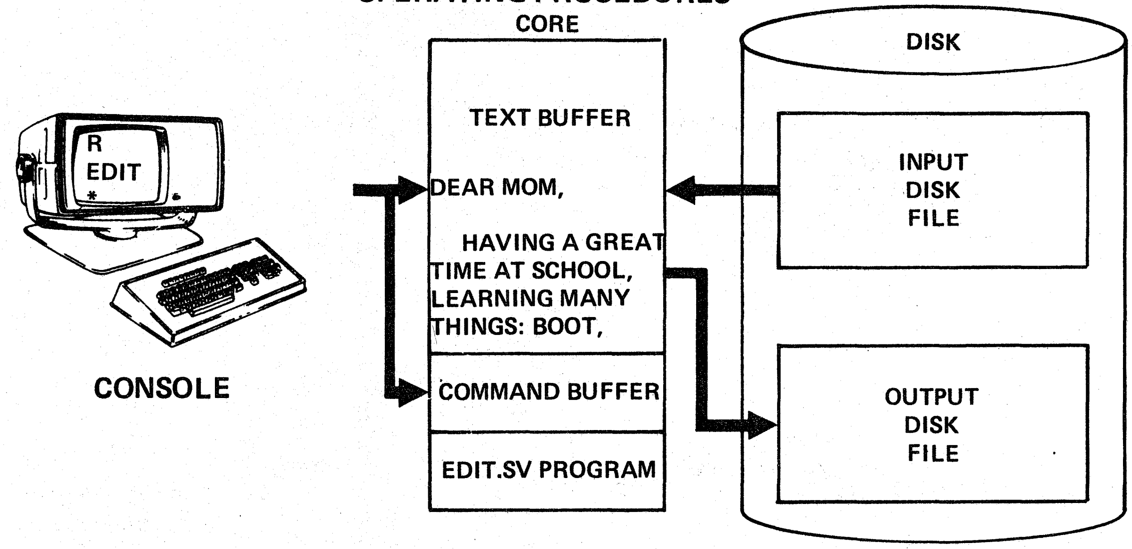
PROGRAM DEVELOPMENT

Upon successful completion of this module you will be able to:

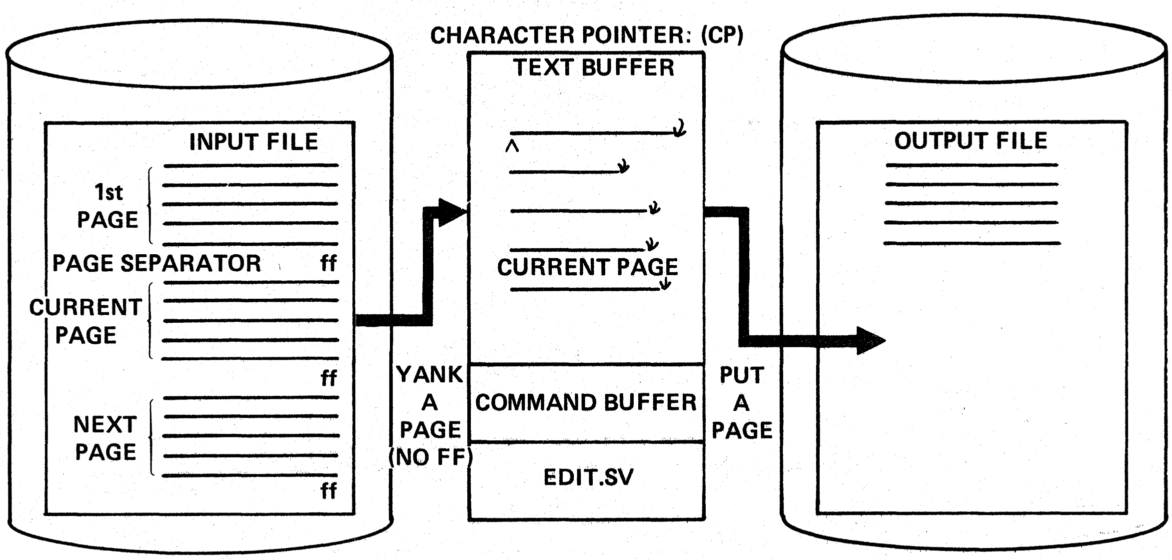
* OPERATE THE TEXT EDITOR
* PERFORM PROGRAM DEVELOPMENT ON SOURCE MODULES
* UNDERSTAND PROGRAM DEVELOPMENT WELL ENOUGH TO ACCOMODATE STRATEGIC ERRORS


PROGRAM DEVELOPMENT OVERVIEW
CORE
EDIT PHASE
ACCEPT I
DO 100 I=1,N
WRITE I
100 CONTINUE
EDIT.SV
PROG.FR
FORTRAN SOURCE FILE (.FR)
COMPILE PHASE
PROG.FR
FORTRAN
FORT.SV
FIV.SV
PROG.SR
ASSEMBLY SOURCE FILE (.SR)
ASSEMBLE PHASE
PROG.SR
LDA 0, DKP
0110111010
10111101111
01110000
STA 3,2
ASM.SV
PROG.RB
RELOCATABLE BINARY FILE (.RB)
LOAD PHASE
PROG.RB
RUNTIME LIBRARIES.LB
LINKS RESOLVED
RLDR.SV
RLDR.OL
PROG.SV
EXECUTABLE CORE IMAGE SAVE FILE (.SV)
PROG.SV
PROG.SV
EXECUTING PROGRAM

SOURCE CREATION EDIT.SV OPERATING PROCEDURES



FORMATS & FILE ASSOCIATIONS



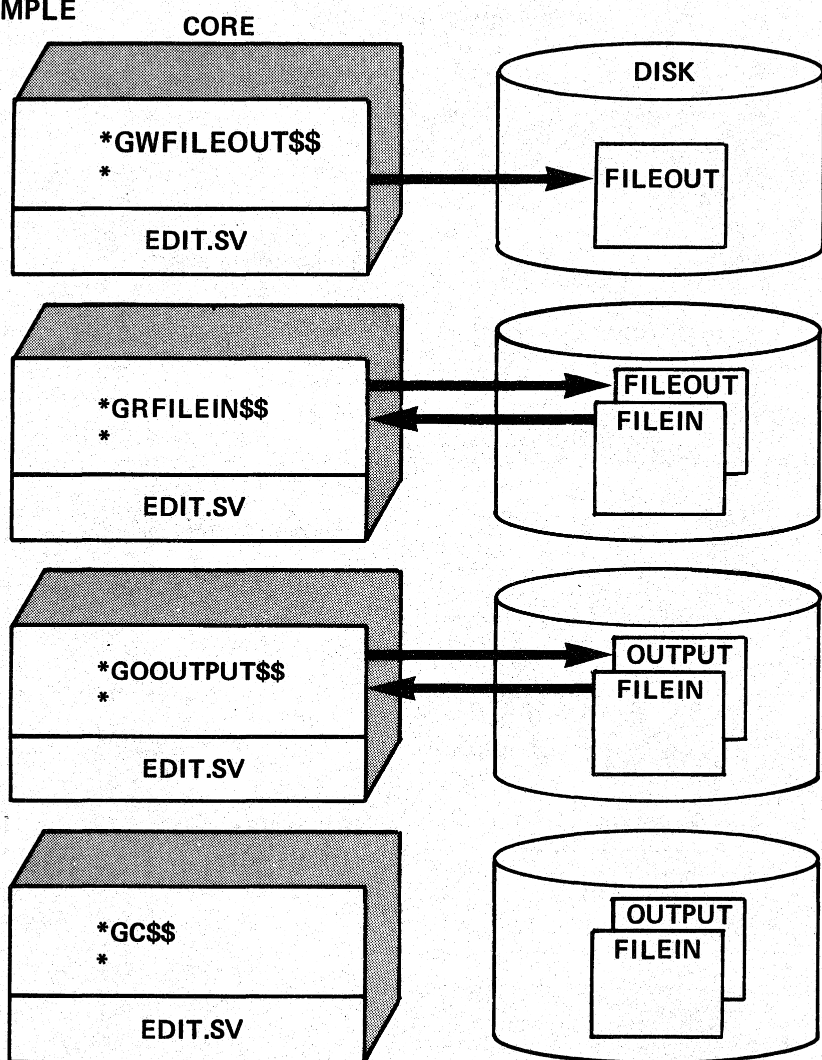
TO ASSOCIATE AN INPUT FILE:	*GRFILEINPUT$$
TO ASSOCIATE AN OUTPUT FILE:	*GWFILEOUTPUT$$
PAGES ARE INPUT WITH YANK:	*Y$$
PAGES ARE OUTPUT WITH PUT:	*P$$

EDIT COMMANDS AND EXAMPLES

File Association Commands:

* GRFILENAME$$	—	Associate FILENAME for Input
* GWFILENAME$$	—	Associate FILENAME for Output
* GOFILENAME$$	—	Close current output file, open FILENAME for Output
* GC$$	—	Close all file associations.

EXAMPLE



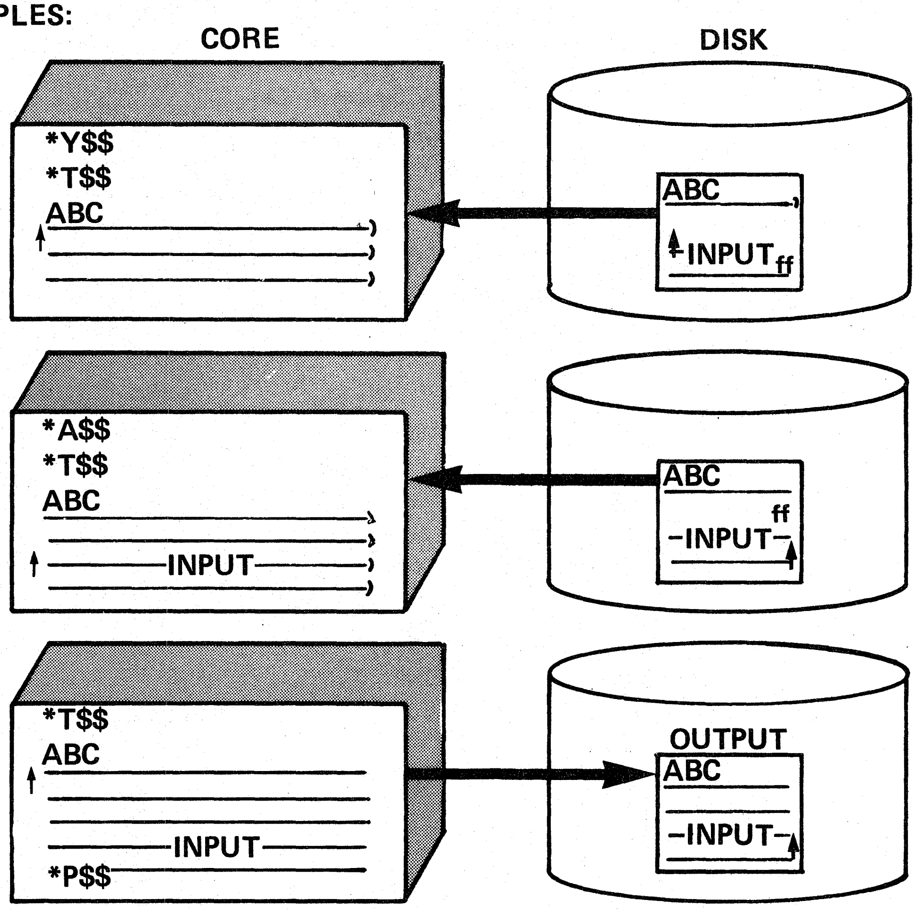
System Errors occur usually because the Association has already been done; or the file does not exist.

EDIT COMMANDS AND EXAMPLES

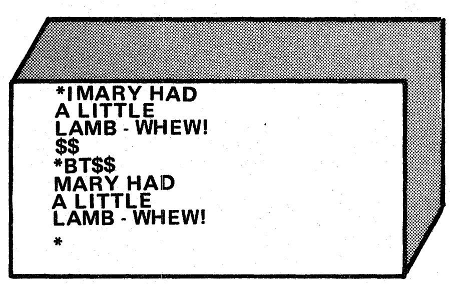
Input/Output Commands:

*Y$$	—	Yank the next page into the command buffer, overwrite previous data. Position CP to top of page.
*A$$	—	Append the next page to the bottom of the current page.
*P$$	—	Put the current page to the Output file.
*Itext$$	—	Insert the text following the "I" command from the current position of the CP.

EXAMPLES:



NOTE: PREVIOUSLY, A GWOUTPUT$$ WAS PERFORMED!



NOTE: THE *B$$ REPOSITIONS THE CHARACTER POINTER (CP) TO THE BEGINNING OF THE BUFFER.

EDIT COMMANDS AND EXAMPLES

Delete Commands

*nK$$	-	Kill n lines relative to the CP.
*nD$$	-	Delete n characters relative to the CP.

Examples:

```
 LINE 1
↑LINE 2                LINE 3
 LINE 3⇒ *2K$$⇒       ↑LINE 4
 LINE 4

A B C D  ⇒ *3D$$ ⇒     D
↑                     ↑
```

Character Pointer (CP) Positioning Commands:

*B$$	—	Reset CP to beginning of buffer
*Z$$	—	Reset CP to end of buffer
*nJ$$	—	Place CP at start of line n.
*nL$$	—	Move CP n lines from current CP position.
*nM$$	—	Move CP n characters from current CP position.
*CNTRL I	—	Move CP to the next TAB position. This is the TAB function, can be affected via TAB key.

Examples:

```
 LINE  1                LINE  1                LINE  1
 LINE  2               ↑LINE  2                LINE  2
 LINE  3↑⇒ *B$$ ⇒       LINE 3⇒   * Z$$ ⇒      LINE  3
 LINE  4                LINE  4                LINE  4

 LINE  1      3         LINE  1                LINE 1↑
↑LINE  2⇒  * 4J$$ ⇒     LINE 2⇒   *–2L$$ ⇒     LINE  2
 LINE  3                LINE  3               ↑LINE  3
 LINE  4                LINE  4                LINE  4
                        ↑

A B C D E ⇒ *2M$$ ⇒  A B C D E ⇒ * –4M$$ ⇒ A B C D E
   ↑                         ↑               ↑
*I↑I1↑I2$L1T$$      ⇒_ _ _ _ _ _ _1_ _ _ _ _ _ _2
                     7 spaces        7 spaces
```

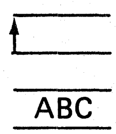
NOTE: The TAB character (↑I) is kept as octal 11, not actually spaces,
EDIT translates this on output

EDIT COMMANDS AND EXAMPLES

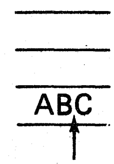
Search Command:

*Stext$$	–	The buffer is searched for text, CP is positioned immediately following the first occurrance of text. If text is not found a message results and CP is put to the buffer start.

Example:



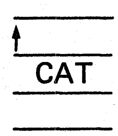
*SAB$$



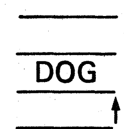
Change Command:

* Ctext1$text2$$	–	The buffer is searched for text1, text2 is substituted for text1 and CP is positioned immediately following.

Example:



*CCAT$DOG$$



Display Commands:

*nT$$	–	Type n lines from the position of the CP.
*U?$$	–	Display the Input & Output file associations
*.$$	–	Display the line number containing the CP.
*:$$	–	Display the total number of lines in the buffer.
*=$$	–	Display the total number of characters in the buffer.

EDIT COMMANDS AND EXAMPLES

Display Commands:

Examples:

ONCE		ONCE		
UPON	*T$$	UPON	*.$$	*B.$$
A ↑		A ↑	2	1
TIME		TIME	*	*

THERE				
↑ WAS	*1T$$	↑ THERE	*:$$	*=$$
A			4	14
SMALL			*	*

Macro Implementation:

*XMCOMMAND$$	–	Store COMMAND in the macro register.
*XD$$	–	Clear the macro register.
*X$$	–	Execute the COMMAND within the macro register
*X?$$	–	Display the contents of the macro register.

Examples:

LINE1		PAGE1
↑LINE2	*XMCLINE$PAGE$$	PAGE2
LINE3	*5 X$$	PAGE3
LINE4		PAGE4
LINE5		PAGE5
		↑

*X?$$

CLINE$PAGE$$

*XD$$

*X?$$

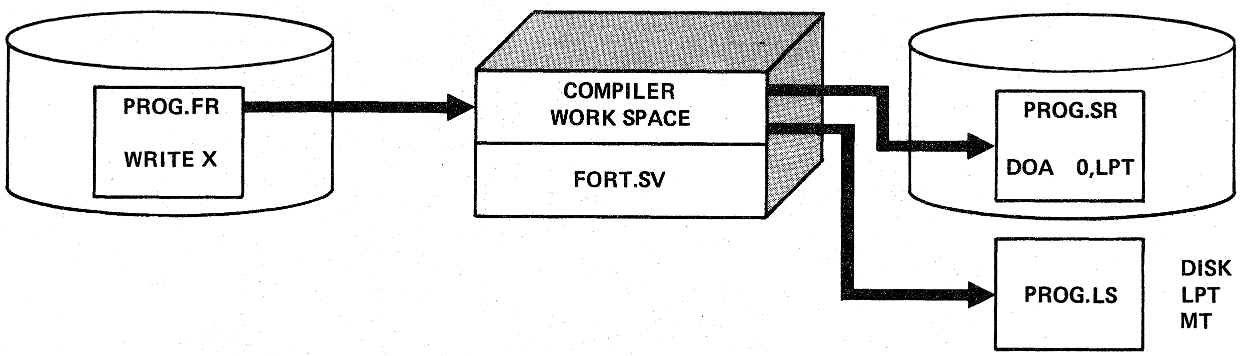
INCORRECT OR UNDEFINED MACRO

*

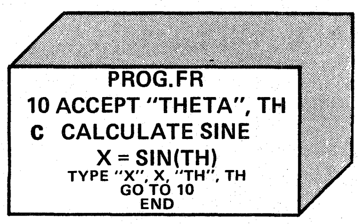
Watch Out***

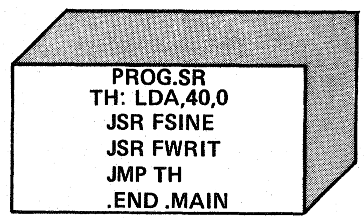
- As the Yank Command clears the buffer, be sure to output the current buffer first.
- Make sure all commands are entered after the asterisk prompt; if in Insert mode, commands are not executed, they're treated as data.

COMPILE PHASE – FORTRAN EXAMPLE



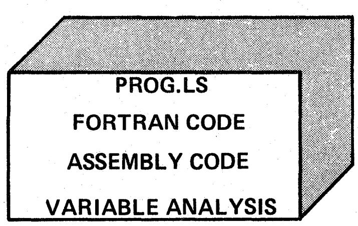
* A Compiler translates a source language into a more fundamental language.



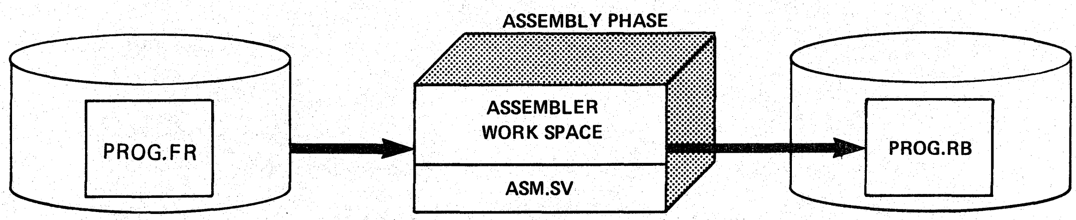


* Source files are translated into Assembly Language.

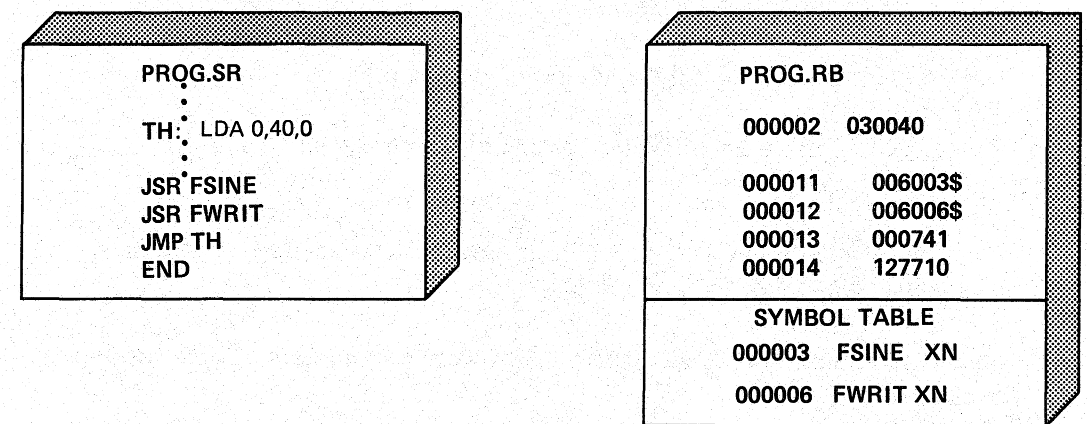
* Names generated, call Runtime Support Routines for desired functions.



* Errors are reported at the console and within the listing file.



* The Assembler generates relative binary instructions; their location is measured from the first instruction and modules referenced are unresolved.



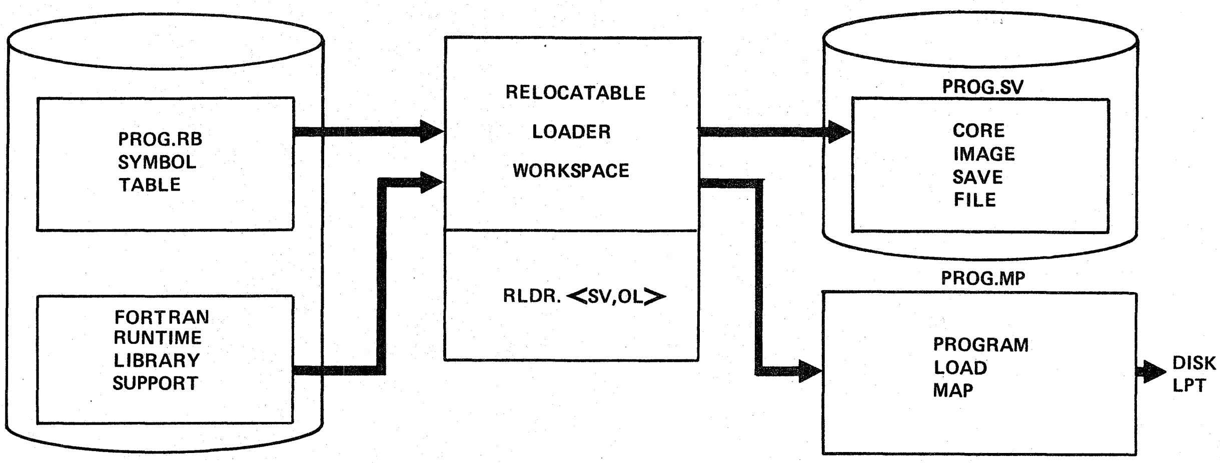
* The Assembly first pass translates symbology into binary.

* Backward references and self-contained instructions are completely resolved.

* Assembly second pass resolves internal forward references.

* All other unresolved references are installed in a symbol table by statement number and routine referenced.

LOAD PHASE



* The Relocatable Loader builds the Core Image Save File according to the RB file and Support Libraries.

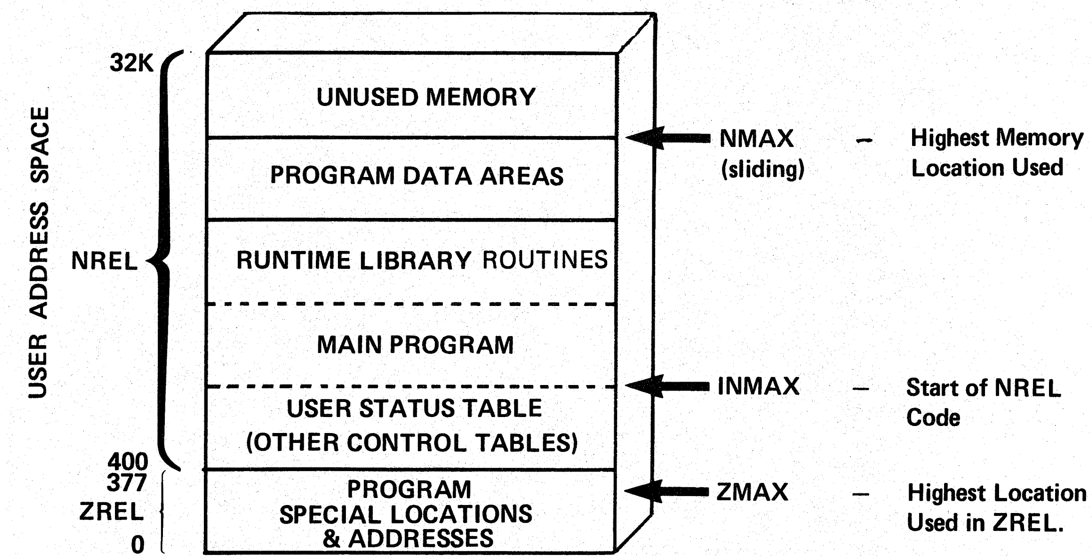
* The Program Code is first installed into the Save File.

* Library Modules are scanned and compared to the Symbol Table for load on reference.

* The Load Map records where in memory modules are loaded; additionally, unresolved references are flagged in error.

* Logical Errors are referenced to the Load Map for the module with functional maladies.

* Debugging Software can be loaded to interact with the running program for diagnosis of functional errors.



* The Relocatable Loader stacks information into NREL & ZREL memory areas.
* ZREL, accessable from any memory location, contains the addresses of the Runtime Library software.
* NREL is loaded with Control Tables, Program code, the Runtime Library Routines, and the Program Data Storage Area.
* RDOS controls the entire program via the User Status Table, it contains the amount of core usage, flags and some important pointers.

S200

RDOS USER

MODULE 9

OTHER RDOS EDITORS

MODULE 9

OBJECTIVES

OTHER RDOS EDITORS

Upon successful completion of this course you will be able to:

* DISTINGUISH BETWEEN THE APPROPRIATE USES OF THE REMAINING RDOS EDITORS

BINARY EDITORS

SYMBOLIC EDITOR : SEDIT.SV

* Single User, Single Location, Symbolic Editor
* Symbol Table may be used to specify file offsets

```
R
SEDIT PROG.SV
SEDIT REVISION X.X
.START + 10 / 105433 105427↲
.
.ESC Z
DONE
R
```

OCTAL EDITOR : OEDIT.SV

* Single User, Single Location, Octal Editor
* Locations specified via octal numeric offset

```
R
OEDIT PROG.SV
OEDIT REVISION X.X
. 472/105427 105433↲
.
.ESC Z
R
```

NOTE: All binary editors employ the predominantly same command set.

OTHER RDOS EDITORS

TEXT EDITORS

SUPER EDITOR : SPEED.SV

* Single User, Multibuffer Super Text Editor
* Global file associates allow I/O to any buffer, local file associations allow I/O to different files for each buffer.
* Buffers can hold macro definitions

```
R
SPEED FILENAME↵
!   —   all edit commands supported
!   —   additional super edit commands
!   —   buffer commands supported
!  H$$
R
```

MULTIEDITOR : MEDIT.SV

* Multiuser, Text Editor
* All EDIT commands available over multiplexor lines

```
R
MEDIT 16↵
.
.   —   edit command available at QTY:0 to QTY:15
↑A —    interrupt   MEDIT .SV
INT
R
```

OTHER RDOS EDITORS

LIBRARY FILE EDITOR : LFE.SV

* Single User, Single Scan, Library File Editor
* Edits the collection of .RB files, .LB

R		
LFE T FORT.LB	–	Type Module Names in FORT.LB
R		
LFE X RBNAME SYS.LB	–	Extract RBNAME from SYS.LB
R		
LFE M LIB/O <1,2,3,4>.LB	–	Merge libraries 1,2,3,4 into LIBRARY

S200

RDOS USER

MODULE 10

PROGRAMMING TECHNIQUES

TO

MANAGE MEMORY

MODULE 10

OBJECTIVES

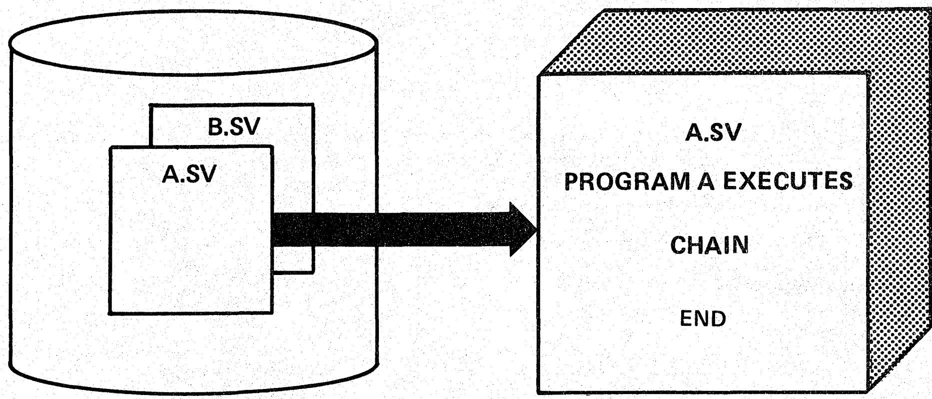
PROGRAMMING TECHNIQUES TO MANAGE MEMORY

Upon successful completion of this module you will be able to:

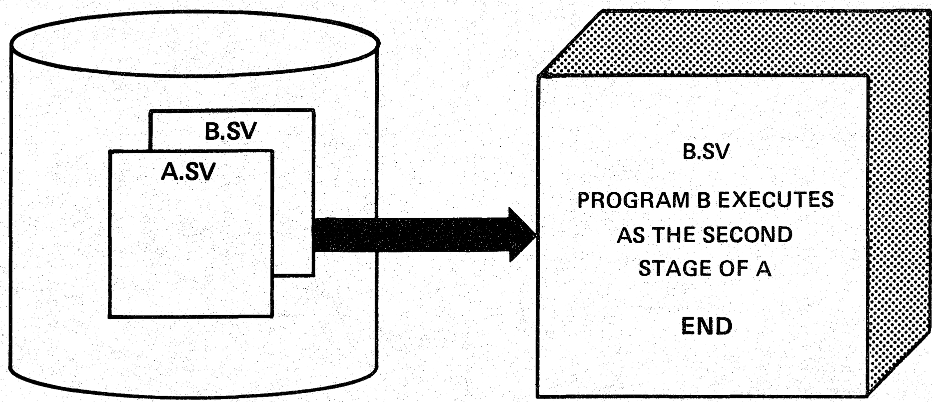
* DESCRIBE RDOS PROGRAM CONTROL OVER MEMORY
* DESCRIBE RDOS PROGRAM MEMORY MANAGEMENT TECHNIQUES

Ab Rev. 7.0 ist möglich Daten zu tauschen bis Speicherkapazität - 64 KByte

PROGRAMMING TECHNIQUES TO MANAGE MEMORY
CHAIN EXECUTION OF PROGRAMS



Program A is Invoked from CLI
R
A↵



Program B.SV Overwrites A.SV in Memory!

* Entire programs can be manipulated via the chain form of execution without special considerations in their programming.

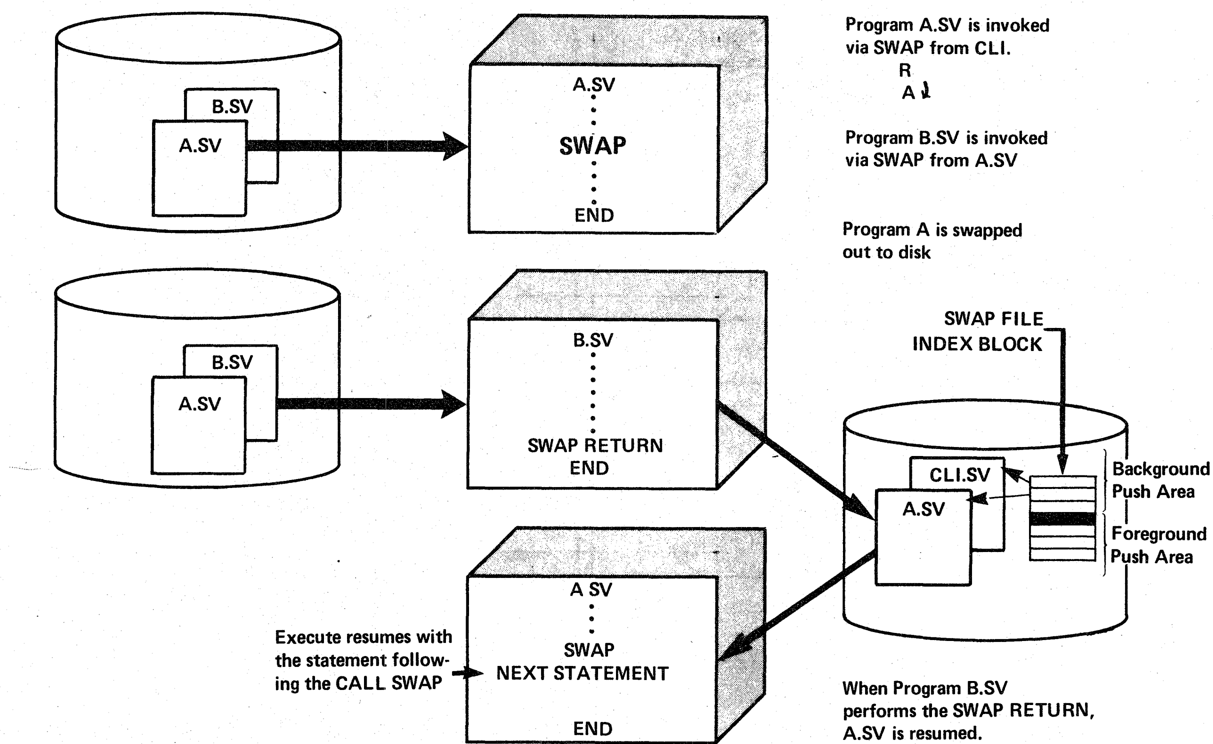
* The called program is loaded into memory destructively.

* Program Applications can be infinitly large if manipulated in a chained fashion.

* CLI can execute programs via chain:

R
CHAIN A↵

PROGRAMMING TECHNIQUES TO MANAGE MEMORY
SWAP EXECUTION OF PROGRAMS

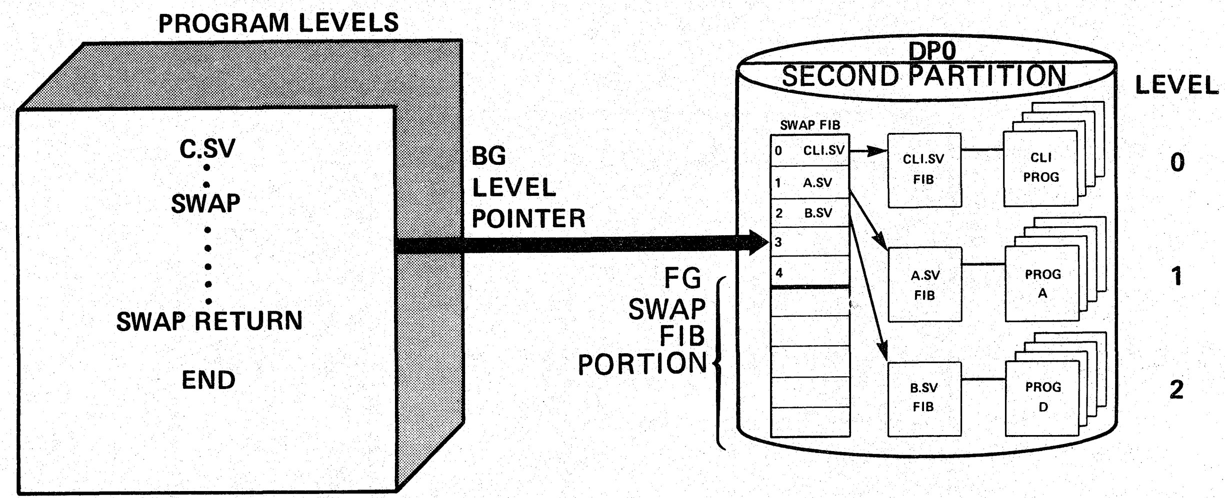


* Swap Execution of programs stores a snapshot of the executing program on disk prior to loading the called program.

* Programs are said to execute at a Program Level, when the swap is employed the next program level is used, CLI executes at level zero, programs may use levels 1 - 4.

PROGRAMMING TECHNIQUES TO MANAGE MEMORY

SWAP EXECUTION OF PROGRAMS



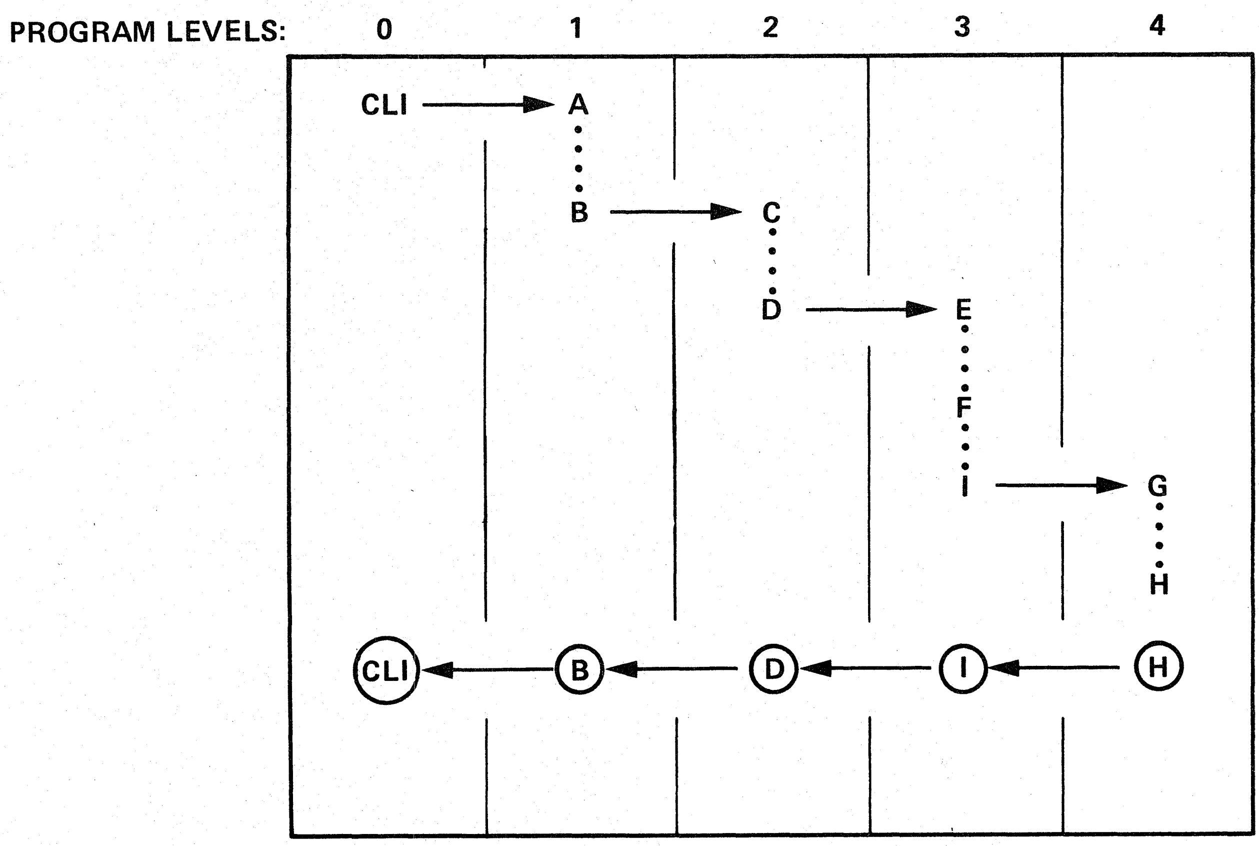
* Program Levels allow subordinate execution of entire programs; swap Returns always pass control back to the previous level.

* Partitions hold the Swap File Index Block which controls the core image snapshots.

* Foreground & Background share the Swap FIB and are each limited to levels 0 through 4.

* The Swap FIB: points to a File Index Block which points to the core image data blocks.

PROGRAMMING TECHNIQUES TO MANAGE MEMORY
SWAPS & CHAINS TOGETHER



* An unlimited number of programs can be executed; the greatest nested level is 4.

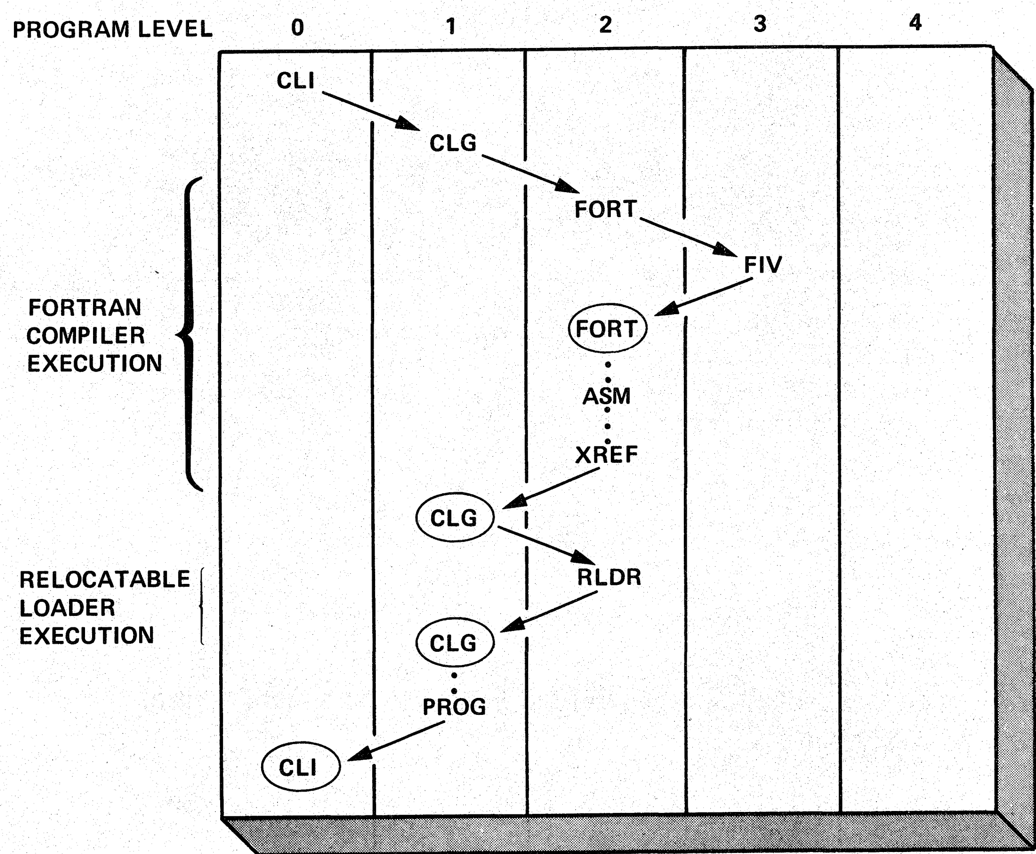
* RDOS searches the disk for CLI.SV, upon initialization, to execute at level zero.

* THE SWAP RETURN RESUMES THE PROGRAM SWAPPED FROM.

KEY
WHERE ⋮ : CHAIN
→ : SWAP
(B) : SAVED PROGRAM LEVEL

PROGRAMMING TECHNIQUES TO MANAGE MEMORY
SWAPS & CHAINS

EXAMPLE: COMPILE LOAD & GO PROGRAM CLG.SV

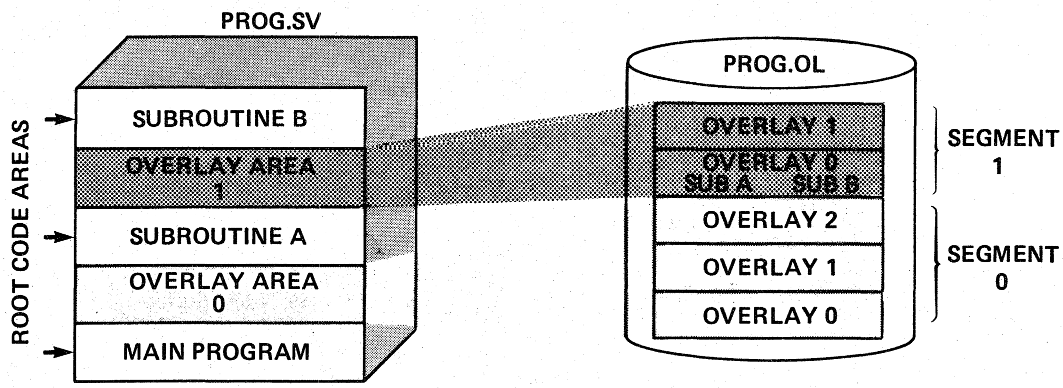


* Programming Structures may be incorporated into larger schemes providing level 4 is not exceeded.

* Communications between Levels is performed via common disk files. com.cm

PROGRAMMING TECHNIQUES TO MANAGE MEMORY

OVERLAYS



* Segments within the Overlay file are associated to core resident Overlay Areas.

* After the channel association to the Overlay File, overlays are loaded into the Overlay Areas by name.

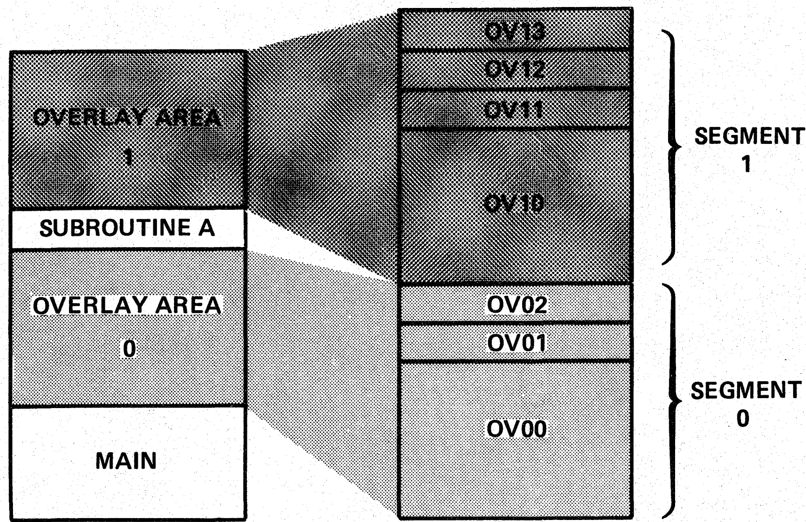
* The Overlay is a vehicle, to bring infrequently used subroutines into core; more than one subroutine may occupy the overlay.

* Overlays within a segment are each exclusively accessable.

* When the Overlay is resident, its subroutine may be called.

* Overlays are controlled by the Overlay Directory in core following the User Status Table.

PROGRAMMING TECHNIQUES TO MANAGE MEMORY
OVERLAY OPTIMIZATION



RLDR MAIN [OV00, OV01, OV02] SUBA↓

[OV10, OV11, OV12, OV13] LIBRARY ↓

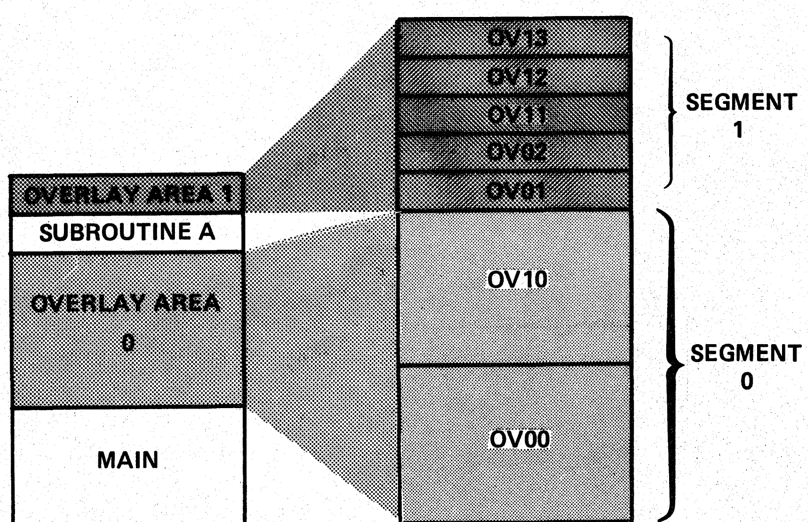
* Overlay Areas accomodate the largest Overlay with an integral number of blocks (256 words)

* Speed may be optimized, Overlay files are contiguous and block transfers are used to load overlays.

* Core Usage may be optimized by grouping similarly sized overlays together.

RLDR MAIN [OV00, OV10] SUBA

[OV0 < 1.2 >, OV1 < 1,2,3 >] LIBRARY



S200

RDOS USER

MODULE 11

SYSTEM INITIALIZATION

ON A

FORMATTED DISK

MODULE 11

OBJECTIVES

SYSTEM INITIALIZATION

ON A FORMATTED DISK

Upon successful completion of this module you will be able to:

* USE DKINIT TO INITIALIZE AN RDOS DISK
* MAKE AN RDOS DISK BOOT'ABLE
* INSTALL THE REMAINING RDOS SYSTEM SOFTWARE

SYSTEM INSTALLATION ON A FORMATTED DISK
RDOS STARTER SYSTEM ON MAG TAPE

SEQ OF EXEC	TAPE FILE #	PROGRAMS	FORMAT	PURPOSE
	0	Bereich 0 TBOOT.SV	XFER	Tape Bootstrap Program
3B	1	CLI.<SV,ER,OL> BOOTSYS.SV BOOT.SV RCLI.SV	DUMP	Archival Storage of Important Files
3	2	BOOTSYS.SV	XFER nur Inhalt der Daten	The Starter System ohne Header
3A	3	BOOTSYS.OL	DUMP	Starter System Overlay File
1	4	DKINIT.SV	XFER	Disk Initializer
2	5	BOOT.SV	XFER	Disk Bootstrap Program
4	6 *	RDOS UTILITIES	DUMP	The Remaining System Software
5	7 *	RDOS LIBRARIES	DUMP	

* 6,7 muß man selber mit LOAD laden

Daten Inhalt mit UFD (Header)

die 0-5 macht TBOOT.SV selbständig

* Files in the XFER format are executable from tape, they offer initializing features which must be accessable.

* DKINIT inspects the disk for bad blocks and builds the REMAP table.

* BOOT installs HIPBOOT on blocks 0 & 1, making the disk BOOT'able.

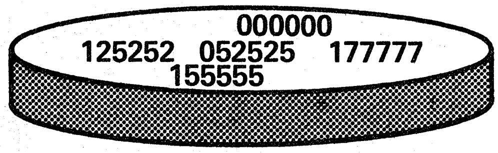
* BOOTSYS is the starter system, it loads MT0:3 & 1 thereby gaining control of the system via CLI.

* The remaining software comprises the RDOS system; it is loaded via CLI commands.

SYSTEM INSTALLATION ON A FORMATTED DISK

DISK INITIALIZER : DKINIT.SV

* Disk initializer installs a disk ID, Frame Size.
* A full initialization inspects the disk for bad blocks to build a REMAP Table.



FROM MT0: 4↵

DISK INITIALIZER – REV X.X

DISK DRIVE MODEL NUMBER ? 4234↵ –TOP LOADER 4047↵ – FRONT LOADER

DISK UNIT? DPO↵

COMMAND? FULL↵

DGC Model #	Disk Drive Type	Type In
6001-6008	Fixed-head (no cartridge)	6001) to 6008)
4047A, 4047B, 4237, 4238	Front-loading cartridge	4047) 4237) or 4238)
4234A	Top-loading cartridge	4234)
4048A	Top-loading pack (6 platters)	4048)
4057A	Top-loading pack (11 platters)	4057)
4231A	Top-loading pack (11 platters)	4231)

DGC Model #	Disk Drive Type	Type In*
6001-6008	Fixed-head (no cartridge)	DKØ)
4047A, 4047B, 4237, 4238	Front-loading cartridge	DPØ)
4234A	Top-loading cartridge RDOS on cartridge / RDOS on fixed disk	DPØ) / DPØF)
4048A	Top-loading pack (6 platters)	DPØ)
4057A	Top-loading pack (11 platters)	DPØ)
4231A	Top-loading pack (11 platters)	DPØ)

*disk identifying mnemonics

* Other commands allow further manipulation of bad blocks.

SYSTEM INSTALLATION ON A FORMATTED DISK

DKINIT COMMANDS

FULL — Performs the full initialization shown below

```
COMMAND DESTROYS ANY PREVIOUS RDOS DISK STRUCTURE
RDOS INIT/F MUST BE DONE ON DISK AFTER COMMAND
TYPE CONTROL-A NOW TO ABORT WITHOUT LOSS

NUMBER OF PATTERNS TO RUN (1-5)  5)

*** PATTERN #1 (125252) ***

*** PATTERN #2 (052525) ***

*** PATTERN #3 (155555) ***

*** PATTERN #4 (177777) ***

*** PATTERN #5 (000000) ***

*** ALL PATTERNS RUN ***

DO YOU WISH TO DECLARE ANY BLOCKS BAD
THAT ARE NOT ALREADY IN THE BAD BLOCK TABLE?  NO)

DEFAULT REMAP AREA SIZE IS 12 BLOCK(S) LONG
IT NEEDS TO BE AT LEAST 0 BLOCK(S) LONG

REMAP AREA SIZE (TYPE RETURN FOR DEFAULT)? )

REMAP AREA START BLOCK NUMBER (TYPE RETURN FOR DEFAULT)? )

DEFAULT FRAME SIZE IS 37,
MIN IS 1, AND MAX IS 406

DISK FRAME SIZE (TYPE RETURN FOR DEFAULT)? )

FULL DISK INIT COMPLETE

COMMAND?  STOP)
```

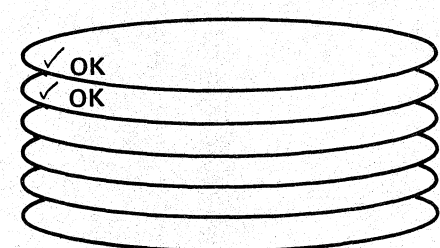
DKINIT.SV DIALOGUE

PARTIAL — Interrogates each block for validity and reports bad blocks, disk contents is maintained.

ENTER — Incorporates additional bad blocks into the REMAP table.

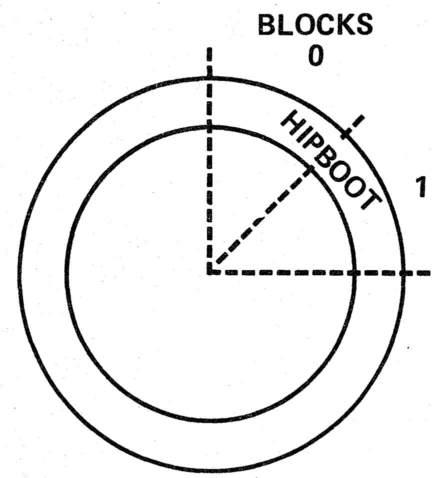
LIST — Displays disk status, frame size, and lists bad blocks within the REMAP area.

STOP — Halts DKINIT, rehomes disk heads.



SYSTEM INSTALLATION ON A FORMATTED DISK

INSTALLATION OF RDOS SOFTWARE



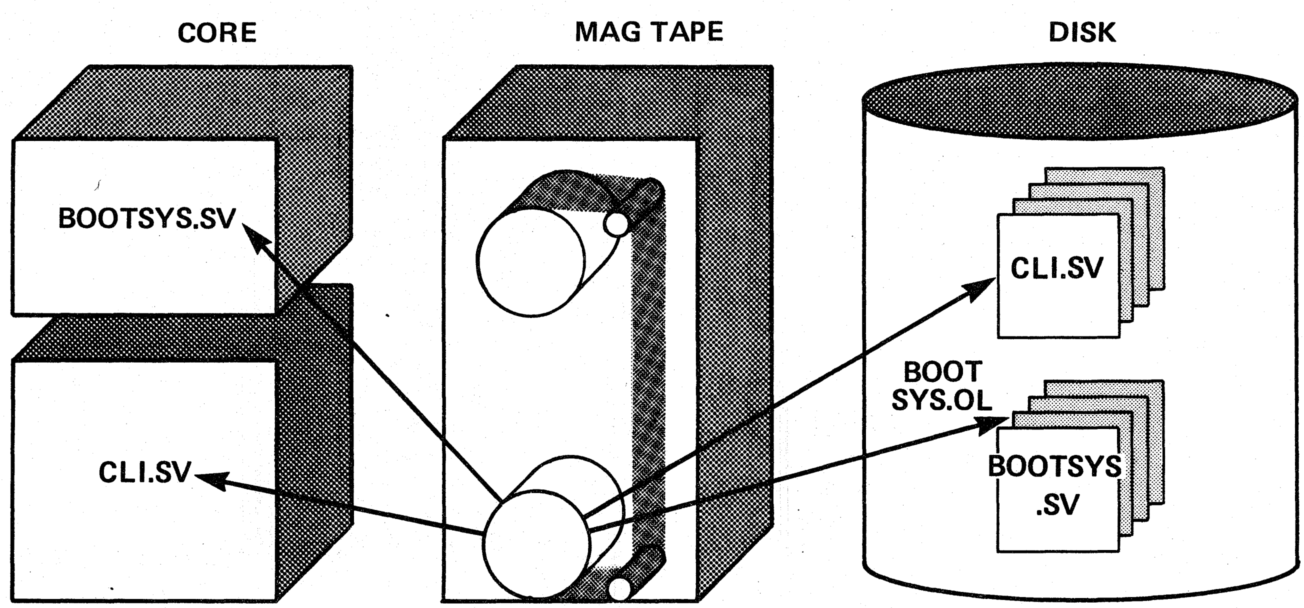
FROM MT0: 5 ↵
BOOTSTRAP DEVICE SPECIFIER? DP0 ↵
INSTALL BOOTSTRAP (Y OR N)? Y ↵

HALT COMPUTER

BOOTSYS.SV: Starter System

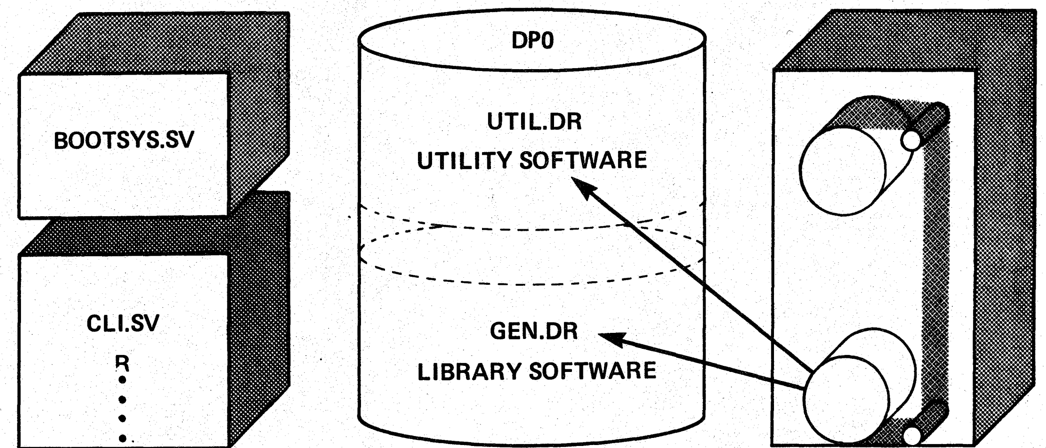
FROM MT0: 2 ↵
FULL (F) OR PARTIAL (P OR <CR>) ? F ↵
INITIALIZING WHAT DISK DP0 ↵

DATE (M/D/Y) 6/7/79 ↵
TIME (H:M:S) 11:35:00 ↵
R



SYSTEM INSTALLATION ON A FORMATTED DISK

INSTALLATION OF THE RDOS SOFTWARE



* The remaining software is loaded via CLI control.
* A utility directory holds the utility software on MT0:6.

```
R
INIT MT0↲
R
(CDIR,DIR) UTIL↲
R
LOAD/A/V  MT0:6↲
R
RELEASE UTIL↲
R
```

* The system generation software resides in a GEN directory; additional utilities from MT0:7 will aid generation.

```
R
(CDIR,DIR) GEN↲
R
LOAD/A/V MT0:6*SYSGEN.SV   RLDR.<SV,OL>↲
R
LOAD/A/V  MT0:7↲
R
```

NOTE: BOOTSYS.SV is not a large RDOS System, it affords no line printer, nor more than one directory initialized. Links cannot be resolved to alternate directories; therefore, the SYSGEN & RLDR files should be put into the GEN directory.

S200

RDOS USER

MODULE 12

SYSTEM GENERATION

MODULE 12

OBJECTIVES

SYSTEM GENERATION

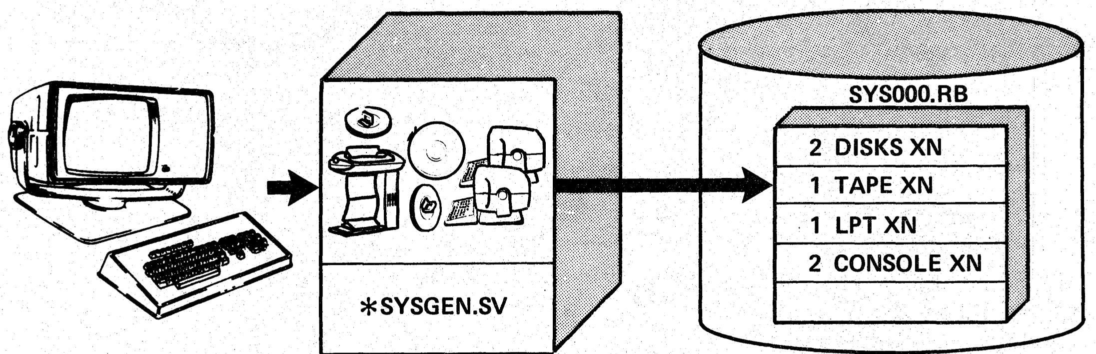
Upon successful completion of this module you will be able to:

* LIST THE SOFTWARE REQUIREMENTS TO SUPPORT SYSTEM GENERATION
* GENERATE AN RDOS SYSTEM APPROPRIATE FOR ANY USER HARDWARE
* GIVEN TUNING FILE OUTPUT DETERMINE THE APPROPRIATE SPECIFICATION OF RDOS SYSTEM COMPONENTS

SYSTEM GENERATION

SYSGEN EXECUTION

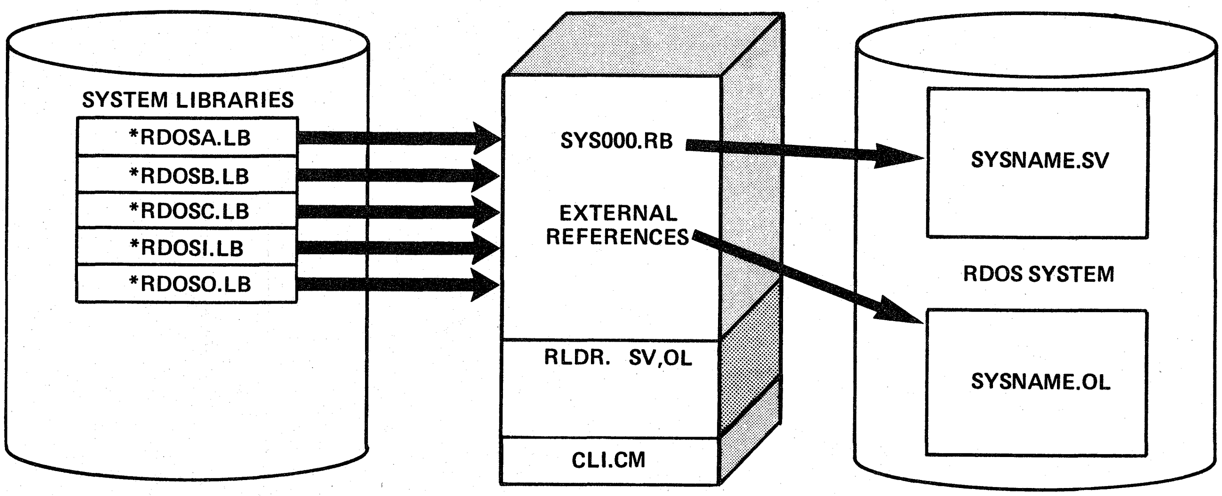
* The ✻SYSGEN program asks the user what to incorporate into the system.
* Affirmative responses cause external references to be loaded into an .RB trigger file : SYS000.RB.



* SYS000.RB is loaded with the RDOS libraries to produce a system save and overlay file.

SYSTEM GENERATION

LOAD PHASE



- The system load is conducted via an RLDR command line which SYSGEN, writes into the file CLI.CM

- The asterisk denotes the system flavor: * Z, A, B, M, U, N

ZRDOS <A, B, C, I, O> .LB	–	Large Mapped Eclipse
ARDOS <A, B, C, I, O >.LB	–	Small Mapped Eclipse
BRDOS <A, B, C,I, O> .LB	–	Unmapped Eclipse
MRDOS < A, B, C, I, O > LB	–	Mapped Nova System
URDOS < A, B, C, I, O >.LB	–	Unmapped Nova System
NRDOS < A, B, C, I, O >.LB	–	**Mapped Nova 3/4 System**

SYSTEM GENERATION

MECHANICS

* The SYSGEN Command:

*SYSGEN/N SYSNAME. </S SG/V LM/L>
/N – Halts sysgen prior to load to adjust the CLI.CM load line.

* The CLI.CM load line:

RLDR/Y/N/P SYS000 SYSNAME.SV/S^
*RDOSA.LB *RDOSB.LB BADSP ALMSPD *RDOSC.LB *RDOSI.LB^
[*RDOSO.LB] SYSNAME.LM/L ^
DELETE BADSP.RB SYS000.RB

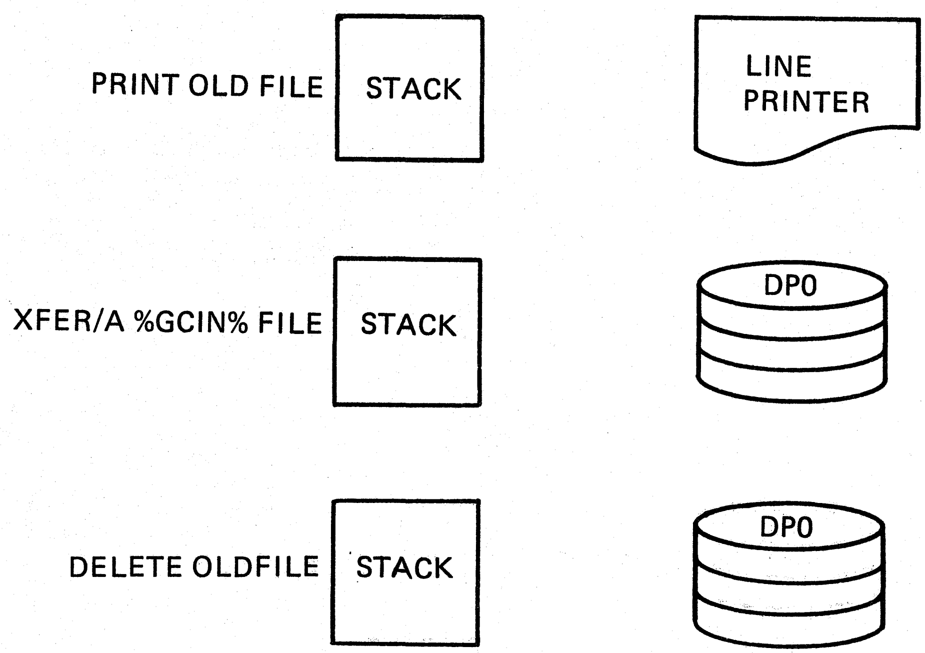
* The System Files Generated

SYSNAME.SV	–	The core resident, executable file
SYSNAME.OL	–	The system overlay file
SYSNAME.LM	–	The system load map
SYSNAME.SG	–	The system SYSGEN dialogue file

SYSTEM GENERATION
SYSTEM COMPONENTS

*THE SYSTEM STACK: ALLOWS FOR CONCURRENT SYSTEM TASKS

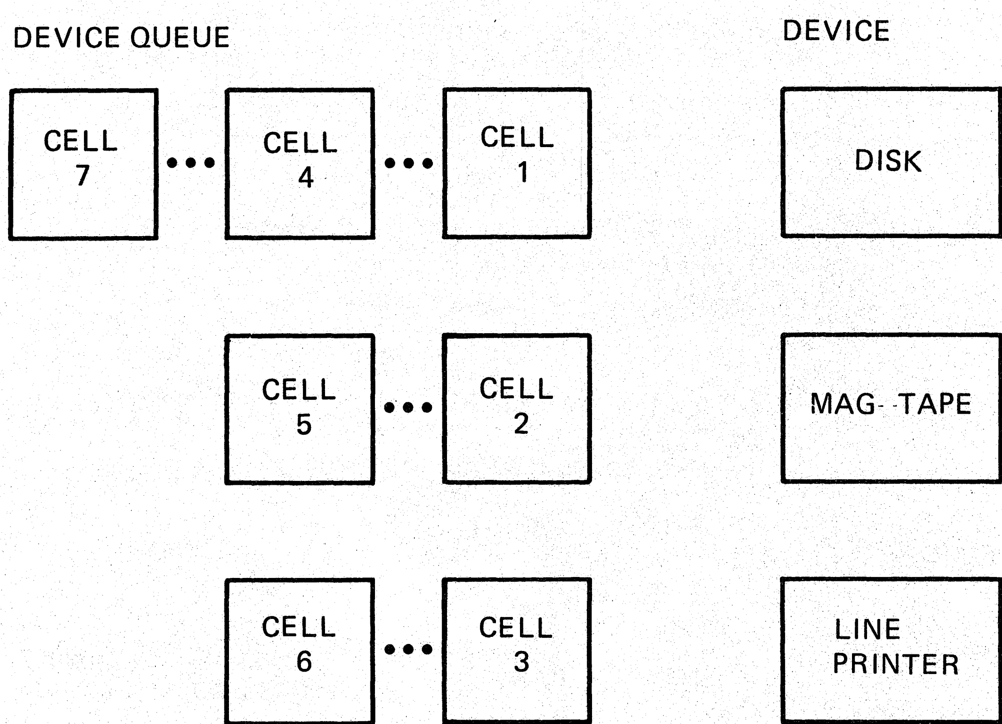
STACKS USED FOR: DISK I/O
SPOOLING
SYSTEM CALLS



SYSTEM COMPONENTS

* CELLS

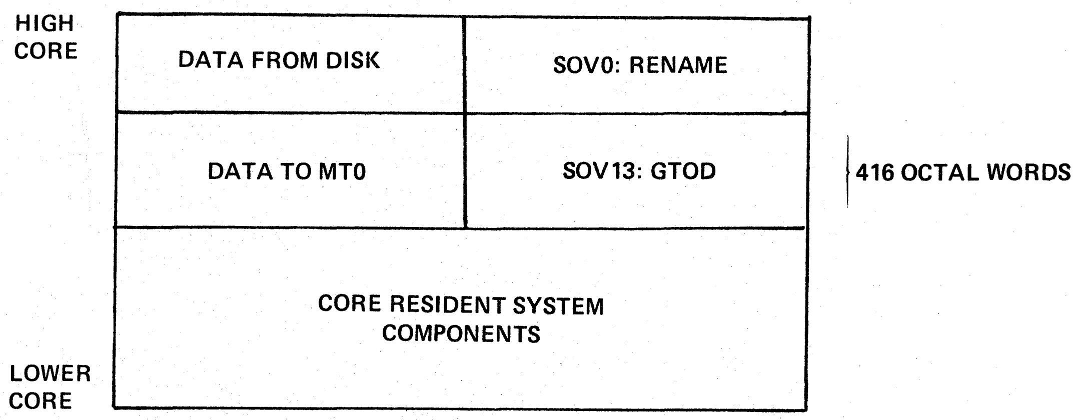
- 16 words long
- hold system task information
- 1 cell for each active system call
- 2 cells for each active spool request
- SYSGEN allocates minimum of 3 cells per system stack



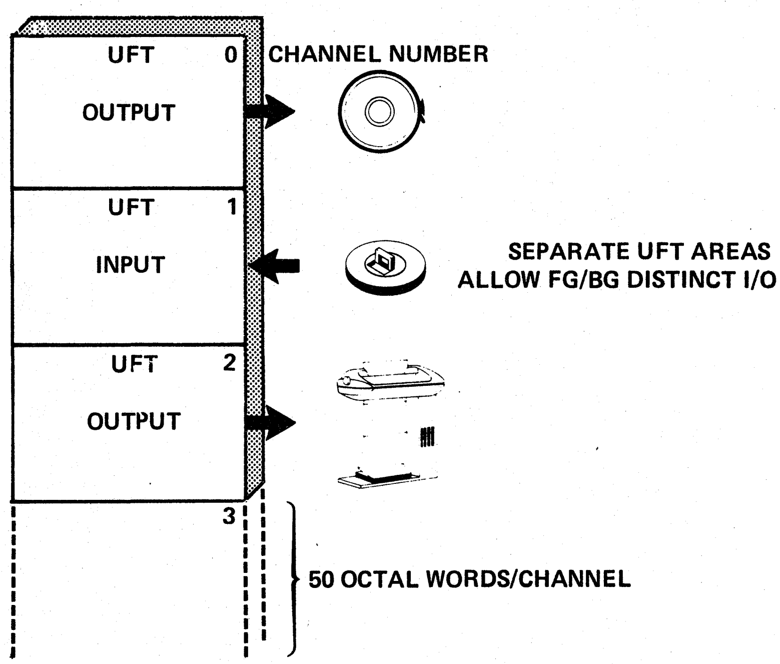
SYSTEM GENERATION

SYSTEM COMPONENTS

* Buffers hold System Overlay Code and Block Oriented Data



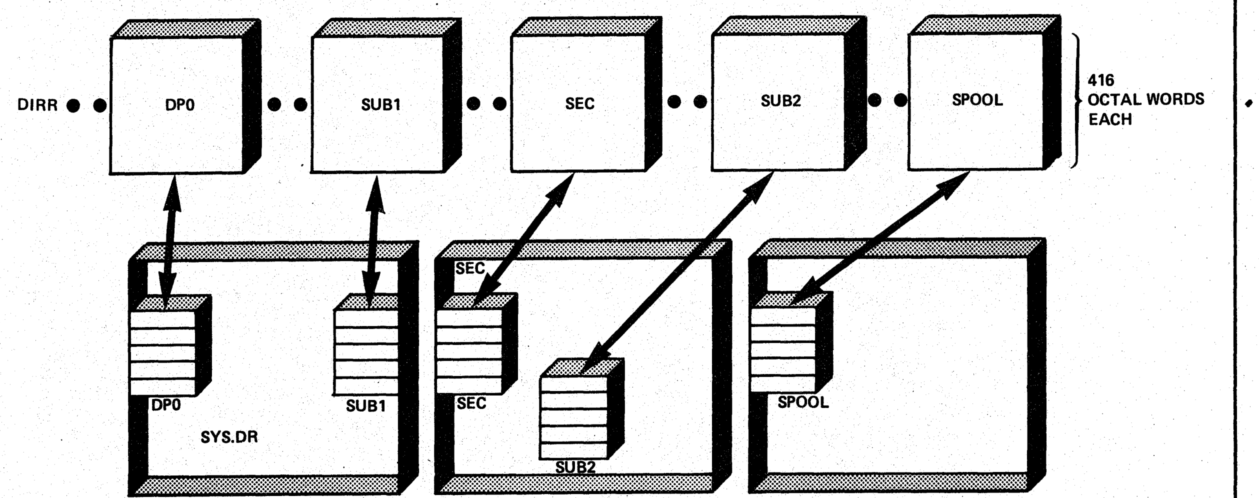
* User File Tables: UFT—Control Distinct I/O Transport
  Mapped—UFT's are in the operating system
  Unmapped—UFT'S are in User Address Space



SYSTEM GENERATION

SYSTEM COMPONENT

* System File Device Control Blocks (DCB) coordinate initialized directories.



* Other Core Resident Components

Scheduler: Decides which system task is ready to execute.
System Call Processor: Assigns each cell to a system task
Drivers & Service Routines: Manipulate devices & respond to interrupts
Interrupt Handler: Save machine state and pass control to service routines.
Overlays: Vehicles for the call logic.

S200

RDOS USER

MODULE 13

SYSTEM UPDATES

PATCH FACILITIES

MODULE 13

OBJECTIVES

SYSTEM UPDATES

PATCH FACILITIES

Upon successful completion of this module you will be able to:

* PERFORM UPDATES TO ANY RDOS SYSTEM
* USE THE ENPAT UTILITY TO CREATE PATCH DATA
* USE THE PATCH UTILITY TO PATCH ANY PROGRAM OR OPERATING SYSTEM

SYSTEM UPDATES/PATCH FACILITIES

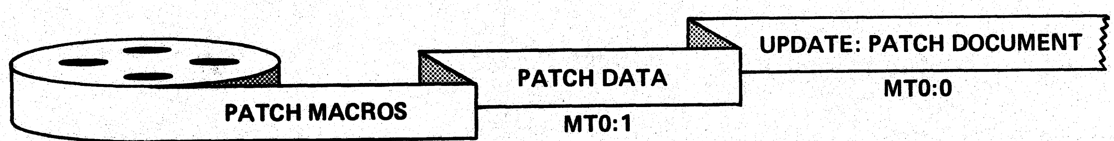
RDOS REVISION 19.84

Update Number

Minor Revision Number

Major Revision Number

* The Update Tape provides documentation, patch data, and macro files for easily implemented updates.



SYSTEM UPDATES & PATCH FACILITIES

PATCH & ENPAT

A patch is a one word change to a save or overlay file.

* ENPAT.SV creates the patch data
* An interactive dialogue creates an entry for each patch:

S	10422	177777	177723	ALPHA
SAVE OR OVERLAY FILE	PATCH LOCATION	OLD CONTENTS	NEW CONTENTS	Conditional Symbol, Apply patch only if ALPHA appears within the load map

* PATCH.SV installs the patch data into a save or overlay file
* PATCH execution:

PATCH SYSNAME.SV/S	SYSNAMELM/L	PATCH FILE.PF/P)
SAVE FILE TO PATCH	LOAD MAP FILE FOR CONDITIONAL PATCHES	PATCH DATA FILE

SYSTEM UPDATES/PATCH FACILITIES
PERFORMING THE UPDATE

* Creation of the UPDATE directory:

```
R
(CDIR,DIR) UPDATE
R
```

* Loading of the update software:

```
R
INIT MT0
R
LOAD/A/V MT0:(0,1)
R
PRINT UPDATE
R
```

* LINK'ing the update files & utilities

```
R
LINK <,UTIL: >PATCH.SV
R
LINK <,GEN: >SYSNAME.(SV,OL)
R
```

SYSTEM UPDATES/PATCH FACILITIES
PERFORMING THE UPDATE

* Performing an Update:

  - via macro:

    R
    LINK <,GEN: > BSYSGEN.SV
    R
    BSGENPATCH – Installs patches

  - via patch utility:

    R
    PATCH SYSNAME/S SYSNAME.LM/L ARDOS.PF/P
    :
    : ———— Installation of Patches
    R

S200

RDOS USER

MODULE 14

MONITORING AN RDOS SYSTEM

MODULE 14

OBJECTIVES

MONITORING AN RDOS SYSTEM

Upon successful completion of this module you will be able to:

* GENERATE AND EFFECTIVELY EMPLOY TUNING UNDER RDOS
* ASSESS AND OPTIMIZE THE RESOURCE ALLOCATION WITHIN AN RDOS SYSTEM

MONITORING AN RDOS SYSTEM

TUNING

- Tuning measures the systems usage of stacks, cells, and buffers.
- Tuning is a SYSGEN option:
  TUNING? ("0" = NO, "1" = YES)
- The Tuning report shows the total number generated, the total number of requests, and a percentage failure rate for stacks, cells, and buffers. Additionally, a buffer itemization may be requested according to overlays used.
- Tuning report data is recorded with the file SYSNAME.TU. CLI commands control and report tuning data using the system tuning file.
- A properly generated system suffers a 5% failure rate on all software resources. Typical numbers are shown below:

16 channels/ground	8 stacks
8 buffers	12 cells

NOTE: 16 channels required for CLI

MONITORING AN RDOS SYSTEM

CLI TUNING MECHANICS

- To initiate the capture of tuning data:

  R
  TUON
  R

- To halt tuning:

  R
  TUOFF
  R

- To obtain a tuning file report:

  R
  TPRINT/L/O SYSNAME
  R

L — produce a line printer listing
O — include an overlay report.

```
R
TUON TUSYS
R
MESSAGE COMMANDS FOLLOWING USE RDOS SOFTWARE FACILITIES
COMMANDS FOLLOWING USE RDOS SOFTWARE  FACILITIES
R
GMEM
BG: 28    FG:  29
R
SMEM 16
R
GMEM
BG:  16   FG:  41
R
LIST/E/A TUSYS.–
TUSYS.OL                                    GEN:TUSYS.OL
TUSYS.TU             1536                  C             06/22/79 11:23 06/22/79 [004516]    1
TUSYS.SV                                    GEN:TUSYS.SV
R
MESSAGE NOTE THE TUNING FILE IS CONTIGUOUS – FASTER ACCESS
NOTE THE TUNING FILE IS CONTIGUOUS – FASTER ACCESS
R
TPRINT/L/O TUSYS
R
TUOFF
R
ENDLOG
```

The LOG.CM file above demonstrates tuning implementation on the RDOS system.

Pages following contain the tuning file printout.

S200

RDOS USER

MODULE 15

SYSTEM BACKUP: STARTER TAPE EMULATION

MODULE 15

OBJECTIVES

SYSTEM BACKUP: STARTER TAPE EMULATION

Upon successful completion of this module you will be able to:

* EMPLOY VARIOUS TRANSFER COMMANDS TO EFFECT BACKUPS TO TAPE OR DISK
* CREATE A TAPE BACKUP MACRO

SYSTEM BACKUP: STARTER TAPE EMULATION

TRANSFER MECHANICS

XFER:

File Contents Transported Only
One Disk File Per Command
Source Files→Destination File.
Tape Files are Boot'able

Examples:

Transfers from disk to tape; the first argument is the source file, the second argument is the destination file:

```
XFER TBOOT.SV MT0:0
XFER/A BOOTSYS.SV  MT0:2
XFER  DKINIT.SV  MT0:4
XFER BOOT.SV  MT0:5
```

Transfers from tape to disk

```
XFER  MT0:1   COPY.SV/R
XFER  MT0:3   MYFILE/C
XFER  MT4:4   KATHY
XFER  MT0:4   DKINIT.SV/R
```

SYSTEM BACKUP : STARTER TAPE EMULATION

TRANSFER MECHANICS

DUMP/LOAD:

UFD and contents transported
many disk files/tape file
directory structure maintained
tape file not BOOT'ABLE

EXAMPLES:

All Files in a Directory Dumped
DIR UTIL; DUMP/A/V MT0:6

All Files in a Partition Dumped
DIR PART; DUMP/A/L MT0:7

Certain Files can be Dumped
DUMP/A/V MT0:1 CLI.–, BOOTSYS.SV, BOOT.SV

The Entire Disk Structure may be LOADed
GDIR; LOAD/A/L MT0:0
DPO
R

SYSTEM BACKUP : STARTER TAPE EMULATION

TRANSFER MECHANICS

Die beide Laufen nicht bei BOOTSYS.SV

FDUMP/FLOAD:

Three mag tape files/command all files in
All files in current directory transported
Fastest backup method
Most condensed new tape volume cantrols

EXAMPLES:

Multiple copies of the disk 012 3,4,5
GDIR; FDUMP/L MT0: (0, 3, 6) 6,78
DISK TO TAPE TRANSFER
DPO

Multiple volume controls
FDUMP/L MT0:36
DISK TO TAPE DUMP
END OF TAPE
MOUNT NEXT REEL – STRIKE KEY WHEN READY
.
.
.

FILENAME

BURST /DUMP
/LOAD
/DUPLICATE
/VERIFY

MTØ:Ø (Default)
MTØ:2

T BOOT.SV	MTO:0
BURST.SV	MTO:1
	MTO:2

SYSTEM BACKUP : STARTER TAPE EMULATION

TRANSFER MECHANICS

MOVE:

Directory to Directory transport UFD and
UFD and file contents transferred specifier
Must include a directory specifier
May use filename templates

EXAMPLES:

Move all save and overlay files to the alternate directory.

```
MOVE/A/V  ALTERNATE   -.SV,   -.OL
```

Move all files, not links, only move recent files

```
MOVE/A/V/K/R  UTIL
```

Make a copy of DP0 using DP0F

```
MOVE/A/V   DP0F
.
.                     load & boot another pack
.
DIR DP0F
R
MOVE/A/V   DP0                 a copy of the original DP0.
```

SYSTEM BACKUP : STARTER TAPE EMULATION

A TAPE BACKUP MACRO

```
MESSAGE "BACKUP IN PROGRESS"
\
\ STARTER TAPE STANDALONE FACILITIES
XFER TBOOT.SV  MT0:0
DUMP/A MT0:1 CLI .<SV,ER,OL> , BOOTSYS.SV,BOOT.SV
XFER BOOTSYS.SV MT0:2
DUMP/A MT0:3 BOOTSYS.OL
XFER DKINIT.SV MT0:4
XFER BOOT.SV MT0:5
\
\ TWO COPIES USER SOFTWARE – PARITY PROTECTION
DUMP/A/L  MT0: (6,7)
MESSAGE "BACKUP COMPLETE"
```

S200

RDOS USER

MODULE 16

RDOS SPOOLING

MODULE 16

OBJECTIVES

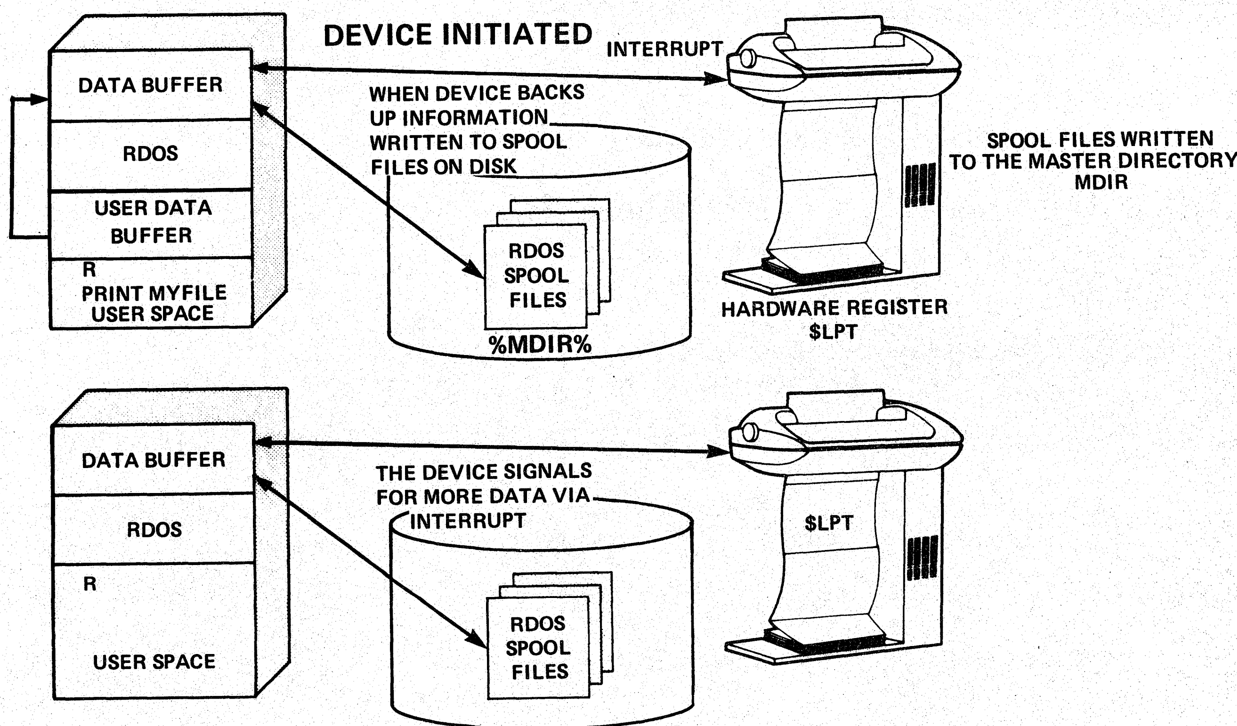
RDOS SPOOLING

Upon successful completion of this module you will be able to:

* DEFINE METHODS TO CONTROL RDOS SPOOLING
* RECOVER LOST SPOOL BLOCKS

RDOS SPOOLING

Spooling is a process to optimize CPU usage during an output request to a slow device. The device is started and data passed to it; upon backup, information is written to disk in a temporary file. When the device can output more data, it signals the CPU via an interrupt.



RDOS SPOOLING

Although Spool Control is limited, its optimizing effect is remarkable.

Spooling is enabled by default; to disable spooling:

R
SPDIS devicename
R

To Reenable Spooling:

R
SPEBL devicename
R

To kill a spool train of data to a device:

R
SPKILL devicename
R

The following are spoolable devices:

$DP0 $LPT(1) $PTP(1) $TTO(1) $TTP(1)

MCA
Dual Processor
Output

RDOS SPOOLING

SPOOL File Loss & Recovery

If the system crashes during spooling, spool file blocks are left in use; the spool files are maintained by logical block address in resident RDOS and may not be deleted through normal means.

Recovery Techniques

Initialization After Backup

- Mag Tape will contain only files in the directory structure
- DKINIT destroys all previous block information
- The Back Up Procedure recreates an optimum directory structure. (Max contiguous space preserved)

RDOS SPOOLING

Spool File Loss & Recovery

Boot the System from a Secondary Partition

- Master Directory is Secondary Partition
- Spool Files Localized to Secondary Partition
- Partition may be deleted to dispense with Spool Files.

Secondary Partition Bootstrap

- Links to Operating System must be available
  LINK <,GEN: > MYSYS.(SV,OL)
- CLI files must exist physically
  MOVE/A/V SPOOLPART CLI. < SV, ER, OL>
- Boot Program will Accept a Partition Specifier
  FILENAME? SPOOLPART:MYSYS

S200

RDOS USER

MODULE 17

RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND

MODULE 17

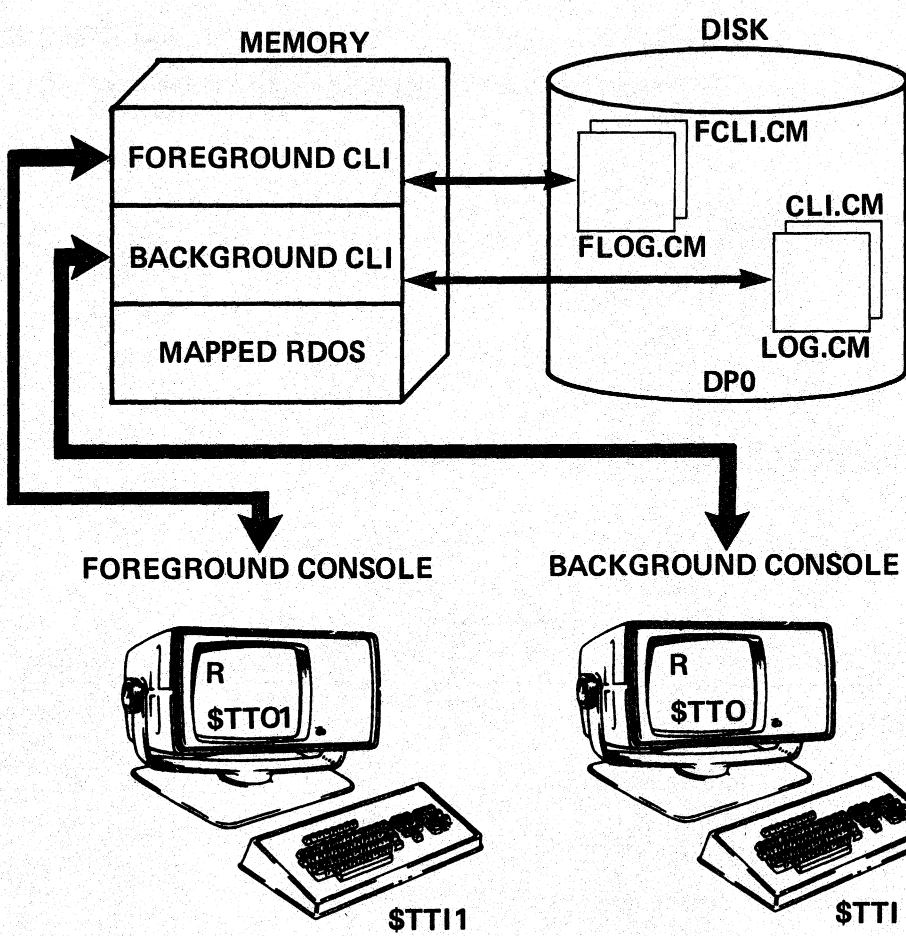
OBJECTIVES

RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND

Upon successful completion of this module you will be able to:

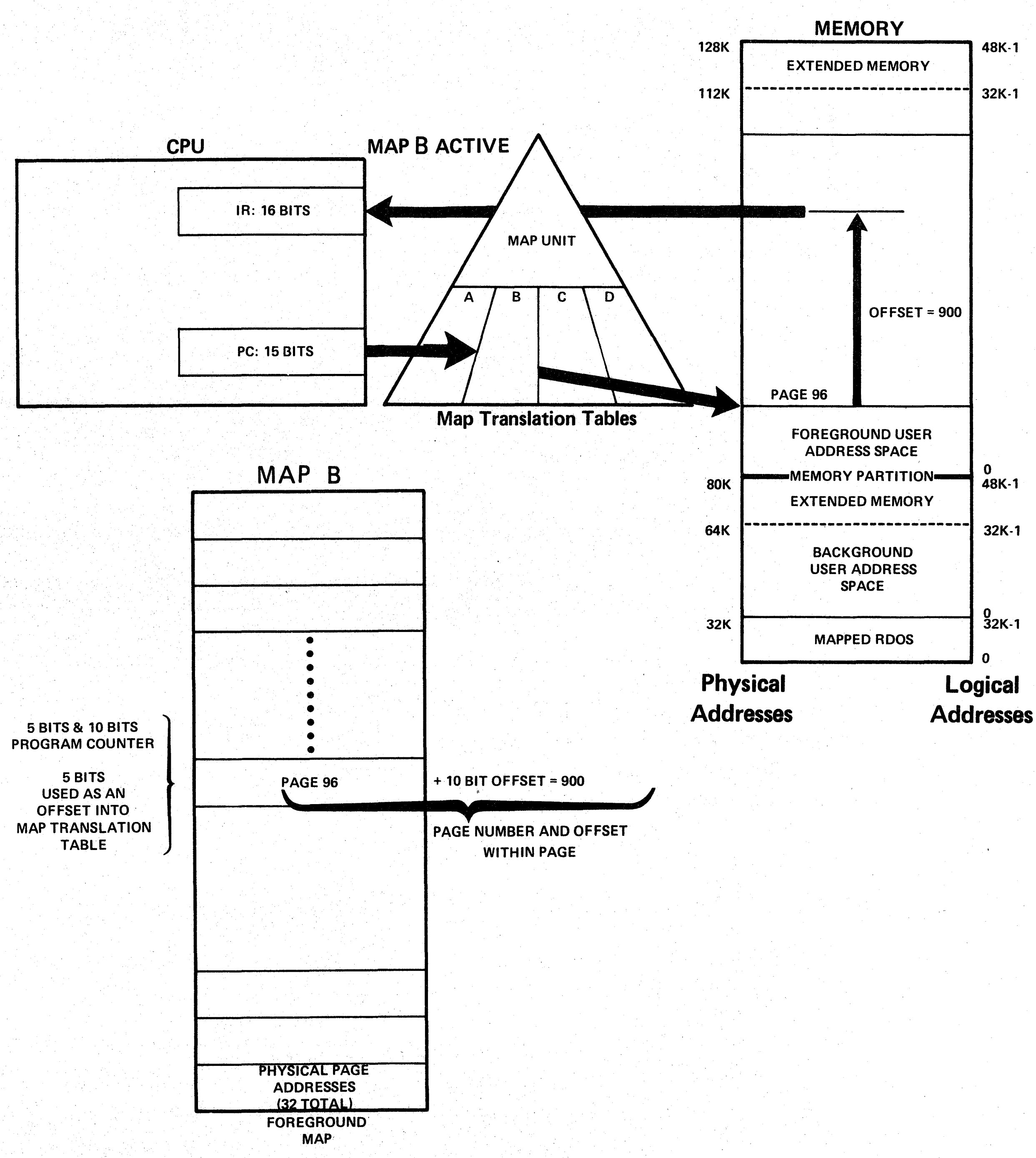
* CONTROL THE MAP UNIT TO ALLOCATE MEMORY FOR RDOS'S DUAL PROCESS MANAGEMENT
* IMPLOY APPROPRIATE CLI COMMANDS TO EXECUTE PROGRAMS IN THE FOREGROUND/BACKGROUND

RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND



- Foreground is a separate User Program in High Memory
- Background is a separate User Program in Lower Memory
- Each Communicate via CLI to their respective consoles.
- RDOS preserves all facilities for both programs

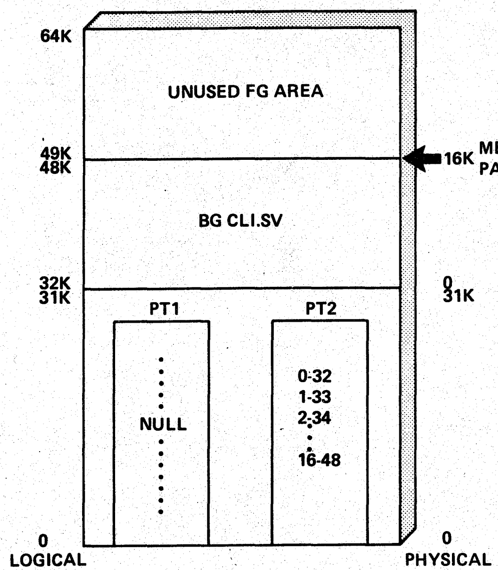
RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND MAP UNIT ADDRESS TRANSLATION

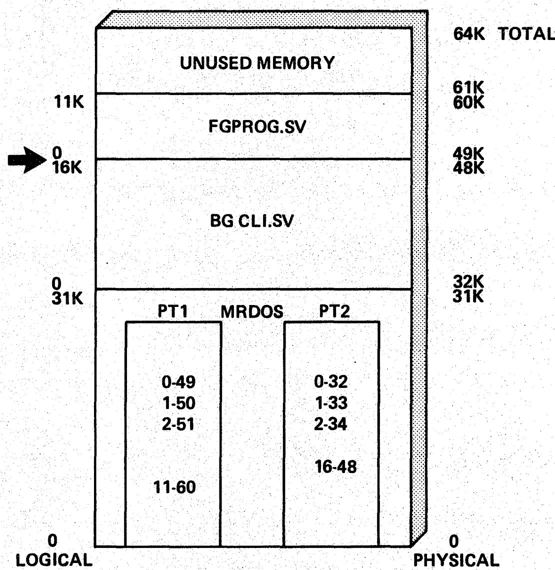


RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND

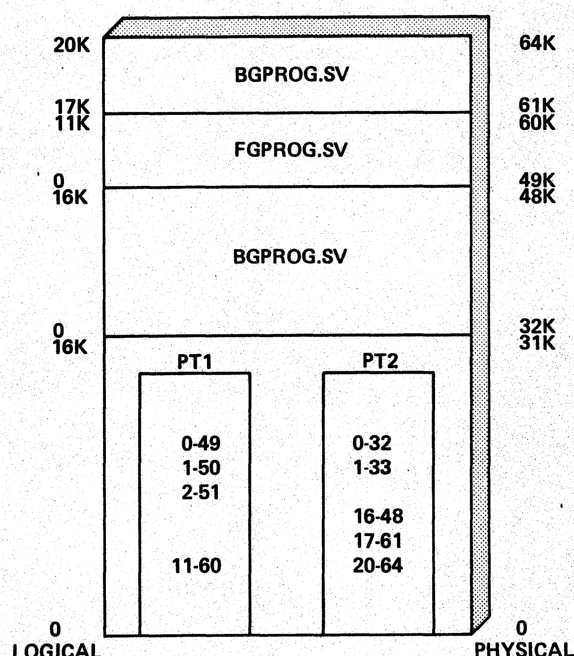
RDOS TRANSLATION TABLE MANIPULATION

- Two resident Program Tables record and control page allocation
- Initially, CLI occupies background space, foreground is unallocated.
- Under background control, a foreground program is executed.

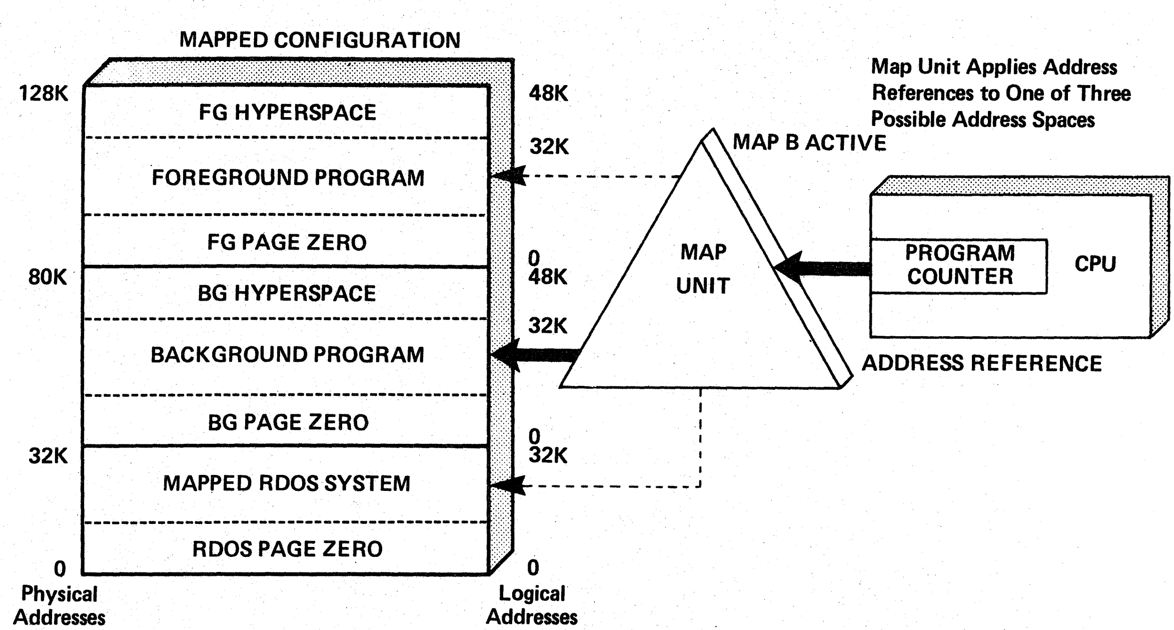




- If a 20K program is swapped to BACKGROUND unused pages are allocated.

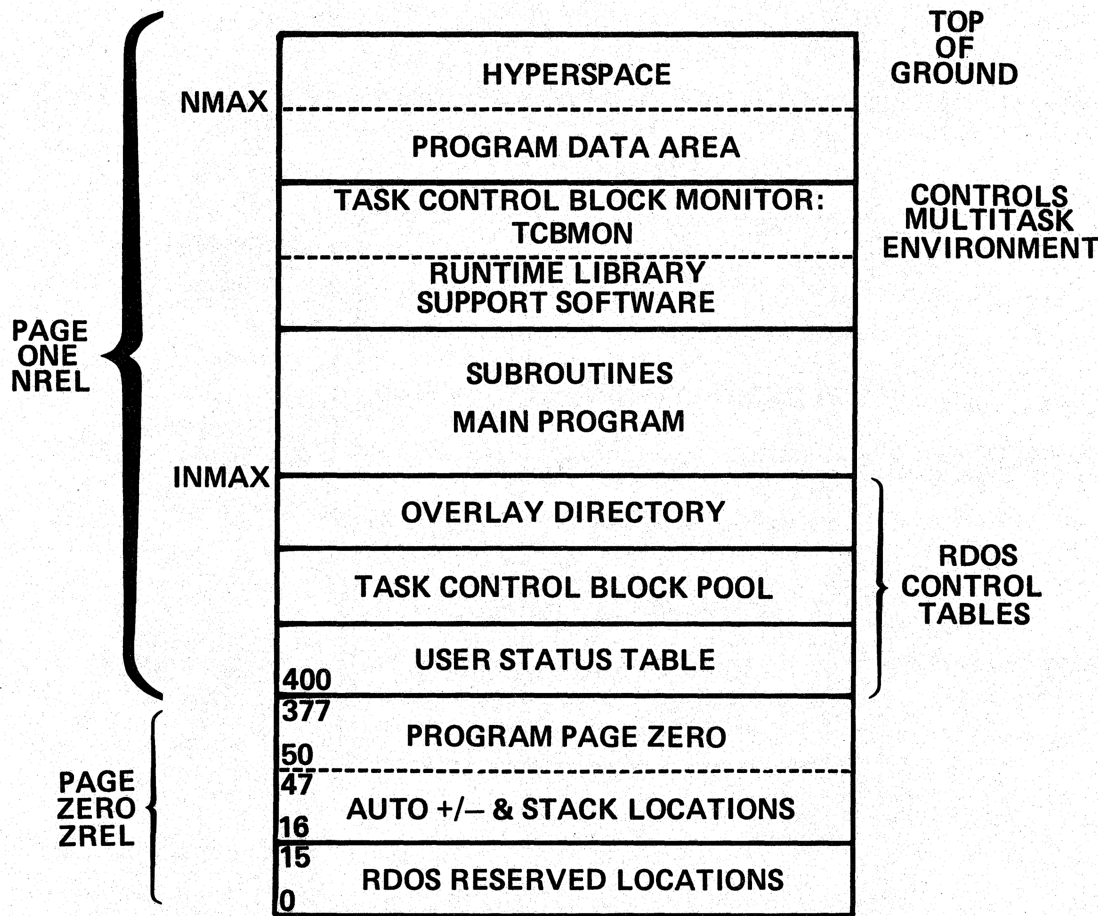


RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND



- No Special RLDR Considerations
- The Map Unit Translates All References to One of Three User Address Spaces
- Each Ground may occupy a full 32K words
- Virtual Techniques Permit Extended Memory Access above 32K words.

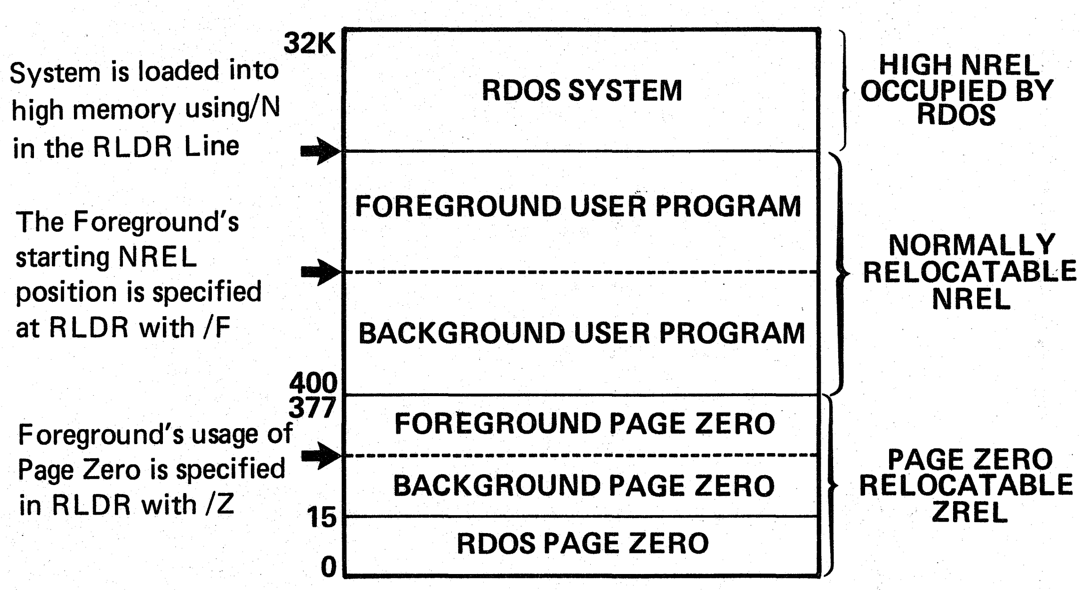
RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND



- Program Status is held in the User Status Table
- Multitasking is controlled by TCBMON and the TCB Pool.
- Overlays are controlled with the Overlay Directory

RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND

UNMAPPED CONFIGURATION



Example:

RLDR $\#_8$/F $\#_8$/Z FGPRGM LIBRARY

- Unmapped FG programs are specially loaded, the .RB file must be accessable
- Commands to control FG programs are identicle to the mapped environment.

RDOS PROCESS MANAGEMENT: FOREGROUND/BACKGROUND

CLI CONTROLS

- GMEM — Get Memory displays FG/BG page usage:

  GMEM
  BG: 29   FG: 29

- SMEM — Set Memory sets BG page usage. (FG gets remainder)

  SMEM 31
  R
  GMEM
  BG: 31   FG: 27

- FGND — Interrogates whether an FG program is executing

  FGND
  NO FOREGROUND PROGRAM

- EXFG/E — Execute a program in the Foreground with equal priority; Foreground at a higher priority is the default.

  R
  EXFG/E FGPRGM
  R

- CNTRL F — Terminate a Foreground Program

  ↑F
  FG TERM

S200

RDOS USER

MODULE 18

RDOS EXTENDED MEMORY: VIRTUAL TECHNIQUES

MODULE 18

OBJECTIVES

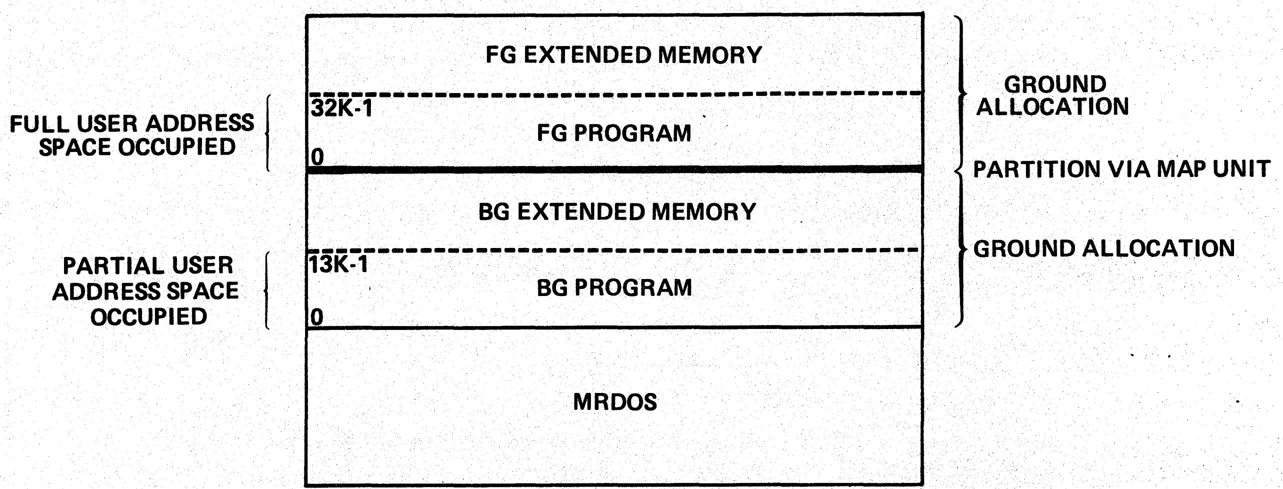
RDOS EXTENDED MEMORY: VIRTUAL TECHNIQUES

Upon successful completion of this module you will be able to:

* DESCRIBE TECHNIQUES FOR EXTENDING USER ADDRESS SPACE
* DEFINE THE OPTIMUM USAGE OF VIRTUAL TECHNIQUES TO MANAGE MEMORY

RDOS EXTENDED MEMORY: VIRTUAL TECHNIQUES

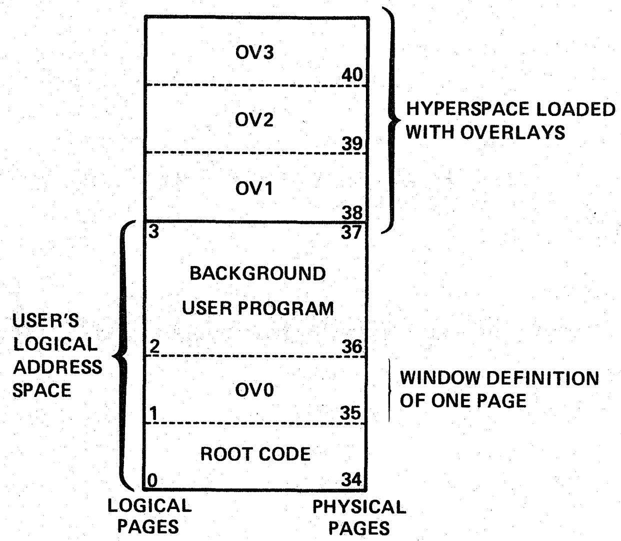
DEFINITION OF EXTENDED MEMORY/HYPERSPACE



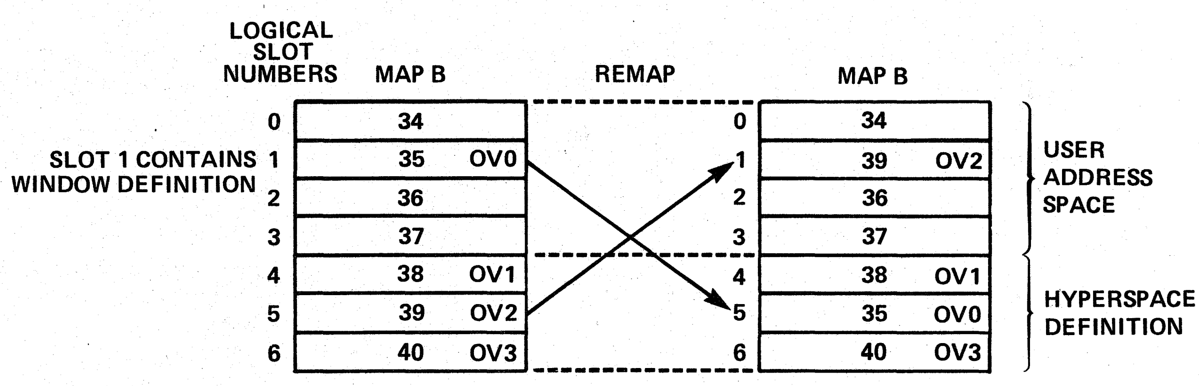
- Extended Memory is that memory within the ground not occupied by the program.
- Access is provided by RDOS via Map manipulation
- Data access in Hyperspace constitutes Window Mapping
- Code access in Hyperspace constitutes Virtual Overlays.

RDOS EXTENDED MEMORY: VIRTUAL TECHNIQUES

VIRTUAL OVERLAYS



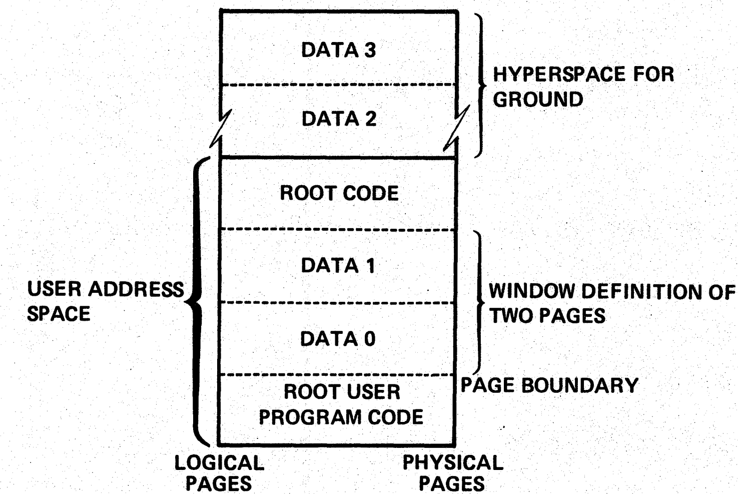
- Virtual Overlays require a local /V after the overlay designation:
  RLDR PRGM [OV0, OV1, OV2, OV3] /V LIBRARY
- During overlay channel open, overlays are loaded as shown above.
- For alternate overlay access the Map Translation Table is altered:



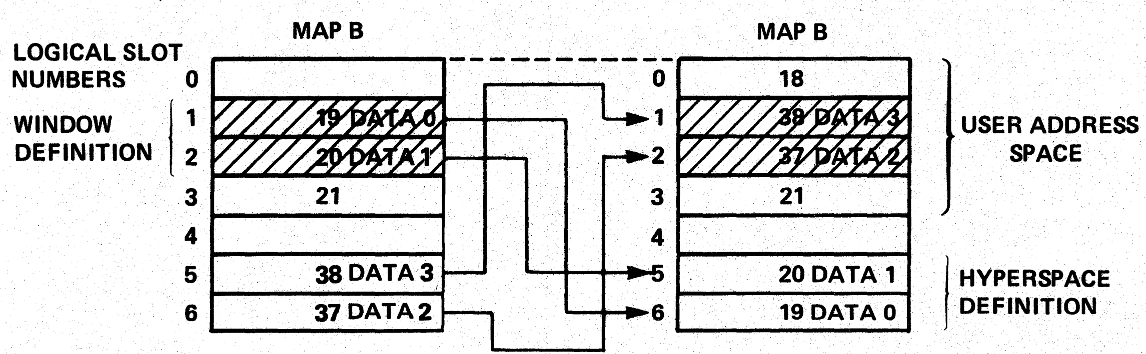
- Alternate overlay access is very fast, two numbers are moved.

RDOS EXTENDED MEMORY: VIRTUAL TECHNIQUES

DATA ACCESS: WINDOW MAPPING



- Special Commands, **ERDB,EWRB** allow reading/writing virtual areas to/from disk.
- The REMAP call allows individual page mapping



- Again Access is extremely fast because a minimum of information is actually moved.

S200

RDOS USER

MODULE 19

RDOS MALFUNCTIONS & RECOVERY

MODULE 19

OBJECTIVES

RDOS MALFUNCTIONS & RECOVERY

Upon successful completion of this module you will be able to:

* EMPLOY RECOVERY TECHNIQUES FOR RDOS MALFUNCTIONS
* ENUMERATE DOCUMENTATION REQUIRED BY DGC PERSONNEL TO DEBUG OR DIAGNOSE SYSTEM MALADIES

RDOS MALFUNCTIONS & RECOVERY

TRAP Program Counter AC0 AC1 A3
BREAK
R

- Traps occur via the map unit due to:

Validity	–	Unauthorized Memory Reference
I/O	–	Unauthorized device I/O attempted
Read/Write	–	Unauthorized I/O to memory attempted
Defer	–	Greater than 16 levels of indirection attempted

- Correction – Copy the message at $TTO(1) above
  – FPRINT/L BREAK.SV or SAVE USERNAME
  – User program is amiss – correct it . . .

- The accumulators have the indicated value at error.

- The program counter has the memory location of the improper instruction.

RDOS MALFUNCTIONS & RECOVERY

SYSTEM CRASH:

ACO AC1 AC2 AC3 Panic Code

- Panics occur whenever a vital error occurs, RDOS stops at the start of the core dump algorithm.

- Documentation for DG System's Engineer:
  - Copy front panel lights
  - Core Dump to Mag. Tape
  - Print System Generation Dialogue
  - Print System Patch Dialogue
  - Print System Load Map
  - Document Hardware Configuration

- Producing a Core Dump
  - Answer SYSGEN question affirmative
  - Upon Panic put all front panel switches down
  - Press Continue twice

- Reboot the system and clear all files left open to recover.

- If software suspicious, perform backup and reboot.

- CPU power off/on and reboot the system.

System Generation and Patch

APPENDIX A

- System Example
- Performing an RDOS Update
- Guidelines for Sizing RDOS 6.10
- RDOS Error Summary

```
ECLIPSE SYSGEN REV 6.40
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

MAPPED SYSTEM? ("0"=NO "1"=YES) $1
  S/250 OR C/350 TYPE PROCESSOR? ("0"=NO "1"=YES) $0
  S/200 OR C/300 MAP? ("0"=NO "1"=YES) $1
  MAXIMUM NUMBER OF CHANNELS BACKGROUND WILL USE(1-255) $32
  MAXIMUM NUMBER OF CHANNELS FOREGROUND WILL USE(0-255) $32
NUMBER OF NOVADISK DISK CONTROLLERS(0-2) $0
NUMBER OF 6063/6064 DISK CONTROLLERS(0-2) $0
NUMBER OF 6060/6061/6067 DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $1
NUMBER OF OTHER TYPES OF MOVING HEAD DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $2
    TOP LOADER(S)? ("0"=NO "1"=YES) $1
ENTER BAD BLOCK POOL SIZE IN BLOCKS (0-512) $12
DUAL PROCESSORS (IPB)? ("0"=NO "1"=YES) $0
ENTER NUMBER OF STACKS (1-10) $8
ENTER NUMBER OF EXTRA CELLS (0-64) $12
TUNING? ("0"=NO "1"=YES) $0
ENTER NUMBER OF EXTRA BUFFERS REQUIRED (0-63) $8
MAXIMUM NUMBER OF SUB-DIRECTORIES/SUB-PARTITIONS
ACCESSIBLE AT ONE TIME (0-64) $12
ENTER NUMBER OF CONTROLLERS FOR MTA(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER NUMBER OF DEVICES FOR CONTROLLER #1 (1-8) $1
ENTER NUMBER OF CONTROLLERS FOR CTA(0-2) $0
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) $0
OPERATOR MESSAGES? ("0"=NO "1"=YES) $0
RTC? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) $1
ENTER NUMBER OF PTR(0-2) $0
ENTER NUMBER OF PTP(0-2) $0
ENTER NUMBER OF LPT(0-2) $1
  ENTER COLUMN SIZE FOR LPT #1 (80 OR 132) $80
    DATA CHANNEL LINE PRINTER? ("0"=NO  "1"=YES) $1
ENTER NUMBER OF CDR(0-2) $0
ENTER NUMBER OF PLT(0-2) $0
ENTER NUMBER OF MCA(0-2) $0
QTY? ("0"=NO "1"=YES) $0
ULM? ("0"=NO "1"=YES) $0
ALM? ("0"=NO "1"=YES) $0
SECOND TTY? ("0"=NO "1"=YES) $1
CORE DUMP FACILITY? ("0"=NO  "1"=LPT  "2"=MTA  "3"=6030) $2
```

```
ECLIPSE SYSGEN REV 6.40
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

MAPPED SYSTEM? ("0"=NO "1"=YES) $1
  S/250 OR C/350 TYPE PROCESSOR? ("0"=NO "1"=YES) $0
  S/200 OR C/300 MAP? ("0"=NO "1"=YES) $1
  MAXIMUM NUMBER OF CHANNELS BACKGROUND WILL USE(1-255) $32
  MAXIMUM NUMBER OF CHANNELS FOREGROUND WILL USE(0-255) $32
NUMBER OF NOVADISK DISK CONTROLLERS(0-2) $0
NUMBER OF 6063/6064 DISK CONTROLLERS(0-2) $0
NUMBER OF 6060/6061/6067 DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $1
NUMBER OF OTHER TYPES OF MOVING HEAD DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $2
    TOP LOADER(S)? ("0"=NO "1"=YES) $1
ENTER BAD BLOCK POOL SIZE IN BLOCKS (0-512) $12
DUAL PROCESSORS (IPB)? ("0"=NO "1"=YES) $0
ENTER NUMBER OF STACKS (1-10) $8
ENTER NUMBER OF EXTRA CELLS (0-64) $12
TUNING? ("0"=NO "1"=YES) $0
ENTER NUMBER OF EXTRA BUFFERS REQUIRED (0-63) $8
MAXIMUM NUMBER OF SUB-DIRECTORIES/SUB-PARTITIONS
ACCESSIBLE AT ONE TIME (0-64) $12
ENTER NUMBER OF CONTROLLERS FOR MTA(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER NUMBER OF DEVICES FOR CONTROLLER #1 (1-8) $1
ENTER NUMBER OF CONTROLLERS FOR CTA(0-2) $0
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) $0
OPERATOR MESSAGES? ("0"=NO "1"=YES) $0
RTC? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) $1
ENTER NUMBER OF PTR(0-2) $0
ENTER NUMBER OF PTP(0-2) $0
ENTER NUMBER OF LPT(0-2) $1
  ENTER COLUMN SIZE FOR LPT #1 (80 OR 132) $80
    DATA CHANNEL LINE PRINTER? ("0"=NO  "1"=YES) $1
ENTER NUMBER OF CDR(0-2) $0
ENTER NUMBER OF PLT(0-2) $0
ENTER NUMBER OF MCA(0-2) $0
QTY? ("0"=NO "1"=YES) $0
ULM? ("0"=NO "1"=YES) $0
ALM? ("0"=NO "1"=YES) $0
SECOND TTY? ("0"=NO "1"=YES) $1
CORE DUMP FACILITY? ("0"=NO  "1"=LPT  "2"=MTA  "3"=6030) $2
```

```
NOVA 3 SYSGEN REV 6.40
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

MAPPED SYSTEM? ("0"=NO "1"=YES) $1
  MAXIMUM NUMBER OF CHANNELS BACKGROUND WILL USE(1-255) $32
  MAXIMUM NUMBER OF CHANNELS FOREGROUND WILL USE(0-255) $32
NUMBER OF NOVADISK DISK CONTROLLERS(0-2) $
NUMBER OF 6063/6064 DISK CONTROLLERS(0-2) $
NUMBER OF 6060/6061/6067 DISK CONTROLLERS(0-2) $
NUMBER OF OTHER TYPES OF MOVING HEAD DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $1
    TOP LOADER(S)? ("0"=NO "1"=YES) $1
ENTER BAD BLOCK POOL SIZE IN BLOCKS (0-512) $12
DUAL PROCESSORS (IPB)? ("0"=NO "1"=YES) $0
ENTER NUMBER OF STACKS (1-10) $5
ENTER NUMBER OF EXTRA CELLS (0-64) $12
TUNING? ("0"=NO "1"=YES) $0
ENTER NUMBER OF EXTRA BUFFERS REQUIRED (0-63) $5
MAXIMUM NUMBER OF SUB-DIRECTORIES/SUB-PARTITIONS
ACCESSIBLE AT ONE TIME (0-64) $12
ENTER NUMBER OF CONTROLLERS FOR MTA(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER NUMBER OF DEVICES FOR CONTROLLER #1 (1-8) $1
ENTER NUMBER OF CONTROLLERS FOR CTA(0-2) $
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) $
OPERATOR MESSAGES? ("0"=NO "1"=YES) $
RTC? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) $1
ENTER NUMBER OF PTR(0-2) $
ENTER NUMBER OF PTP(0-2) $
ENTER NUMBER OF LPT(0-2) $1
  ENTER COLUMN SIZE FOR LPT #1 (80 OR 132) $80
    DATA CHANNEL LINE PRINTER? ("0"=NO  "1"=YES) $0
ENTER NUMBER OF CDR(0-2) $
ENTER NUMBER OF PLT(0-2) $
ENTER NUMBER OF MCA(0-2) $
QTY? ("0"=NO "1"=YES) $
ULM? ("0"=NO "1"=YES) $
ALM? ("0"=NO "1"=YES) $
SECOND TTY? ("0"=NO "1"=YES) $1
CORE DUMP FACILITY? ("0"=NO  "1"=LPT  "2"=MTA  "3"=6030) $2
```

```
ECLIPSE SYSGEN REV 6.40
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

MAPPED SYSTEM? ("0"=NO "1"=YES) $1
  S/250 OR C/350 TYPE PROCESSOR? ("0"=NO "1"=YES) $0
  S/200 OR C/300 MAP? ("0"=NO "1"=YES) $1
  MAXIMUM NUMBER OF CHANNELS BACKGROUND WILL USE(1-255) $32
  MAXIMUM NUMBER OF CHANNELS FOREGROUND WILL USE(0-255) $32
NUMBER OF NOVADISK DISK CONTROLLERS(0-2) $0
NUMBER OF 6063/6064 DISK CONTROLLERS(0-2) $0
NUMBER OF 6060/6061/6067 DISK CONTROLLERS(0-2) $0
NUMBER OF OTHER TYPES OF MOVING HEAD DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $3
    TOP LOADER(S)? ("0"=NO "1"=YES) $1
ENTER BAD BLOCK POOL SIZE IN BLOCKS (0-512) $12
DUAL PROCESSORS (IPB)? ("0"=NO "1"=YES) $0
ENTER NUMBER OF STACKS (1-10) $5
ENTER NUMBER OF EXTRA CELLS (0-64) $12
TUNING? ("0"=NO "1"=YES) $0
ENTER NUMBER OF EXTRA BUFFERS REQUIRED (0-63) $5
MAXIMUM NUMBER OF SUB-DIRECTORIES/SUB-PARTITIONS
ACCESSIBLE AT ONE TIME (0-64) $12
ENTER NUMBER OF CONTROLLERS FOR MTA(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER NUMBER OF DEVICES FOR CONTROLLER #1 (1-8) $1
ENTER NUMBER OF CONTROLLERS FOR CTA(0-2) $0
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) $0
OPERATOR MESSAGES? ("0"=NO "1"=YES) $1
RTC? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) $1
ENTER NUMBER OF PTR(0-2) $0
ENTER NUMBER OF PTP(0-2) $0
ENTER NUMBER OF LPT(0-2) $1
  ENTER COLUMN SIZE FOR LPT #1 (80 OR 132) $80
    DATA CHANNEL LINE PRINTER? ("0"=NO  "1"=YES) $0
ENTER NUMBER OF CDR(0-2) $0
ENTER NUMBER OF PLT(0-2) $0
ENTER NUMBER OF MCA(0-2) $0
QTY? ("0"=NO "1"=YES) $0
ULM? ("0"=NO "1"=YES) $0
ALM? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ALM CLOCK FREQUENCY? (0-3) $0
  USE DEFAULT ALM/QTY INTERRUPT CHARACTERS? ("0"=NO "1"=YES) $1
  NUMBER OF NULLS AFTER CARRIAGE RETURN? (0-256) $0
SECOND TTY? ("0"=NO "1"=YES) $1
CORE DUMP FACILITY? ("0"=NO  "1"=LPT  "2"=MTA  "3"=6030) $2
```

```
ECLIPSE SYSGEN REV 6.40
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

MAPPED SYSTEM? ("0"=NO "1"=YES) $1
  S/250 OR C/350 TYPE PROCESSOR? ("0"=NO "1"=YES) $0
  S/200 OR C/300 MAP? ("0"=NO "1"=YES) $1
  MAXIMUM NUMBER OF CHANNELS BACKGROUND WILL USE(1-255) $32
  MAXIMUM NUMBER OF CHANNELS FOREGROUND WILL USE(0-255) $32
NUMBER OF NOVADISK DISK CONTROLLERS(0-2) $0
NUMBER OF 6063/6064 DISK CONTROLLERS(0-2) $0
NUMBER OF 6060/6061/6067 DISK CONTROLLERS(0-2) $0
NUMBER OF OTHER TYPES OF MOVING HEAD DISK CONTROLLERS(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  NUMBER OF DEVICES FOR CONTROLLER #1(1-4) $3
    TOP LOADER(S)? ("0"=NO "1"=YES) $1
ENTER BAD BLOCK POOL SIZE IN BLOCKS (0-512) $12
DUAL PROCESSORS (IPB)? ("0"=NO "1"=YES) $0
ENTER NUMBER OF STACKS (1-10) $5
ENTER NUMBER OF EXTRA CELLS (0-64) $12
TUNING? ("0"=NO "1"=YES) $0
ENTER NUMBER OF EXTRA BUFFERS REQUIRED (0-63) $5
MAXIMUM NUMBER OF SUB-DIRECTORIES/SUB-PARTITIONS
ACCESSIBLE AT ONE TIME (0-64) $12
ENTER NUMBER OF CONTROLLERS FOR MTA(0-2) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER NUMBER OF DEVICES FOR CONTROLLER #1 (1-8) $1
ENTER NUMBER OF CONTROLLERS FOR CTA(0-2) $0
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) $0
OPERATOR MESSAGES? ("0"=NO "1"=YES) $1
RTC? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ENTER RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) $1
ENTER NUMBER OF PTR(0-2) $0
ENTER NUMBER OF PTP(0-2) $0
ENTER NUMBER OF LPT(0-2) $1
  ENTER COLUMN SIZE FOR LPT #1 (80 OR 132) $80
    DATA CHANNEL LINE PRINTER? ("0"=NO  "1"=YES) $0
ENTER NUMBER OF CDR(0-2) $0
ENTER NUMBER OF PLT(0-2) $0
ENTER NUMBER OF MCA(0-2) $0
QTY? ("0"=NO "1"=YES) $0
ULM? ("0"=NO "1"=YES) $0
ALM? ("0"=NO "1"=YES) $1
  DEVICE PRIMARY("0") OR SECONDARY("1")? $0
  ALM CLOCK FREQUENCY? (0-3) $0
  USE DEFAULT ALM/QTY INTERRUPT CHARACTERS? ("0"=NO "1"=YES) $1
  NUMBER OF NULLS AFTER CARRIAGE RETURN? (0-256) $0
SECOND TTY? ("0"=NO "1"=YES) $1
CORE DUMP FACILITY? ("0"=NO  "1"=LPT  "2"=MTA  "3"=6030) $2
```

IN ORDER TO USE THIS UPDATE, YOU MUST HAVE ALREADY BROUGHT UP RELEASE 6.40 OF RDOS IN ACCORDANCE WITH ITS RELEASE NOTICE, 085-000022-05, AND THE INSTRUCTIONS IN "HOW TO LOAD AND GENERATE YOUR RDOS SYSTEM".

TO INSTALL THIS UPDATE:

1) CREATE A SUBDIRECTORY TO CONTAIN THE UPDATE FILES, AND THEN MAKE THE NEW SUBDIRECTORY YOUR DEFAULT DIRECTORY. FOR EXAMPLE:

    CDIR RDOSUD641;DIR RDOSUD641

2) LOAD THE FILE NAMED "UPDATE" CONTAINED IN THE UPDATE MATERIALS ACCOMPANYING THIS NOTICE. TO DO THIS, USE THE COMMANDS FROM THE SET BELOW WHICH MATCH THE MEDIUM YOU HAVE:

    FROM MAGNETIC TAPE 071-000224-03
    INIT MTX; LOAD/V MTX:0 UPDATE;RELEASE MTX
    [WHERE "X" IS THE UNIT NUMBER THE TAPE IS ON]

    FROM CASSETTE 070-000175-03
    INIT CTX;LOAD/V CTX:0 UPDATE;RELEASE CTX
    [WHERE "X" IS THE UNIT NUMBER THE CASSETTE IS ON]

    FROM DISKETTE 072-000091-03
    DIR DPX;MOVE/V RDOSUD641 UPDATE;DIR RDOSUD641
    [WHERE "X" IS THE UNIT NUMBER OF THE DISKETTE DRIVE]

    FROM PAPER TAPE 088-000326-03
    088-000344-02
    LOAD/V $PTR UPDATE
    [THEN MOUNT THE FIRST PAPER TAPE IN THE READER, FOLLOWED BY THE REST OF THE TAPES AS REQUESTED]

3) READ AND FOLLOW THE INSTRUCTIONS IN THE FILE "UPDATE". TO SEE THIS INFORMATION ON YOUR LINE PRINTER, USE THE COMMAND

    PRINT UPDATE

IF YOU DON'T HAVE A LINE PRINTER, DISPLAY THIS INFORMATION ON YOUR CONSOLE WITH THE COMMAND

    TYPE UPDATE

THE PURPOSE OF THE PRODUCT UPDATE IS TO REDUCE THE TIME REQUIRED TO RESPOND TO PROBLEMS, BY PROVIDING USERS WITH THE MINIMUM MATERIAL REQUIRED TO UPDATE THE PRODUCT AND ITS STATUS. THIS DOCUMENT WILL DETAIL THE STEPS NECESSARY TO INSTALL THE UPDATE.

THE SPECIFIC CONTENT OF RDOS REV 6.40 UPDATE 1 IS DEFINED BY THE FOLLOWING TABLE.

FILE NAME	DESCRIPTION
UPDATE	THIS FILE.
ARDOS.PF BRDOS.PF MRDOS.PF NRDOS.PF URDOS.PF	PATCH FILES FOR VARIOUS RDOS SYSTEMS (ZRDOS USERS WILL USE ARDOS.PF).
BSYSGEN64.PF	PATCH FILE FOR BSYSGEN
BSGENPATCH.MC	COMMAND FILE TO UPDATE BSYSGEN

MAGTAPE, CASSETTE USERS - - YOU WILL FIND THESE FILES, IN DUMP FORMAT, ON FILE 0 OF YOUR TAPE.

DISKETTE USERS -- YOU WILL FIND THESE FILES, IN FILE FORMAT, ON YOUR DISKETTE.

PAPER TAPE USERS -- YOU WILL FIND THESE FILES, IN SEGMENTED DUMP FORMAT, ON YOUR PAPER TAPES.

TABLE OF CONTENTS

IN THE EXAMPLES GIVEN BELOW, WE WILL MAKE 3 (THREE) ASSUMPTIONS:

1. UTILITIES (EXCEPT WHERE NOTED) RESIDE IN DIRECTORY "UTIL".
2. RDOS LIBRARIES FROM WHICH YOU SYSGEN RESIDE IN DIRECTORY "SYSGEN".
3. UPDATE FILES WILL BE LOADED/MOVED INTO DIRECTORY "RDOSUD641".

COMMANDS WHICH YOU WILL TYPE FROM THE CONSOLE ARE UNDERLINED IN THE FOLLOWING EXAMPLES.

THE SYMBOL "*" (ASTERISK) IN THE FOLLOWING EXAMPLES IS DEFINED AS FOLLOWS:

MAPPED ECLIPSE (S/200, C/300) USERS:	A
MAPPED ECLIPSE (S/130, S/230, C/330) USERS:	Z
MAPPED ECLIPSE (S/250, C/350) USERS:	Z
UNMAPPED ECLIPSE USERS:	B
MAPPED NOVA USERS:	M
MAPPED NOVA 3 USERS:	N
UNMAPPED NOVA USERS:	U

THUS, FOR MAPPED NOVA 3 USERS, THE FILE *RDOSC.LB WOULD REFER TO THE FILE NRDOSC.LB

THIS UPDATE INCLUDES SUPPORT FOR THE ARRAY PROCESSOR SUBSYSTEM ON THE ECLIPSE S/250. THIS SUPPORT HAS THE ADDITIONAL CAPABILITY OF ALLOWING CONFIGURATIONS OF AP MEMORY WHICH CAN EXIST ANYWHERE WITHIN 2 MEGABYTES AS LONG AS IT'S THE LAST 4K OF PHYSICAL MEMORY. IF THE AP IS SYSGENED, RDOS WILL DYNAMICALLY FIND THE LOCATION OF THE AP MEMORY. IT SHOULD BE NOTED THAT RDOS STILL ONLY SUPPORTS UP TO .5 MEGABYTES ON MEMORY, THIS ADDITIONAL SUPPORT BEING JUST A SPECIAL CASE.

HOW TO LOAD THE UPDATE FILES

1. CREATE A SUBDIRECTORY TO CONTAIN THE UPDATE FILES AND MAKE IT YOUR CURRENT DEFAULT DIRECTORY.

   NOTE: IF YOU FOLLOWED THE INSTRUCTIONS ON THE UPDATE NOTICE, YOU WILL HAVE ALREADY PERFORMED THESE STEPS.

EXAMPLE:

R	(CLI READY)
CDIR RDOSUD641	(CREATE UPDATE SUBDIRECTORY)
R	(CLI READY)
DIR RDOSUD641	(MAKE IT CURRENT DEFAULT)
R	(CLI READY)

2. LOAD THE APPROPRIATE UPDATE FILES (ACCORDING TO YOUR SYSTEM).

A) FOR MAGNETIC TAPE
FROM MTX, WHERE X = UNIT NUMBER WHERE UPDATE TAPE RESIDES:

EXAMPLE:

R	(CLI READY)
INIT MTX	(INIT MAG TAPE UNIT)
R	(CLI READY)
LOAD/V MTX:0	(FILES LOADED AND LISTED)
*RDOS.PF BSYSGEN64.PF BSGENPATCH.MC	
R	(UPDATE FILES LOADED)

B) FOR CASSETTE
SUBSTITUTE CTX FOR MTX IN SECTION A, ABOVE.

C) FROM PAPER TAPE

EXAMPLE:

R	(CLI READY–
LOAD/V $PTR	(FILES LOADED AND LISTED)
*RDOS.PF BSYSGEN64.PF BSGENPATCH.MC	
R	(UPDATE FILES LOADED)

C) FROM PAPER TAPE

EXAMPLE:

```
R                                   (CLI READY)
LOAD/V $PTR                         (FILES LOADED AND LISTED)
-----------------------
*RDOS.PF BSYSGEN64.PF BSGENPATCH.MC
-------- ------------ ---------------
R                                   (UPDATE FILES LOADED)
```

D) FOR DISKETTE
FROM DPX, WHERE X = UNIT NUMBER WHERE UPDATE DISKETTE RESIDES:

EXAMPLE:

```
R                                   (CLI READY)
DIR DPX                             (MAKE UPDATE DISKETTE CURRENT)
--------------                      DEFAULT DIRECTORY)
R                                   (CLI READY)
MOVE/V RDOSUD641                    (FILES MOVED AND LISTED)
-----------------------------------
*RDOS.PF BSYSGEN64.PF BSGENPATCH.MC
-------- ------------ ---------------
R                                   (UPDATE FILES MOVED)
```

HOW TO APPLY SYSGEN PATCHES

MAPPED ECLIPSE USERS RUNNING ON THE ECLIPSE S/250 WHO DESIRE ARRAY PROCESSOR SUPPORT SHOULD INSTALL THIS PATCH TO BSYSGEN. ALL OTHER USERS CAN SKIP THIS SECTION ENTIRELY.

TO UPDATE YOUR BSYSGEN PROGRAM, FOLLOW THE PROCEDURE OUTLINED BELOW.

NOTE: WE WILL ASSUME THAT YOUR BSYSGEN PROGRAM RESIDES IN THE DIRECTORY SYSGEN.

EXAMPLE:

```
R                                  (CLI READY)
DIR RDOSUD641                      (GET INTO THE UPDATE DIRECTORY)
----------------------------

R                                  (CLI READY)
LINK BSYSGEN.SV SYSGEN:BSYSGEN.SV
-------------------------------------------------------------------
                                   (CREATE LINK TO BSYSGEN)

R                                  (CLI READY)
LINK PATCH.SV UTIL:PATCH.SV        (LINK TO THE UTILITY PATCH.SV)
-----------------------------------------------------

R                                  (CLI READY)
BSGENPATCH                         (PATCH MACRO FOR BSYSGEN)
-----------------------

R                                  (CLI READY, BSYSGEN PATCHED)
```

HOW TO APPLY RDOS PATCHES

N.B.: RDOS PATCHES MUST BE APPLIED TO EVERY SYSTEM THAT IS SYSGENED.

PATCHING YOUR RDOS SYSTEM INVOLVES THE FOLLOWING GENERAL STEPS:

1. MAKE THE DIRECTORY IN WHICH YOU HAVE YOUR RDOS LIBRARIES ("SYSGEN") YOUR CURRENT ONE.
2. SYSGEN A NEW SYSTEM USING THE LIBRARIES IN DIRECTORY "SYSGEN". BE SURE TO REQUEST A SYSTEM LOAD MAP.
3. LINK TO THE APPROPRIATE PATCH FILE.
4. LINK TO THE UTILITY PATCH.SV.
5. INVOKE THE PATCH UTILITY.

ASSUME YOU HAVE SYSGENED AN NRDOS SYSTEM CALLED "NSYS", WHOSE LOAD MAP NAME IS "NSYS.LM". THE FOLLOWING IS AN EXAMPLE OF WHAT THE PATCH PROCEDURE WOULD BE.

EXAMPLE:

```
R                                        (CLI READY)
DIR SYSGEN                               (GET TO "SYSGEN")
---------------------
R                                        (CLI READY)
LINK NRDOS.PF RDOSUD641:NRDOS.PF   (LINK TO PATCH FILE)
-----------------------------------------------------------------
R                                        (CLI READY)
PATCH NRDOS.PF/P NSYS/S NSYS.LM/L    (INVOKE PATCH)
-----------------------------------------------------------------
- - - -                                  (PATCHES BEING INSTALLED)
R                                        (SYSTEM PATCHED)
```

6. THE PATCH INSTALLATION PROCESS MAY CONTINUE BY ASKING FOR FURTHER INFORMATION. FOR EXAMPLE, THERE MAY BE A PATCH WHICH ONLY A CERTAIN SET OF USERS WOULD LIKE INSTALLED. IN THIS CASE, YOU MUST RESPOND WITH A '1' (YES) OR '0' (NO) ANSWER TO THE QUESTION.

7. YOU MUST SAVE A COPY OF YOUR SYSGEN DIALOGUE, LOAD MAP AND PATCH DIALOGUE (.PD) FILE IN CASE YOU SUBMIT AN STR OR CORE DUMP TO DATA GENERAL FOR ANALYSIS. THIS WILL INSURE THAT DATA GENERAL /CAN TAKE YOUR PATCHES INTO ACCOUNT WHEN ANALYZING YOUR SYSTEM.

8. YOU SHOULD CONTINUE TO UPDATE YOUR RDOS SYSTEMS AS THEY ARE GENERATED. THIS CAN BE DONE EASILY BY KEEPING THE DIRECTORY RDOSUD641 ON DISK UNTIL THE NEXT UPDATE OR SYSTEM REVISION IS ISSUED.

PROBLEMS/STATUS

NOTE: THE NUMBERS IN PARENTHESIS REPRESENT STR NUMBERS AND ARE FOR INTERNAL (DGC) USE.

RDOS

1) (1062) THE VIRTUAL OVERLAY HANDLER DOES NOT CHECK WHETHER AN OVERLAY FILE HAS BEEN PREVIOUSLY OPENED. THIS CAN CAUSE A SYSTEM CRASH ON CONTROL–A OR .RIN.

2) (1194) IF RDOS CANNOT RESOLVE DISK SEEK ERRORS WITHIN THE TIMEOUT PERIOD, THE ERROR RETURNED IS DISK TIMEOUT INSTEAD OF DISK SEEK ERROR. BECAUSE OF THIS, A PANIC 6 (MASTER DEVICE TIMEOUT) CAN RESULT FROM REPEATED DISK SEEK ERRORS ON THE MASTER DEVICE.

3) (1365) THE .OL FILE OF THE CURRENTLY RUNNING SYSTEM HAS A USE COUNT OF 0, PERMITTING ACCIDENTAL DELETION OF THE FILE.

4) (1787, 2572) CHECKPOINTING WILL CAUSE A PANIC 3 ON ECLIPSE SYSTEMS IF THE BACKGROUND HAS BEEN PUSHED. SEE PATCH R-1.

5) (2156) USER .IDEF OF ERCC INTERRUPTS DOES NOT WORK PROPERLY.

6) (2187) WHEN A TAPE DRIVE GOES OFF–LINE, RDOS WILL NOT GIVE AN ERROR UNTIL THE DRIVE COMES BACK ON LINE.

7) (3274) QTASKING WITH OVERLAYS DOES NOT WORK PROPERLY.

8) (3279) WHEN A DEVICE IS ENABLED BY .DEBL IT REMAINS ENABLED UNTIL IT IS CHANGED BY A .DDIS.

9) (4393) THE NOVA 3 MAP IS NOT SET UP ON THE TASK CALL .SMSK. SEE PATCH R–2. (REV 6.41).

10) (2140) ATTEMPTING TO EXTEND A FILE BEYOND THE MAXIMUM LENGTH WILL NOT GIVE AN ERROR MESSAGE. SEE PATCH R–4. (REV 6.41).

11) BUFFER OVERFLOW ON A QTY WILL NOT GIVE ERROR MESSAGES. SEE PATCH R–4. (REV 6.41).

12) SPURIOUS QTY INTERRUPTS WILL CAUSE THE SYSTEM TO HANG. SEE PATCH R–6. (REV 6.41).

13) THE SYSTEM WILL NOT RECOVER FROM ERRORS DETECTED IN ATTEMPTING TO LOAD USER OR VIRTUAL OVERLAYS. SEE PATCH R–8. (REV 6.41).

14) HEAVY I/D IN ONE GROUND AND TERMINAL INTERACTION IN THE OTHER GROUND IS NOT HANDLED PROPERLY IF THE SYSTEM RUNS OUT OF CELLS ON MAPPED NOVAS AND NOVA 3 SYSTEMS. SEE PATCH R–9. (REV 6.41).

UTILITIES

SPEED

----

1) (687) THE CHARACTERISTIC INHIBIT MASK CONTAINS GARBAGE IN THE RIGHT BYTE WHEN OPENING A FILE VIA A "GRFILENAME$" COMMAND.

2) (741) BUFFER COMMANDS OF THE FORM "BNN" AND THE "–N" COMMAND WHEN USED INCORRECTLY PRODUCE THE ERROR MESSAGE "ILLEGAL ARGUMENT TO COMMAND" RATHER THAN "ILLEGAL COMMAND".

3) (811) THE ERROR MESSAGE "STACK OVERFLOW" IS RETURNED BY SPEED (AND NSPEED DIES) WHEN ENOUGH "WC" OR "WM" COMMANDS ARE ENTERED IN A SINGLE COMMAND LINE.

4) (881) SPEED ERRORS CAN CAUSE STACK UNDERFLOW, WHICH CAN RESULT IN THE PROGRAM'S BEING OVERWRITTEN.

5) (902) THE TRACE MODE FEATURE DOES NOT WORK PROPERLY.

OTHER UTILITIES

----

1) THERE IS A PROBLEM IN MEDIT IN HANDLING LINES WHICH CONTAIN TOO MANY CHARACTERS. IF A 'UEH' IS ISSUED ON A FILE WHICH CONTAINS A LINE WITH MORE THAN 132 CHARACTERS, A "LINE TOO LONG" MESSAGE OCCURS, AND MEDIT TERMINATES THE SOURCE FILE AT THE LINE PRIOR TO THE LONG LINE.

2) (761) IN OEDIT, WHEN AN OPEN LOCATION IS USED AS AN ADDRESS FOR THE NEXT OPEN, THE B (BASE) REGISTER IS NOT ADDED IN WHEN COMPUTING THE ADDRESS.

3) (1123) EXITING FROM MEDIT VAI CONTROL–A CAUSES A SYSTEM CRASH.

GUIDELINES FOR SIZING RDOS REV 6.10

NOTE: UNLESS OTHERWISE SPECIFIED, NUMBERS ARE GIVEN IN THE FORMAT OCTAL (DECIMAL).

1. CHANNELS – 45(37) PER CHANNEL SHSGEN'ED

2. FIXED HEAD DISKS

   A. TABLES – 121 (81) WORDS PER CONTROLLER SYSGEN'ED. (3 EXTRA WORDS PER CONTROLLER FOR NOVA'S)

   B. DRIVER (BOTH CONTROLLERS SHARE SAME DRIVER)

MAPPED ECLIPSE	–	163 (115) WORDS
UNMAPPED ECLIPSE	–	152 (105) WORDS
MAPPED NOVA 840	–	153 (107) WORDS
MAPPED NOVA 3	–	153 (107) WORDS
UNMAPPED NOVA	–	153 (107) WORDS

3. MOVING HEAD DISKS (NOT INCLUDING 96MB/192MB)

   A. TABLES

      1. 15 (13) WORDS PER CONTROLLER SYSGEN'ED (3 EXTRA WORDS PER CONTROLLER FOR NOVA'S)*

      2. 104 (60) WORDS PER DEVICE SYSGEN'ED PER CONTROLLER
         104 (60) EXTRA WORDS PER DEVICE IF TOP LOADER*
         210 (136)

   B. DRIVER (BOTH CONTROLLERS SHARE SAME DRIVER)

MAPPED ECLIPSE	–	455 (301) WORDS
UNMAPPED ECLIPSE	–	444 (292) WORDS
MAPPED NOVA 840	–	446 (294) WORDS
MAPPED NOVA 3	–	446 (294) WORDS*
UNMAPPED NOVA	–	442 (296) WORDS*

96MB/192MB MOVING HEAD DISKS

   A. TABLES

      15 (13) WORDS/CONTROLLER SYSGEN'ED
      (3 EXTRA WORDS/CONTROLLER FOR NOVA'S)

      104 (60) WORDS/DEVICE ON EACH CONTROLLER

B. DRIVER (INCLUDING CORE RESIDENT ECC CODE)

MAPPED ECLIPSE	–	1067 (567) WORDS
UNMAPPED ECLIPSE	–	777 (511) WORDS
MAPPED NOVA 840	–	1112 (586) WORDS
MAPPED NOVA 3	–	1105 (581) WORDS
UNMAPPED NOVA	–	1013 (523) WORDS

4. BAD BLOCK POOL

6 WORDS PER DEVICE (I.E. DISK) SYSGEN'ED + 2 * BAD BLOCK POOL SIZE SYSGEN'ED (TOP LOADERS COUNT AS 2 DEVICES)

5. IPB

MAPPED ECLIPSE	–	2776 (1534) WORDS
UNMAPPED ECLIPSE	–	2776 (1534) WORDS
MAPPED NOVA 840	–	3036 (1566) WORDS
MAPPED NOVA 3	–	3036 (1566) WORDS
UNMAPPED NOVA	–	3036 (1568) WORDS

6. STACKS

ANY ECLIPSE – 310 (200) WORDS PER STACK
ANY NOVA – 340 (224) WORDS PER STACK

NOTE: NUMBER OF STACKS SYSGEN'ED ALSO AFFECTS NUMBER OF CELLS AND BUFFERS TABLE BELOW GIVES ACTUAL TOTALS (ALL NUMBERS ARE IN DECIMAL)

STACKS	CELLS	BUFFERS
1	3	6
2	6	8
3	9	6
4	12	8
5	15	10
6	18	12
7	21	14
8	24	16
9	27	18

7. CELLS – 20 (16) WORDS EACH

8. TUNING – 1 EXTRA BUFFER IF NO OVERLAY REPORT REQUESTED.
3 EXTRA BUFFERS IF OVERLAY REPORT WAS REQUESTED.

9. BUFFERS – 416 (278) WORDS EACH

10. SUB-DIRECTORIES/SUB-PARTITIONS ACCESSIBLE AT ONE TIME - 60 (40) WORDS EACH.

11. MAG TAPES

A. TABLES

1. 111 (73) WORDS PER CONTROLLER SYSGEN'ED (3 EXTRA WORDS PER CONTROLLER FOR NOVA'S)

2. 21 (17) WORDS PER DEVICE SYSGEN'ED PER CONTROLLER

B. DRIVER SIZE (BOTH CONTROLLERS SHARE SAME DRIVER)

MAPPED ECLIPSE	–	606 (390) WORDS
UNMAPPED ECLIPSE	–	610 (394) WORDS
MAPPED NOVA 840	–	606 (390) WORDS
MAPPED NOVA 3	–	610 (394) WORDS

12. CASSETTES – SAME AS MAG TAPE

13. AUTO-RESTART AFTER POWER FAIL

MAPPED ECLIPSE	–	365 (245) WORDS
UNMAPPED ECLIPSE	–	325 (213) WORDS
MAPPED NOVA 840	–	423 (275) WORDS
MAPPED NOVA 3	–	441 (289) WORDS
UNMAPPED NOVA	–	403 (259) WORDS

14. OPERATOR MESSAGES

MAPPED ECLIPSE	–	404 (260) WORDS
UNMAPPED ECLIPSE	–	336 (222) WORDS
MAPPED NOVA 840	–	417 (271) WORDS
MAPPED NOVA 3	–	412 (266) WORDS
UNMAPPED NOVA	–	336 (222) WORDS

15. RTC – NO EXTRA WORDS LOAD IF SYSGEN'ED. IF NO RTC IS SYSGEN'ED, THE CLOCK IS JUST NOT STARTED.

16. PAPER TAPE READER

124 (84) WORDS IF ONE (1) SPTR

231 (153) WORDS IF TWO (2) SPTR'S

17. PAPER TAPE PUNCH

115 (77) WORDS IF ONE (1) SPTP

213 (139) WORDS IF TWO (2) SPTP'S

18. LINE PRINTERS

A. TABLES

1. FIRST DCH SLPT – 200 (128) WORDS UNMAPPED
302 (194) WORDS MAPPED

2. SECOND DCH SLPT - 162 (114) WORDS UNMAPPED
264 (180) WORDS MAPPED

3. FIRST PIO SLPT – 174 (124) WORDS UNMAPPED
276 (190) WORDS MAPPED

4. SECOND PIO SLPT – 156 (110) WORDS UNMAPPED
260 (176) WORDS MAPPED

B. DRIVER

1. DCH SLPT (IF 2 DCH SLPT'S, THEY SHARE SAME DRIVER)

MAPPED ECLIPSE	–	200 (128) WORDS
UNMAPPED ECLIPSE	–	153 (107) WORDS
MAPPED NOVA 840	–	203 (131) WORDS
MAPPED NOVA 3	–	201 (129) WORDS
UNMAPPED NOVA	–	174 (124) WORDS

2. PIO SLPT – NO EXTRA WORDS LOADED

19. CARD READERS

420 (272) WORDS IF ONE (1) SCDR (3 EXTRA WORDS IF NOVA)

731 (473) WORDS IF TWO (2) SCDR'S (6 EXTRA WORDS IF NOVA)

20. PLOTTERS

171 (121) WORDS IF ONE (1) SPLT

343 (227) WORDS IF TWO (2) SPLT's

21. MCA

A. TABLES – 264 (180) WORDS PER MCA SYSGEN'ED
(6 EXTRA WORDS PER MCA SYSGEN'ED FOR NOVA'S)

B. DRIVER (IF TWO MCA'S, THEY SHARE THE SAME DRIVER)

MAPPED ECLIPSE	–	676 (446) WORDS
UNMAPPED ECLIPSE	–	633 (411) WORDS
MAPPED NOVA 840	–	710 (456) WORDS
MAPPED NOVA 3	–	704 (452) WORDS
UNMAPPED NOVA	–	633 (411) WORDS

22. QTY

A. TABLES – 141 (97) WORDS

B. DRIVER

MAPPED ECLIPSE	–	1404 (772) WORDS
UNMAPPED ECLIPSE	–	1312 (714) WORDS
MAPPED NOVA 840	–	1444 (804) WORDS
MAPPED NOVA 3	–	1441 (801) WORDS
UNMAPPED NOVA	–	1341 (737) WORDS

23. ALM

A. TABLES AND ALM INIT CODE – 327 (215) WORDS

B. MODEM CONTROL – 114 (76) WORDS

C. DRIVER – USES QTY DRIVER

24. SECOND TELETYPE – 402 (258) WORDS

NOTE: FIRST TELETYPE HAS 454 (300) WORDS OF TABLES AND A 324 (212) WORD DRIVER, WHICH HAS SHARED BY BOTH STTY'S WHEN PRESENT

25. CORE DUMP FACILITY

MAPPED ECLIPSE	–	330 (216) WORDS
UNMAPPED ECLIPSE	–	233 (156) WORDS
MAPPED NOVA 840	–	333 (219) WORDS
MAPPED NOVA 3	–	336 (222) WORDS
UNMAPPED NOVA	–	233 (155) WORDS

BASE SIZES
****

ALL BASE OPERATING SYSTEM SIZES GIVEN BELOW ARE CONFIGURED IN THE SAME WAY.

NO CHANNELS
NO DISKS
NO EXTRA ANYTHING
1 STACK
1 TTY

MAPPED ECLIPSE	–	21252 (6874)
UNMAPPED ECLIPSE	–	16653 (7598)
MAPPED NOVA 840	–	21415 (8073)
MAPPED NOVA 3	–	22202 (9346) *
UNMAPPED NOVA	–	17341 (7905) *

GENERAL NOTES
\- - - - - - - - -

1. ON UNMAPPED SYSTEMS, CHANNELS ARE CONSIDERED PART OF USER SPACE, SO DON'T INCLUDE THEM IN THE OPERATING SYSTEM SIZE.

2. TO SIZE AN UNMAPPED OPERATING SYSTEM WHILE IT IS RUNNING, YOU MUST USE THE SYSTEM CALL '.MEM'. THIS WILL RETURN THE HIGHEST MEMORY ADDRESS AVAILABLE (HMA) TO THE USER. SUBTRACT HMA FROM THE MEMORY SIZE SPECIFIED AT SYSGEN TIME TO GET THE OPERATING SYSTEM SIZE.

3. ON MAPPED SYSTEMS, THE SYSTEM SIZE MUST BE ROUNDED UP TO THE NEXT HIGHEST MULTIPLE OF 1024 (DECIMAL) WORDS, SINCE THE LAST PAGE OF THE OPERATING SYSTEM WILL BE FILLED OUT WITH FROM 0 - 3 EXTRA SYSTEM BUFFERS, DURING SYSTEM INITIALIZATION TO AVOID WASTING THAT SPACE.

4. MAPPED SYSTEMS ARE SIZED WHILE THEY ARE RUNNING BY USING THE CLI GMEM' COMMAND. SUBTRACT SUM OF FG AND BG TOTALS FROM TOTAL MEMORY AGES TO GET OPERATING SYSTEM SIZE.

TRAPS

MAP PROTECTION VIOLATIONS

MESSAGE:

"TRAP=LOCATION AC0 AC1 AC2 AC3
BREAK (copy user program to "BREAK.SV")
R " (CLI)

REACTION:

COPY TRAP NUMBERS FOR PROGRAMMER

FPRINT BREAK.SV

(ALLOWS PROGRAMMER TO RELATE LOCATION TO INSTRUCTIONS)

PRINT RLDR LIST FOR PROGRAM

(ALLOWS PROGRAMMER TO RELATE BREAK.SV TO SOURCE CODE)

ERRORS

UTILITY PROGRAMS

ON ERROR,
SEND ERROR CODE TO CLI
RETURN TO CLI
CLI PRINTS –
"ERROR MESSAGE: PROGRAM"

E.G. "FILE NOT FOUND: RLDR"

DON'T THINK THAT THERE'S
NO RLDR.SV
BUT, WHILE THE PROGRAM
RLDR WAS
RUNNING,
IT COUNT'T FIND ONE OF
FILENAMES YOU PASSED AS AN
ARGUMENT.

ERRORS

PANICS

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

MESSAGE:
AC0 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.

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.

PANIC CODES CON'T

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

APPENDIX B

- Fortran Program Development Example

```
C     THIS IS A FORTRAN TEST FILE, THE PROGRAM DEMONSTRATES
C     THE CODE REPRESENTATION AT EACH PHASE OF PROGRAM DEVELOPMENT.
C     THE ALGORITHM ACCEPTS : LOWER LIMIT, UPPER LIMIT, AND INCRE-
C     MENT TO SUM A GROUP OF NUMBERS OVER.  THE RESULTS ARE
C     PRINTED OUT.
C
10    ACCEPT "LOWER LIMIT, UPPER LIMIT, INCREMENT ", LL, LH, INC
C
      SUM = 0.0
      DO 100 I = LL, LH, INC

        SUM = SUM + FLOAT(I)
        TYPE "I = ",I,"     SUM = ",SUM

100   CONTINUE

      GO TO 10
      END
```

ECLIPSE FORTRAN 5, VERSION 5.30 -- SUNDAY, JUNE 3, 1979 12:27:58 AM

TEST

```
C          THIS IS A FORTRAN TEST FILE, THE PROGRAM DEMONSTRATES
C          THE CODE REPRESENTATION AT EACH PHASE OF PROGRAM DEVELOPMENT.
C          THE ALGORITHM ACCEPTS : LOWER LIMIT, UPPER LIMIT, AND INCRE-
C          MENT TO SUM A GROUP OF NUMBERS OVER.  THE RESULTS ARE
C          PRINTED OUT.
C
10         ACCEPT "LOWER LIMIT, UPPER LIMIT, INCREMENT ", LL, LH, INC
C
           SUM = 0.0
           DO 100 I = LL, LH, INC

             SUM = SUM + FLOAT(I)
             TYPE "I = ",I,"      SUM = ",SUM

100        CONTINUE

           GO TO 10
           END
```

```
          ATTRIBUTES          POSITION   SIZE

-- STACK VARIABLES --

LL       INTEGER                1
LH       INTEGER                2
INC      INTEGER                3
I        INTEGER                4
SUM      REAL                   5

-- EXTERNAL SUBPROGRAMS --

.IACC           .FWRS           .FRDI           .TACC           .ITYP
.FWRI           .FWRR           .TTYP
```

```
000000'163710            SAVE     15              .MAIN
000001'000015
000002'030040            LDA      2,40,0                              LINE 7
000003'006001$           JSR      @.IACC
000004'166470            ELEF     1,72,1
000005'000072
000006'006002$           JSR      @.FWRS
000007'167470            ELEF     1,1,3           LL
000010'000001
000011'006003$           JSR      @.FRDI
000012'167470            ELEF     1,2,3           LH
000013'000002
000014'006003$           JSR      @.FRDI
000015'167470            ELEF     1,3,3           INC
000016'000003
000017'006003$           JSR      @.FRDI
000020'021402            LDA      0,2,3           LH
000021'031403            LDA      2,3,3           INC
000022'151112            SGEZ     2,2
000023'100400            NEG      0,0
000024'041407            STA      0,7,3
000025'030040            LDA      2,40,0
000026'006004$           JSR      @.TACC
000027'122050            FLDS     0,72,1          0.0                 LINE 9
000030'000072
000031'162250            FSTS     0,5,3           SUM
000032'000005
000033'021401            LDA      0,1,3           LL                  LINE 10
000034'041404            STA      0,4,3           I
000035'021404            LDA      0,4,3           I                   LINE 12
000036'102450            FLAS     0,0
000037'166050            FLDS     1,5,3           SUM
000040'000005
000041'104050            FAS      0,1
000042'166250            FSTS     1,5,3           SUM
000043'000005
000044'030040            LDA      2,40,0                              LINE 13
000045'006005$           JSR      @.ITYP
000046'166470            ELEF     1,55,1
000047'000055
000050'006002$           JSR      @.FWRS
000051'167470            ELEF     1,4,3           I
000052'000004
000053'006006$           JSR      @.FWRI
000054'166470            ELEF     1,52,1
000055'000052
000056'006002$           JSR      @.FWRS
000057'167470            ELEF     1,5,3           SUM
000060'000005
000061'006007$           JSR      @.FWRR
000062'006010$           JSR      @.ITYP
000063'021403            LDA      0,3,3           INC                 LINE 15
000064'031404            LDA      2,4,3           I
000065'143000            ADD      2,0
000066'041404            STA      0,4,3           I
000067'031403            LDA      2,3,3           INC
000070'151112            SGEZ     2,2
000071'100400            NEG      0,0
000072'031407            LDA      2,7,3
000073'111010            SLT      2,0
000074'000741            JMP      -37
000075'000705            JMP      -73                                 LINE 17
```

```
000076'127710        RTN

000077'046117        046117              LOWER LIMIT, UP
000100'053505        053505
000101'051040        051040
000102'046111        046111
000103'046511        046511
000104'052054        052054
000105'020125        020125
000106'050120        050120
000107'042522        042522
000110'020114        020114
000111'044515        044515
000112'044524        044524
000113'026040        026040
000114'044516        044516
000115'041522        041522
000116'042515        042515
000117'042516        042516
000120'052040        052040
000121'000000        000000
000122'000000        000000              0.0
000123'000000        000000
000124'044440        044440              I =
000125'036440        036440
000126'000000        000000
000127'020040        020040                   SUM =
000130'020040        020040
000131'020123        020123
000132'052515        052515
000133'020075        020075
000134'020000        020000
```

IDENTIFIER				REFERENCES
I	10	12	13	15
INC	7	10	15	
LH	7	10		
LL	7	10		
SUM	9	12	13	
10	7	17		
100	10	15		

```
NO COMPILATION ERRORS  --  TERMINATED AT 12:28:37 AM
```

```
TEST.SV LOADED BY RLDR REV 07.10 AT 00:44:08 06/03/79
 .MAIN
 FORT5
 TMIN
 NSAC3
 ITACC
 ITTYP
 GET                           NMAX      007132
 PUT                           ZMAX      000130
 UFMT                          CSZE      000000
 IOINI                          EST      000000
 IOTER                          SST      000000
 FREAD
 FWRIT                         .SLEF     000000
 DNFMT                         .RLEF     000000
 CVB                           ESV.Z     000002
 CVD                           .SP       000040
 IFILE                         .FP       000041
 DGCPC                         .SSE      000042
 LINES                         .SOV      000043
 IOCB                          .IACC     000050
 ILEN                          .TACC     000051
 IATT                          .ITYP     000052
 FOPEN                         .TTYP     000053
 NCAL                          .GCH      000054
 RTER                          .GREC     000055
 TRACE                         .PNM      000056
 TRTN                          .PCH      000057
 CONSO                         .PCR      000060
 ECODE
 FT
 SOVL
 FINIT
 ZERO
 STACK
 DNMAX
 DLEF
 DRTN
 F5END
```

.PREC	000061
.UFMT	000062
.FRDI	000063
.FRDL	000064
.FRDR	000065
.FRDC	000066
.FRDD	000067
.FRDX	000070
.FWRI	000071
.FWRL	000072
.FWRR	000073
.FWRC	000074
.FWRD	000075
.FWRX	000076
.FWRS	000077
.NFMT	000100
.CVB	000101
.CVD	000102
.ROUN	000103
.IFIL	000104
.IUCB	000105
.ILEN	000106
.IATT	000107
.FOP	000110
.FOPE	000111
.ERET	000113
.RTER	000114
.TRTN	000116
.CWCH	000120
.CRLF	000121
.CWRL	000122
.CNMO	000123
.CNMD	000124
.FT	000125
.GP	000126
ESV.S	000126
.RP	000127
USTAD	000400
.LEFD	000401
.LEFE	000401
.MAIN	000445
.F5	000602
TMIN	000716
?IFRD	001426
?IURD	001432
?IFWR	001451
?IUWR	001455
?IREA	001511
?IPRT	001517
?IPUN	001525
?IACC	001533
?ITYP	001544
?IDEC	001565
?IENC	001570
?TFRD	001625
?TFWR	001631
?TUWR	001640
?TACC	001643
?TTYP	001656
?TURD	001663
.SFMT	002223
.PCT	004572
FOPEN	005164

.NCAL	006112
.TRAC	006115
.SOVL	006453
.FSIS	006471
.FSIU	006471
I.SOV	006566
.FSIN	006570
I.SP	007035
I.SSE	007105
.NMAX	007131
.AFTE	007132
.REV.	010530
.SAC0	020016
.SAC1	024016
.SAC2	030016
.SAC3	034016
F?SQR	046015
F?EXP	046016
F?LOG	046017
F?ASC	046020
F?ATN	046021
F?PWR	046022
F?INT	046023
F?RTN	046024
L.I.N	100204
.LEFS	102460
F?FMT	106004
F?INM	106005
F?RCL	106006
F?RCS	106007
F?IFN	106010
F?ATT	106011
F?SEK	106012
F?STK	106013
F?EVT	106014
F?FNU	106025
F?MOP	106026
F?RCR	106027
F?W1K	106031
F?BLN	106032
F?BLC	106033
F?NPC	106034
F?RLN	106035
F?LEF	106036
F?ONO	106037
F?OAO	106040
F?TIO	106041
F?PRI	106042
F?EVN	106043
F?PNA	106044
F?ITO	106045
F?RTC	106046
F?TMO	106047
F?FPO	106050
F?MDV	106051
F?MEM	106052
F?IOP	106053
F?PTO	106054
F?ITL	106055
F?IRN	106056
F?IFV	106057
F?SOV	146001
F?DAT	146002
F?SBS	146003
F?EOB	146030
LN.RT	177777
LC.RT	177777
S.TAS	177777
PC.RT	177777

```
      10  ----    ----    ----    ----    ----    ----   000000 000753 ..............K
      20 000000 000000 000000 000000 000000 000000 000000 000000 ...............
***
      40 007035 000000 007105 006566 000000 000000 000000 000000 .....E.V........
      50 001533 001643 001544 001656 001011 001053 001114 001121 .L.#.D.....+.L.Q
      60 001141 001166 001360 001673 001677 001703 001707 001713 .A.v.P.;.?.C.G.K
      70 001717 002246 002252 002256 002262 002266 002272 002276 .O.&.*...2.6.:.>
     100 003220 003225 004211 004414 004453 004727 005037 005122 .........+.W...R
     110 005155 005164 005354 005366 005367 005642 006050 000000 .M.T.L.V.W.".(..
     120 006107 006133 006141 006155 006202 006347 000000 000000 .G.[.A.M...G....
     130 000000 000000 000000 000000 000000 000000 000000 000000 ...............
***
     400 000000 000130 000000 000000 007132 000716 177777 007132 ...X.....Z.N...Z
     410 000000 177777 177777 000410 000424 000424 177777 000445 ...............%
     420 177777 000000 002436 000000 015360 000000 000000 000000 .........P......
     430 000000 000000 000000 177777 000000 000000 000000 000000 ................
     440 000000 000000 000000 000000 000000 163710 000015 030040 ..........GH..0
     450 006050 166470 000072 006077 167470 000001 006063 167470 .(M8.:.?08...308
     460 000002 006063 167470 000003 006063 021402 031403 151112 ...308...3#.3.RJ
     470 100400 041407 030040 006051 122050 000072 162250 000005 ..C.0 .)$(.:D(..
     500 021401 041404 021404 102450 166050 000005 104050 166250 #.C.#..(L(...(L(
     510 000005 030040 006052 166470 000055 006077 167470 000004 ..0 .*M8.-.?08..
     520 006071 166470 000052 006077 167470 000005 006073 006053 .9M8.*.?08...;.+
     530 021403 031404 143000 041404 031403 151112 100400 031407 #.3.F.C.3.RJ..3.
     540 111010 000741 000705 127710 046117 053505 051040 046111 ...A.E/HLOWER LI
     550 046511 052054 020125 050120 042522 020114 044515 044524 MIT, UPPER LIMIT
     560 026040 044516 041522 042515 042516 052040 000000 000000 , INCREMENT ....
     570 000000 044440 036440 000000 020040 020040 020125 052515 ..I = ..      SUM
     600 020075 020000 010530 041517 050131 051111 043510 052040  = ..XCOPYRIGHT
     610 024103 024440 042101 052101 020107 042516 042522 040514 (C) DATA GENERAL
     620 020103 047522 050117 051101 052111 047516 026040 030471  CORPORATION, 19
     630 033463 026061 034467 032054 030471 033465 026061 034467 73,1974,1975,197
     640 033054 030471 033467 026061 034467 034056 040514 046040 6,1977,1978.ALL
     650 051111 043510 052123 020122 042523 042522 053105 042056 RIGHTS RESERVED.
     660 046111 041505 047123 042504 020115 040524 042522 044501 LICENSED MATERIA
     670 046040 026440 050122 047520 042522 052131 020117 043040 L - PROPERTY OF
     700 042101 052101 020107 042516 042522 040514 020103 047522 DATA GENERAL COR
     710 050117 051101 052111 047516 000000 000753 032433 025005 PORATION...K5.*.
     720 125132 000424 021001 025002 035010 054016 034001 175234 *Z..".*.:.X.8...
     730 000406 035004 031003 020415 127510 060202 035000 175220 ..:.2.!./H .:...
     740 054407 035004 031003 060177 002403 152400 000435 000000 Y.:.2. ...U.....
     750 000006 000414 000000 054777 034001 175014 060277 175015 ......Y.8... ?..
     760 010001 036770 041401 045402 051403 171000 034016 055004 ..=XC.K.S.R.8.Z.
     770 034762 024016 045010 025400 045006 175400 175500 055000 9R(.J.+.J....@Z.
    1000 025005 127240 045005 034001 175234 002002 141000 103510 *.. J.8.....B..H
    1010 060202 161000 036126 051422 041423 031536 021540 142427  .B.<VS.C.3^# E.
    1020 000405 142710 011536 031422 003423 031427 151014 000407 ..EH.^3...3.R...
    1030 101015 000771 045424 004420 025424 000760 101014 000405 ...YK...+..P....
    1040 021473 030447 142405 000757 020445 031531 153100 151112 #;1'E..O!%3YV@RJ
    1050 000753 030441 006113 054127 036126 021534 041536 031532 .K1!.KXW<V#\C^3Z
    1060 150015 002127 025531 125113 000420 153254 000406 030424 P..W+Y*K...V,..1.
    1070 025377 030125 133000 031004 006017 015477 006113 104110 *.0U6.2....?.K.H
    1100 036126 123000 041540 002127 006017 015077 006113 000771 <V&.C .W...?.K.Y
    1110 177775 000040 106007 004572 173110 036126 163770 000060 ... ....VH<VGX.0
    1120 000403 173110 036126 031536 025540 132415 000405 011536 ..VH<V3^+ 5....^
    1130 143010 143210 117710 006061 025427 125015 000765 030504 F.F..H.1+.*..U1D
    1140 006113 173110 036126 025427 125014 000756 025531 125113 .KVH<V+.*..N+Y*K
    1150 000753 030473 101014 142255 000402 000746 127113 000437 .K1;..D-...F.K..
    1160 142014 000466 031536 011536 143010 000462 173110 036126 D..63^.^F..2VH<V
    1170 031532 150015 000461 025531 125112 000406 021534 006017 3ZP..1+Y*J..#\..
```

```
                              .
                              .
                              .
                              .
                              .
              THIS GOES ON FOR A COUPLE OF WORDS
                              .
                              .
                              .
                              .
                              .
   7040 000000 000000 000000 000000 000000 000000 000000 000000 ................
***
   7130 000000 077777 000000 000000 000000 000000 000000 000000 ................
   7140 000000 000000 000000 000000 000000 000000 000000 000000 ................
***
   7410 000000 000000 000000 000000 000000 000000  ----   ----  ................
```

APPENDIX C

- Detailed Front Panel
- Optional Front Panel Exercise

FRONT PANEL

INTRODUCTION

The front panels of the NOVA line computers contain all the function switches and display all the information needed to operate them. As shown in the figure, all the consoles are essentially the same. The console at the top is for the NOVA computer, beneath it is the SUPERNOVA computer console, next is the console for NOVA 1200, NOVA 800, and NOVA 2 computers. Next is the console found on NOVA 3 computers. The bottom console is a turnkey console, which is available for all NOVA line computers. This console is designed for those computers that will be running in dedicated environments and contains only those switches needed to initiate processing. These switches, and the one light, operate exactly the same as those found on the other consoles.

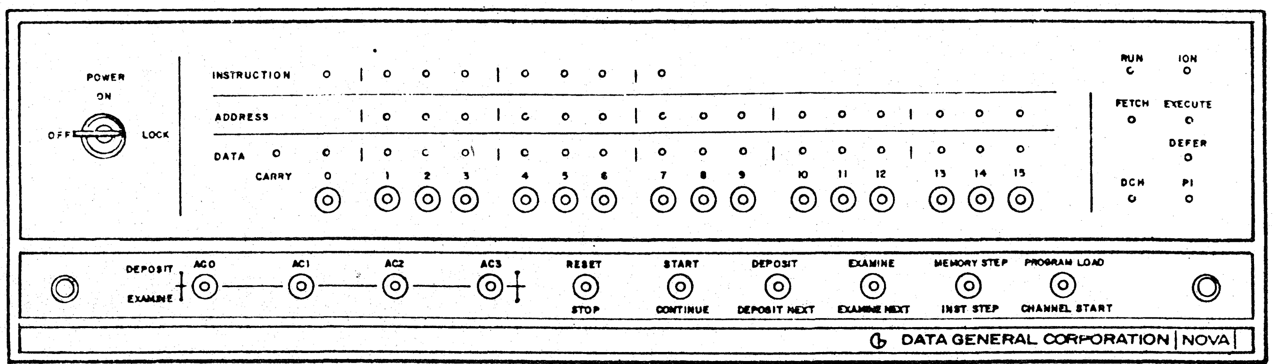
The function and data switches on the consoles allow the operator to perform many useful operations and the lights reflect the current state of the machine. If a light is lit, it means the corresponding bit is 1. If the light is not lit, the corresponding bit is 0. The lights and their meanings are described below.

FRONT PANEL LIGHTS

LIGHT	MEANING WHEN LIT	LIGHT	MEANING WHEN LIT
ADDRESS	These 15 lights display what is currently in the memory address register.	MEM PAR	The memory parity feature has detected a memory error. (NOVA 3 computers only.)
CARRY	The carry bit is 1.	MEM PWR	Power is being supplied to the semiconductor memories. (NOVA 3 computers only)
DATA	These 16 lights display what is currently on the memory bus.		
DCH	The next CPU cycle will be used by the data channel to gain access to memory. (NOVA, SUPERNOVA, and NOVA 3 computers only.)	ON	5V power is being supplied to the CPU. (NOVA 3 computers only.)
		OVERLAP	Two Accumulator-multiple operation format instructions are being executed out of read-only memory and the CPU is overlapping the execution of one with the fetching of the next. (SUPERNOVA computer only.)
DEFER	The next CPU cycle will be used to follow an indirection chain.		
EXECUTE	The next CPU cycle will be used to execute an instruction.		
FETCH	The next CPU cycle will be used to fetch an instruction.	PI	The next CPU cycle will be used to start a program interrupt by storing the program counter in location 0. (NOVA and SUPERNOVA computers only.)
INSTRUCTION	These 8 lights display the high-order 8 bits of the instruction just completed. (NOVA and SUPERNOVA computers only.)	PROTECT	The MAP feature is operating in user mode. (SUPERNOVA computers only.)
ION	The Interrupt On flag is 1.		
MAP B	Program map "B" or data channel map "B" is enabled. (NOVA 3 computers only)	RUN	The CPU is executing instructions or data is being transferred via the data channel.
MAP ENABLED	One of the two program maps is enabled and not inhibited or a data channel map is mapping addresses. (NOVA 3 computers only.)		

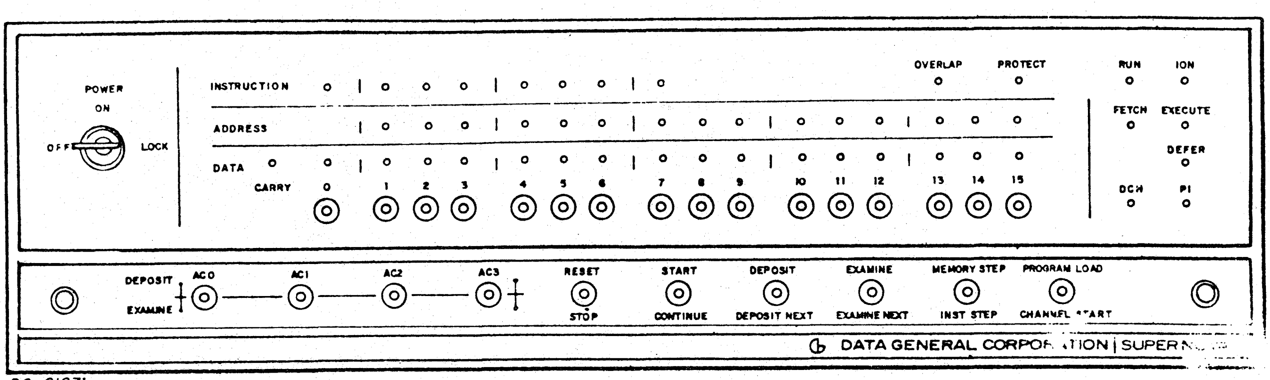
DG-01929

For the NOVA 3 series of computers, there is one row of lights that serves the function of both ADDRESS and DATA in the above table. The current contents of the program counter is displayed in these lights unless a console function is being performed.



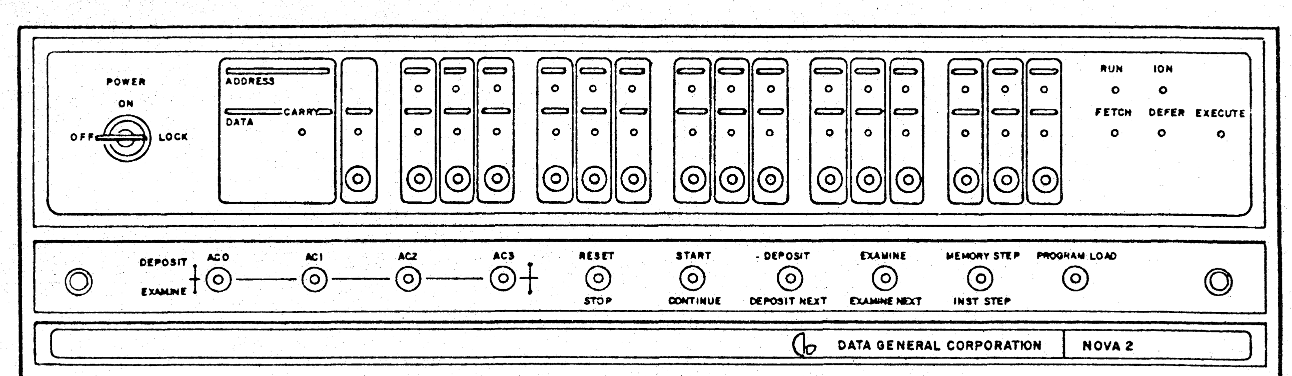
DG-01872

NOVA



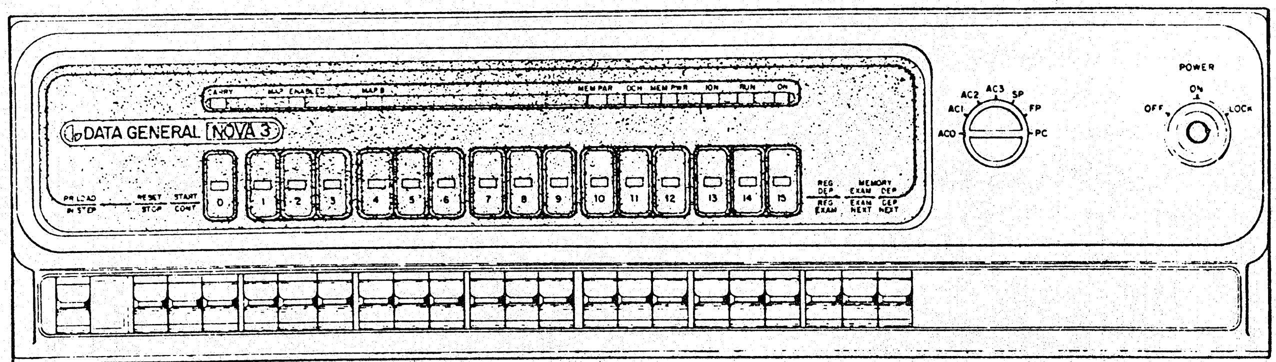
DG-01871

SUPERNOVA



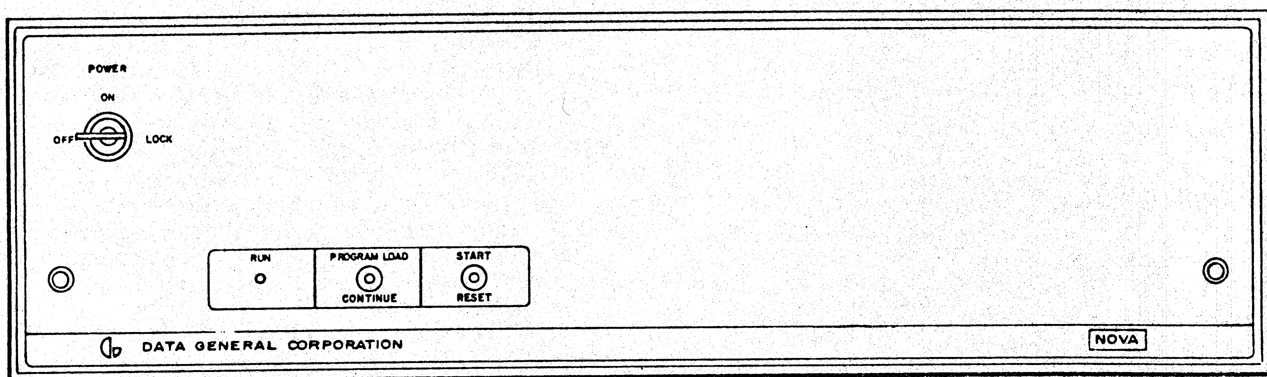
DG-01870

NOVA 800/1200 and NOVA 2



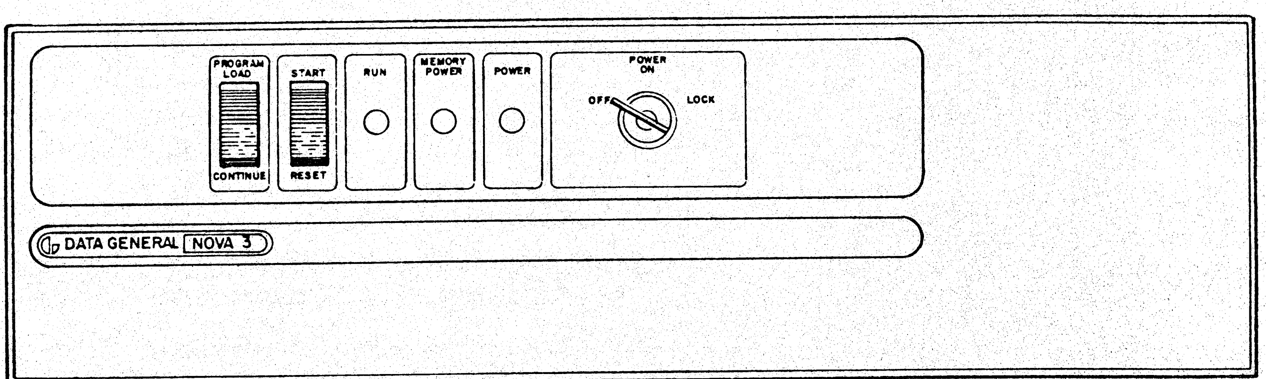
DG-01867

NOVA 3



DG-01869

NOVA TURNKEY



DG-01868

NOVA 3 TURNKEY

DATA SWITCHES

Beneath the data lights is a row of 16 switches. These switches are used to enter either data or addresses and can be read using the READ SWITCHES instruction. Only switches 1-15 are used for entering addresses. When these switches are in the up position, they represent a 1; when down, they represent a 0.

CONSOLE SWITCHES

In addition to the data switches, there are a number of function switches. These switches are spring loaded. When pushed up, they perform the function labeled above the switch, and when pushed down, they perform the function labeled below the switch. When released, these switches return to a neutral "off" position. The switches and their functions are explained below.

Accumulator Deposit--Examine

On all consoles except the NOVA 3 consoles, the left-hand four switches reference the four CPU accumulators. The switches are numbered 0-3 from left to right. Each switch affects only its corresponding accumulator. When one of these switches is pushed up, the current setting of the data switches is deposited into the corresponding accumulator. The data lights display the information placed in the AC. When one of these switches is pushed down, the contents of the corresponding accumulator are displayed in the data lights.

Reg Dep -- Reg Exam

For the NOVA 3 computers, the accumulator deposit and examine functions are performed by the combination of one function switch and a 7-position rotary switch. The seven registers available for depositing and examining are the four accumulators, the stack pointer, the frame pointer, and the program counter. When the function switch is pushed up, the contents of the data switches are deposited into the register indicated by the current setting of the rotary switch. As long as the switch is pushed up, the value indicated by the data switches is displayed in the lights. When the switch is released, the program counter is displayed in the lights.

When the function switch is pushed down, the contents of the register indicated by the current setting of the rotary switch are displayed in the lights. As long as the switch is held down, the value is displayed in the lights. When the switch is released, the program counter is displayed in the lights.

Reset--Stop

When this switch is pushed up, the RESET function is performed and an I/O RESET instruction is executed. The CPU is stopped after completing the current processor cycle. The Interrupt On flag, the 16-bit priority mask, and all Busy and Done flags are set to 0.

When this switch is pushed down, the STOP function is performed. The CPU is stopped after completing the current instruction and before executing the next instruction. If an I/O device requests an interrupt during the execution of the current instruction, it is honored before the CPU is stopped. All outstanding data channel requests are honored before the CPU is stopped. For the NOVA 3 series of computers, data channel requests are honored while the machine is in the stopped state. After the CPU is stopped, the address lights display the address of the next instruction to be executed and the data lights display the current contents of the memory bus.

Start--Continue

When this switch is pushed up, the START function is performed. The address indicated by data switches 1-15 is placed in the program counter and sequential operation of the processor begins with the word addressed by the updated value of the program counter.

When this switch is pushed down, the CONTINUE function is performed. Sequential operation of the processor continues from the current state of the machine.

Deposit--Deposit Next

When this switch is pushed up, the DEPOSIT function is performed. The current setting of the data switches is placed into the word addressed by the current value of the program counter. The updated value of the altered word is displayed in the data lights.

When this switch is pushed down, the DEPOSIT NEXT function is performed. The program counter is incremented by one and the current setting of the data switches is placed into the word addressed by the updated value of the program counter. The updated value of the program counter is displayed in the address lights and the updated value of the altered word is displayed in the data lights.

NOTE For the NOVA 3 computers, these functions are performed by the MEMORY DEP--DEP NEXT switch. As long as the switch is held in either the up or down position, the value indicated by the data switches is displayed in the lights. When the switch is released, the program counter is displayed in the lights.

Examine--Examine Next

When this switch is pushed up, the EXAMINE function is performed. The address indicated by data switches 1-15 is placed in the program counter. This value is displayed in the address lights. The contents of the word addressed by the program counter are then read and displayed in the data lights.

When this switch is pushed down, the EXAMINE NEXT function is performed. The current value of the program counter is incremented by one and the new value is displayed in the address lights. The contents of the word addressed by the updated value of the program counter are then read and displayed in the data lights.

NOTE For the NOVA 3 computers, these functions are performed by the MEMORY EXAM--EXAM NEXT switch. As long as the switch is held in either the up or down position, the value contained in the memory location is displayed in the lights. When the switch is released, the program counter is displayed in the lights.

Memory Step--Inst Step

When this switch is pushed up, the MEMORY STEP function is performed. The CPU performs a single processor cycle and stops. After the processor stops, the lights indicate the next cycle to be executed.

When this switch is pushed down, the INSTRUCTION STEP function is performed. The instruction contained in the word addressed by the current value of the program counter is executed and then the CPU is stopped. The address lights display the updated value of the program counter and the data lights display the contents of the memory bus.

Program Load

In the NOVA 1200, NOVA 800, and NOVA 2 computers, when this switch is pushed up, the PROGRAM LOAD function is performed if the Program Load option is installed on the machine. The contents of the bootstrap read-only memory are placed in memory location $0\text{-}37_8$ and a "JMP 0" instruction is performed. If the option is not installed, this switch has no effect.

In the SUPERNOVA computer, when this switch is pushed up, the PROGRAM LOAD function is performed. Thirty-three words are read from the device whose device code is set in data switches 10-15 on the console. These words are placed in locations $0\text{-}40_8$ of main memory. After the last word is read, a "JMP 40" instruction is performed.

NOTE For the NOVA 3 computers, the MEMORY STEP function has been deleted. The PROGRAM LOAD and INSTRUCTION STEP functions share the same function switch.

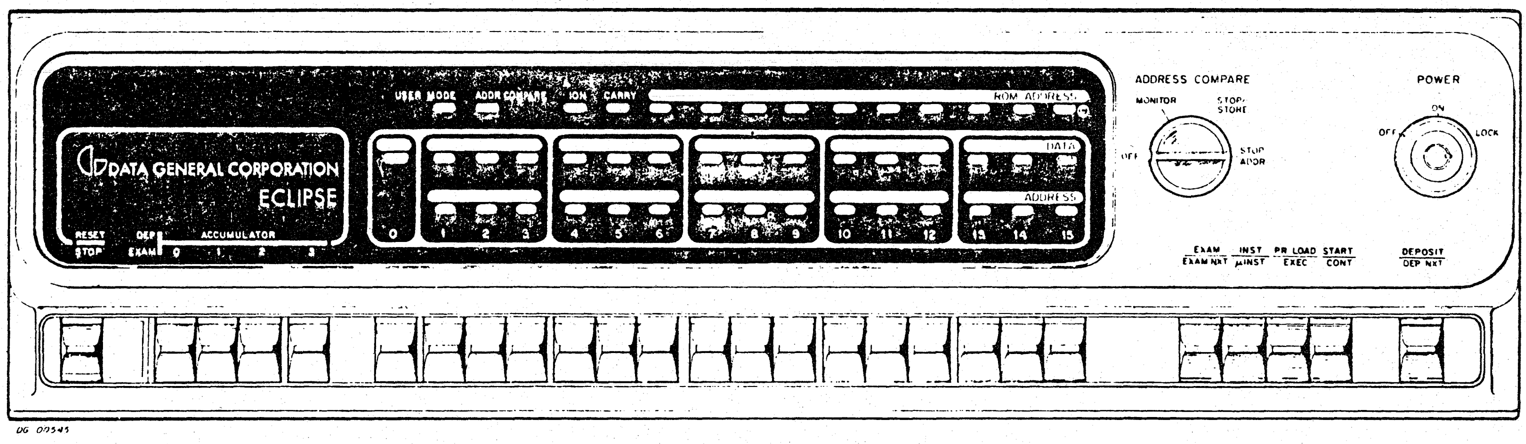
Channel Start

When this switch is pushed down, the CHANNEL START function is performed. A "JMP 377" instruction is placed in location 377_8 of main memory. Then a DATA IN A with a Start (DIAS) instruction is issued to the device whose device code is set in data switches 10-15 on the console. After the instruction is issued, a "JMP 377" instruction is performed.

Power

The POWER switch is a three position key switch. The three positions are labeled "OFF", "ON", and "LOCK". With the switch in the OFF position all power to the CPU is shut off and the machine will not run. Turning the switch to the ON position turns on the power and enables all the switches.

Turning the switch to the LOCK position enables the key to be removed. While the CPU is processing and the switch is in the LOCK position, all console functions are disabled. If the switch is turned to the LOCK position while the CPU is stopped or if the CPU executes a HALT instruction while the switch is in the LOCK position, all the function switches are enabled.



ECLIPSE LINE Computer Console

FRONT PANEL

INTRODUCTION

The front panel of the ECLIPSE line of computers contains all the functions' switches and displays all the information needed to operate the machine. The function and data switches allow the operator to perform many useful operations and the lights reflect the current state of the machine. If a light is lit, it means the corresponding bit is 1. If the light is not lit, the corresponding bit is 0. The lights and their meanings are described below.

LIGHT	MEANING WHEN LIT
USER MODE	The MAP feature is translating addresses in the user mode.
ADDR COMPARE	Operation of the machine is suspended because the comparison requested by the ADDRESS COMPARE switch has come up true.
ION	The Interrupt On flag is 1.
CARRY	The carry bit is 1.
ROM ADDRESS	These ten lights display the address in the micro-code of the next micro-instruction to be fetched.
DATA	These 16 lights display what is currently in general register 0 of the micro-code processor.
ADDRESS	These 15 lights display what is currently in the memory address bus.

CONSOLE SWITCHES

In a row along the bottom of the console are 26 switches. These are broken down into three groups; 5 function switches, 16 data switches, and 5 more function switches. The ten function switches are spring loaded. When pushed up, they perform one function, when pushed down, they perform another function. When released, these switches return to a neutral "off" position. The 16 data switches are two-position toggle switches. When in the up position, they represent a 1; when in the down position, they represent a 0. These switches have no neutral position. These 16 switches can be used to enter either data or addresses. If the switches are to be interpreted as data, all 16 data switches are used and they correspond to the bits in an internal 16-bit word. The leftmost switch of this group corresponds to bit 0 and the rightmost switch corresponds to bit 15. If the switches are to be interpreted as an address, only the rightmost 15 switches are used. When interpreted as an address, the second switch from the left is the high-order bit of the address and the rightmost switch is the low-order bit. All addresses coming from the console are treated as logical addresses.

Starting from the left of the console and proceeding to the right, the function switches and their meanings are described below.

Reset-Stop

When this switch is pushed up, the RESET function is performed and an I/O RESET instruction is executed. The CPU is stopped after completing the current processor cycle. The Interrupt On flag, the 16-bit priority mask, and all Busy and Done flags are set to 0. While in this state, the CPU will honor data channel requests.

When this switch is pushed down, the STOP function is performed. The CPU is stopped after completing the current instruction and before executing the next instruction. If an I/O device requests an interrupt during the execution of the current instruction, it is not honored before the CPU is stopped. All outstanding data channel requests are honored before the CPU is stopped. Data channel requests are continually honored while the machine is in the stopped state. After the CPU is stopped, the address lights display the address of the next instruction to be executed. The contents of the data lights are unpredictable.

Deposit-Examine

The next four switches are the accumulator DEPOSIT-EXAMINE switches. The switches are numbered 0-3 from left to right. Each switch affects only its corresponding accumulator. When one of these switches is pushed up, the current setting of the data switches is deposited into the corresponding accumulator. The data lights display the information placed in the AC.

When one of these switches is pushed down, the contents of the corresponding accumulator are displayed in the data lights.

Exam-Exam Nxt

When this switch is pushed up, the EXAMINE function is performed. The address indicated by data switches 1-15 is placed in the program counter. This value is displayed in the address lights. The contents of the word addressed by the program counter are then read and displayed in the data lights.

When this switch is pushed down, the EXAMINE NEXT function is performed. The current value of the program counter is incremented by one and the new value is displayed in the address lights. The contents of the word addressed by the updated value of the program counter are then read and displayed in the data lights.

Inst-μ/Inst

When this switch is pushed up, the INSTRUCTION STEP function is performed. The instruction contained in the word addressed by the current value of the program counter is executed and then the CPU is stopped. The address lights display the updated value of the program counter. The contents of the data lights are unpredictable.

> **NOTE** If the machine is stopped while in the user mode and the LOAD EFFECTIVE ADDRESS instruction is enabled for the current user, and a LOAD EFFECTIVE ADDRESS instruction is executed by use of the instruction step function, the action of the console is undefined.

When this switch is pushed down, the MICRO-INSTRUCTION STEP function is performed. The next micro-instruction in logical sequence is performed and the micro-code processor is stopped. The ROM address lights display the micro-code address of the next microinstruction to be fetched. The address lights display the contents of the memory address bus, and the data lights display the contents of the memory bus for the microinstruction just performed.

PR Load-Exec

When this switch is pushed up, the program load function is performed. The contents of the bootstrap read-only memory are placed in memory locations 0-37_8 and a "JMP 0" instruction is performed.

When this switch is pushed down, the EXECUTE function is performed. The current setting of the data switches is interpreted as an instruction and that instruction is executed as if it were in memory at the location specified by the program counter. After the instruction is stopped, the address lights display the updated value of the program counter. The contents of the data lights are unpredictable.

> **NOTE** If the machine is stopped while in the user mode and the LOAD EFFECTIVE ADDRESS instruction is enabled for the current user, and a LOAD EFFECTIVE ADDRESS instruction is executed by use of the execute function, the action of the console is undefined.

Start-Cont

When this switch is pushed up, the START function is performed. The address indicated by data switches 1-15 is placed in the program counter and sequential operation of the processor begins with the word addressed by the updated value of the program counter.

When this switch is pushed down, the CONTINUE function is performed. Sequential operation of the processor continues from the current state of the machine.

Dep-Dep Next

When this switch is pushed up, the DEPOSIT function is performed. The current setting of the data switches is placed into the word addressed by the current value of the program counter. The updated value of the altered word is displayed in the data lights.

When this switch is pushed down, the DEPOSIT NEXT function is performed. The program counter is incremented by one and the current setting of the data switches is placed into the word addressed by the updated value of the program counter. The updated value of the program counter is displayed in the address lights and the updated value of the altered word is displayed in the data lights.

Address Compare

The ADDRESS COMPARE switch is a four position rotary switch. The four positions are labeled "OFF", "MONITOR", STOP/STORE", and "STOP/ADDR". The functions of these four positions are described below.

Off

When the switch is in the OFF position, the ADDRESS COMPARE feature is disabled.

Monitor

When the switch is in the MONITOR position, it is possible to examine and monitor locations in memory while the CPU is running. When the switch is in this position, the contents of the memory location addressed by the current setting of the data switches is displayed in the data lights each time the location is accessed by the CPU. The data is not displayed until either the CPU accesses the location or the EXAM-EXAM NXT switch is pushed up. The data lights continue to display this information until either the contents of the addressed location are altered by the CPU or the setting of the data switches is changed. In the first case, the updated value of the location is displayed in the data lights. In the second case, the old data remains in the lights until either the CPU accesses the location addressed by the new data switch setting or the EXAM-EXAM NXT switch is pushed up. As soon as the CPU accesses the location addressed by the new switch setting or the EXAM-EXAM NXT switch is pushed up, the contents of the location addressed by the new switch setting will be displayed in the data lights.

Stop/Store

With the switch in the STOP/STORE position, the ADDRESS COMPARE feature will suspend the operation of the CPU if the CPU tries to alter the location whose address is set in the data switches. The addressed location is altered. The ADDR COMPARE light is lit to indicate that the ADDRESS COMPARE feature has suspended the operation of the machine. The contents of the data and address lights are unpredictable.

Stop/Addr

With the switch in the STOP/ADDR position, the ADDRESS COMPARE feature will suspend the operation of the CPU if the CPU tries to access the location whose address is set in the data switches. The addressed location is neither read nor written. The ADDR COMPARE light is lit to indicate that the ADDRESS COMPARE feature has suspended the operation of the machine. The contents of the data and address lights are unpredictable.

Power

The POWER switch is a three position key switch. The three positions are labeled "OFF", "ON", and "LOCK". With the switch in the OFF position, all power to the CPU is shut off and the machine will not run. Turning the switch to the ON position turns on the power, performs a RESET function, and enables all the switches. Turning the switch to the LOCK position allows the key to be removed. While the switch is in the LOCK position, all console functions except the MONITOR function of the ADDRESS COMPARE feature are disabled.

S200 RDOS USER LAB EXERCISE

MONDAY

Hardware Familiarization

In order to power up all devices in the system, the black breaker switch(es) located on the rear of the cabinet must be in the up position. Turn these on if necessary. All remaining power switches are located on the front of the machine. Turn on the Central Processing Unit (CPU) via the key switch; it should be in the "ON" position as the "LOCK" position will disable the front panel. It would seem appropriate to learn a little more detail about the most fundamental hardware unit, the front panel.

Front Panel Operation

You have one or two sets of lights, these are for display of addresses and data; examine, examine next, deposit, deposit next allow the display and modification of any memory location. Set the number switches to some non-zero value (i.e., 15 bit, octal); then switch to examine.

What happened?

Next, switch to examine next, note the lights in both positions of the switch; first the data is displayed. Then the address.

What does examine next do?

Repeat the above process with the deposit, deposit next switches to store the octal numbers 0 through 10 in the accumulators first and the remaining digits in the first memory locations. Don't let variation in front panel fool you; some have distinct examine/examine next and deposit/deposit next switches, others imploy a rotary selector switch. So, deposit the values; and then check them by examine/examine next.

Front Panel Operation

Let's execute a small program; the two instructions with their binary equivalents will execute a program which requires all of memory. The first instruction loads accumulator zero with the binary value of the second instruction (40401), located one memory location beyond this instruction. The second instruction stores accumulator zero in the memory location to be executed next.

LDA	0, . + 1	020401
STA	0,.+1	040401

Store these instructions in memory anywhere, but record their addresses.

____________	LDA 0, .+1	020401
____________	STA 0,. +1	040401

Now examine any other memory location, this will alter the Program Counter; now if you toggle in your start address and press start your program will execute, at that address. This is the only difference between START and CONTINUE.

What did your program do? Examine very low memory locations and large ones; what value do they contain?

What's in accumulator zero?

How did it get there?

Why did your program modify location zero? You began execution at some higher location, but the program wrote to location zero. Examine memory location 77777, then examine next; what happens when programs execute location 77777 + 1?

BOOTSTRAPPING

Power up the remaining equipment; terminals have black on line switches, the line printer switch is hiding on the lower right, and disk units have arrow boxes pointing to the disk they switch power to.

To be safe you should let your instructor show you disk loading; you need only slide the switch over and lift the black handle to disengauge the dust cover from the disk pack. For top loading drives, unlatch the drawer at the bottom or on the sides and slide it out. Pull the drawer wings out if applicable and, mount the disk by rotating the platter into position with the slot in the pack pointing toward the processor.

Put the dust cover on top, slide the drawer in and place the disk on-line. At this point, everything waits until the disk drive is ready.

Now, ready for bootstrap; load a 100033 into the number switches and depress stop, reset, program load. When you get FILENAME, you've accessed HIPBOOT on blocks zero and one.

```
FILENAME: BOOTSYS
DATE (M/D/Y)  Today's Date
TIME (H:M:S)  The time

R
```

RDOS is initialized and CLI.SV is executing. Now you have CLI to work with.

The LIST Command will report bookkeeping information about files, so try it. If things scroll too quickly; you can stop your console with control S(↑S) and resume with control Q. If you want an "R" prompt at any time hit ↑A.

BOOTSYS is a starter operating system and you will invoke it later to initialize an RDOS System; however, it is very small and should be replaced by a more appropriatly generated system. Look for all the names of possible operating systems with the following list command.

LIST –.SV –.OL

This will give you a clue as to more appropriate systems to BOOT; Nova 3 systems might be N3SYS or Eclipse systems ASYS or BSYS and all other Nova systems USYS or MSYS. To be sure of something appropriate, ask your instructor; he probably created the systems. Now type the following:

BOOT SYSTEMNAME↵

Once you've initialized your new system with a better device configuration, the CLI exercise may begin.

Several commands may be used to interrogate system parameter values, what do the following tell you.

GTOD↵

GSYS↵

REV SYSTEMNAME↵

GCIN↵ and GCOUT↵ executed in the foreground and background

The XFER Command can copy files, enter data, and transfer files to devices.

Type in: XFER/A $TTI(1) YOURNAME ↵

Anything you type in now is transferred into the file you have created, until you hit control Z. Enter some ASCII text to the file and hit control Z; the global slash A alters the XFER command to observe ASCII conventions and must be used with ASCII devices.

Now print YOURNAME: XFER/A YOURNAME $LPT

And again : PRINT YOURNAME

Put YOURNAME out to $TTO first using the XFER command and then using the TYPE command.

Let's transfer YOURFILE to another group in the class; use either paper tape or mag. tape depending upon what your system configuration is:

Paper Tape: You may transfer out using XFER/A YOURFILE $PTP, or PUNCH YOURFILE. ↵

How would you transfer back into the system from th paper tape reader ($PTR)

Mag. Tape

Mag Tape must be initialized prior to device access; so after loading a volume, powering up the drive, and setting it on-line, issue the following INIT command.

INIT MT0↵

Now you may transfer your file to mag tape file zero; the emphasis with the XFER command is that only the data is transferred and so files may be executed from mag. tape.

XFER/A YOURFILE MT0:0↵

And you may bring in a mag. tape file by switching around the arguments as the first is the source file which must exist on disk, the second argument is the destination file or device which is assumed not to exist in the disk file case thereby allowing RDOS to create a file. If a destination disk file does exist a global slash B must be used so that the new information is appended to the existent material.

XFER/A MT0:0 NEWFILE↵

Additionally, XFER may be used to transfer a set of information from any device to any other device. Go through all device transfers, a set of examples following will get you started.

XFER/A $TTI(1) $LPT↵

XFER/A MT0:0 $LPT↵

XFER/A $TTI(1) QTY:0↵ ; for systems with terminals only

XFER/A YOURFILE $PTP.

If you have difficulties with these, ask your instructor; however, usually problems arise from either the hardware not configured into the system or the system is not generated to support the configured hardware.

The log file, (F)LOG.CM, may be used to record CLI and master console communications; the foreground operations will record FLOG.CM but still use the same command formats

To start the log: LOG/H/T↵

You should have the "R" ready prompt; if not you have a LOG.CM file with a use count of 1 which must be cleared first. Try the CLEAR command, and then try deleting LOG.CM:

```
CLEAR/A/V/D      LOG.CM↵
DELETE/V         LOG.CM↵
```

Try, LOG/H/T again; CLI is recording to (F)LOG.CM . . . The global switches: H - for heading (date, time) and T - for trace which will expand CLI macros (.MC). Above the switches are somewhat more general: V – usually for verify, A – usually for permanent files (those which cannot be deleted.), in this case D – for devices.

Ending the log file report is by ENDLOG, if you have used the password implementation it must also accompany the ENDLOG. You'll have to ENDLOG prior to shutdown; but for now it will document your lab responses.

Let's implement a macro file using percent variables, indirect files and CLI constructions; the macro name must contain the .MC extension and any legitimate RDOS filename. Record and create the file with the XFER command.

```
.MC

MESSAGE "          : TIME", %TIME%,"DATE", %DATE%
MESSAGE "LIST OF ALL SAVE FILES"↵
LIST/E/A   –.SV↵
MESSAGE "DIRECTORY FILES"
DELETE DIRS
BUILD DIRS –.DR <SYS.DR, MAP.DR >/N
TYPE @DIRS@
```

Invoke your macro by typing its name; to correct it involves editing the file which you'll do later. In the meantime, pick a new name or delete the old and recreate it.

What does the build command do?

What do "@" and "%" signs do?

What makes one file or another a macro file?

To interrogate RDOS/CLI interpretation one may use commands with the MESSAGE command, all constructions will be expanded, etc. Use the following to further inspect CLI expanders.

Type: MESSAGE F < 1, 2, 3, 4, 5 >ILE ↵
What happened:

Type MESSAGE F(1, 2, 3, 4, 5)ILE
What happened:

How about together, what happens in each case, below:

MESSAGE (1, 2, 3), (A, B, C) ↵

MESSAGE < 1, 2, 3 > (A, B, C) ↵

MESSAGE (1, 2, 3) <A, B, C, > ↵

MESSAGE <1, 2, 3 > < A, B, C > ↵

Now try each of the above without a comma between the constructions. Any difference, What might constructions be useful for?

Some files are not written in ASCII; they're binary code of some sort depending upon their creation. Such files require special editors to alter, but under CLI one may use FPRINT to examine any disk file in a number of formats.

Type FPRINT SYSTEMNAME.SV ⮠

You will be watching the core resident RDOS scroll before your eyes; each word is numbered by line number plus its relative position on the line and the ASCII byte equivalent is on the right. You may notice some familiar bytes like the RDOS title or certain error messages.

Print the same file in hexidecimal from position 400 to position 2000 octal.

Print the same information at the line printer; and you've got an operating system binary file dump.

Let's take a core dump of a program . . . ; execute the program CLI, just type its name. Now hit control C, you have interrupted the program and caused a core image to be written into the file (F)BREAK.SV. Print it on the line printer with the FPRINT command. Then delete it. The numbers and ASCII interpretable code were in memory prior to ↑C.

End this session by typing period, then again. What time did the system have? If the time slot for lab is up; ENDLOG and release the master directory.

ENDLOG ⮠
MDIR ⮠
DPO
RELEASE DPO ⮠
MASTER DEVICE RELEASED

Power down all equipment, including cabinet breakers; please dispose of your line-printer paper.

S200
APPENDIX D

- STUDENT LABORATORIES

S200 RDOS User Laboratory Exercise

BOOTSTRAPPING LAB

The purpose of this lab is to enable you to employ the correct sequence for:

- Bootstrapping an operating system
- Initiating a program in the foreground
- Recovering from a system crash
- Powering down a system

BOOTSTRAPPING AN RDOS SYSTEM

Perform these steps to bootstrap the system:

1. Turn on power to the CPU, background and foreground terminals, and the disk drive.
2. Place a disk in the drive (see Instructor for help).
3. Power on the printer and put it online.
4. Set the CPU switches to: 100033, then press STOP and then RESET.
5. When the Ready light appears on the disk drive, press Program Load.
6. FILENAME? should appear as a prompt on the background terminal. Respond with the name of the operating system you want to boot in (ask your Instructor if necessary).
7. Enter today's date and the time (in 24-hour form) in response to the next two questions.
8. The R prompt that appears indicates that you are running the command line interpreter (CLI) through which you will communicate with RDOS.

INITIATING A PROGRAM IN THE FOREGROUND

Bring up CLI in the foreground in the following manner:

1. GMEM (Find out the available memory left after RDOS takes its share.)
2. SMEM 25 (Reserve 25K for CLI in the background and give the rest of memory to the foreground.)
3. EXFG/E CLI (Execute CLI in the foreground with equal priority.)

4. R prompt should appear now on the foreground terminal.

NOTE: The amount of memory you reserve for the background using SMEM will depend on the needs of the programs you run. You can run any other executable program in the foreground by typing:

EXFG/E PROGRAMNAME

Using the GMEM command, confirm that the memory has indeed been divided.

RECOVERING FROM A SYSTEM CRASH

A crash occurs when your system goes down (voluntarily or not) without the master directory (DP0 here) having been released.

First, crash the system by pressing the STOP switch on the front panel. Notice that at this point nothing can be entered on the keyboard.

Below are the steps you should follow to recover from a system crash:

1. Re-home the disk heads. There are two ways to do this:

   A. (Faster with practice)
   On the front panel:

   - Press STOP and then RESET
   - Switches to 001400 and deposit into AC0 (not the Memory Deposit switch)
   - Switches to 061333 – Deposit
   - Press instruction step.

OR

   B. Toggle the LOAD/RUN switch on the disk drive. First, press LOAD to bring the disk down. Then press RUN to bring it up to speed again.

2. Press STOP and then RESET

3. Switches to 100033 and press PROGRAM LOAD

4. Respond to the FILENAME? prompt

5. Type C when asked

6. Enter the date and time

7. Clear the file use counts by:
   CLEAR/A/V/D
   CLEAR/V CLI.<ER,OL>

ORDERLY SYSTEM POWER-DOWN

To power down the system in an orderly fashion:

1. CTRL F — (Bring down the foreground. Important: be certain that nobody is still operating in the foreground)

2. Release DPO — (or release %MDIR%) Wait for the response: MASTER DEVICE RELEASED

3. Power down the disk, other peripherals, and the CPU.

DAILY LAB START-UP PROCEDURE

At the beginning of each day's lab session - starting with the next lab — you should perform the following steps:

1. Bootstrap the machine and bring up CLI in the foreground as described in the BOOTSTRAP LAB.

2. LOG/H (Begin a log of the session.
   Background CLI activity is held in a file called LOG.CM
   Foreground activity is held in a file called FLOG.CM)

3. Proceed to the lab exercises.

PROCEDURE FOR THE END OF THE LAB SESSION

At the end of each day's lab session, you should perform the following steps prior to leaving:

1. ENDLOG (Close off your log file)

2. DIR DPO (If you're not there already)

3. Get a printout of your log and then erase the log file.
   If there are two of you at the terminal, get two copies before deleting the log.
   To do this:

on the B/G:	on the F/G:
PRINT LOG.CM	PRINT FLOG.CM
DELETE/V LOG.CM	DELETE/V FLOG.CM

4. Once logs of both grounds have been printed, power down using the steps described in the BOOTSTRAP LAB. After you bring the disk down (Load light will go on), remove the disk pack before powering off the drive.

5. Clean up all the excess printer paper.

S200 RDOS User Laboratory Exercise

CLI LAB 1

In this lab, you'll work with disk files through CLI. You'll make, keep track of, back up, destroy, and restore files using commands that will haunt you throughout your life with RDOS. You'll have CLI record your dialog and make one copy of this record file for you to keep to remember how you did all this great stuff. In this exercise, you will create 2 disk files. They are referenced as "S200XXX" in this print out. Where XXX represents your initials, please substitute your own filenames for them. If you are using a foreground console, substitute "$TTI1" for each use of "$TTI", and "$TTO1" for "$TTO", and "FLOG.CM" for "LOG.CM". You will need the RDOS User's Handbook to look up the following commands:

DELETE

DISK

ENDLOG

GTOD

LIST

LOG

MDIR

PRINT

RELEASE

SDAY

STOD

TYPE

XFER

BOOT UP THE SYSTEM

Follow the directions on page 149 of RDOS User's Handbook. Instructor will tell you the "FILENAME" of the system. Fill in the current date and time.

Although CLI is just a user program, and not built into the system, RDOS will start running CLI by default. You should see CLI's "R" prompt on your terminal by now.

RECORD THIS SESSION IN "LOG.CM"

Out with the old . . .

DELETE/V LOG.CM (If the file doesn't exist, that's good.)

In with the new . . .

Start logging with a header and a password.

LOG/H

Get the name of the current operating system:

GSYS

SET UP & CHECK THE RUNNING ENVIRONMENT.

Change time to zero.

STOD 0 0 0

Now, RDOS will track elapsed time of the CLI session.

Verify the new time with

GTOD

Speaking of time, type in a period ("."). CLI will now give you the time after each command.

Change the date to your birthday for this year (SDAY command).
Verify the change. Happy Birthday!

Now change back the date.

Check out disk space with

DISK

How many blocks are used? ____________________

How many blocks are unassigned? ________________

TRANSFER DATA FROM ONE FILE TO ANOTHER.

Make a copy of the contents of a file

XFER S200SHOW S200XXX

This creates S200XXX which contains a copy of the data in S200SHOW.

LIST/E S200XXX

The dates in the list output should match the SDAY you just did. They are the creation date and the date the file was last used.

Now,

LIST/A S200SHOW

The number is the length in bytes.

Do the two file lengths match?____________

How many bytes in each?____________

DISPLAY DATA ON OUTPUT DEVICES.

Display your file on the console:

XFER S200XXX $TTO ($TTO1 if you're at the F/G)

(The letter O, not the number 0.) Was it readable?____________

try

XFER/A S200XXX $TTO

The ASCII device switch (/A) is very important for the correct formatting of data for the system's character devices.

try

TYPE S200XXX

Is the output any different from the XFER/A?____________

Display S200XXX on the line printer by an XFER command with "$LPT" as the destination argument.

Now for a little creativity - You'll be making up some text for a file.

SENDING MESSAGES BETWEEN DEVICES

XFER some text to the line printer ($LPT) from the console keyboard ($TTI). Remember the /A (there are 2 ASCII devices involved here). Every line you type in from now on will be used by CLI as data and not as commands. You must signal the end of data with an ASCII END–OF–FILE character (CTRL Z).

Now try this transfer the wrong way (without /A). The only way to get CLI back to command mode now is to use a console interrupt (CTRL A).

Write some more to the line printer with

PRINT $TTI

(If the cursor goes to the top of the screen, hit the erase page key.)

The command PRINT Q is the same as XFER/A Q $LPT
and TYPE Q is the same as XFER/A Q $TTO

Displaying data on the console and printer is so common that CLI provides these shortcuts.

CREATING AND WRITING TO DISK FILES

XFER to a disk file (S200XXX.2) from $TTI. Notice that the delete (or rubout) key will erase a character if you make mistakes. You'll use this technique later in this lab to create indirect CLI command files.

Display S200XXX.2 on the line printer.
So far you've only been concerned with the contents of files. Now, you'll take a look at some of the bookkeeping info that RDOS also keeps on disk.

First, stop the printing out of the time by typing a period and hitting a carriage return (CR).

DISPLAYING BOOKKEEPING INFO ABOUT FILES ON DISK

THE CLI ALLOWS YOU TO GET A LIST OF FILES ON THE DISK, USING THE "LIST" COMMAND. TRY IT.

```
R
LIST
```

TO GET A HARDCOPY OF THE LIST ON A LINE PRINTER, ADD THE GLOBAL SWITCH /L TO THE LIST COMMAND

```
R
LIST/L
```

THE FILES LISTED ARE IN NO PARTICULAR ORDER. TO GET THEM SORTED ALPHABETICALLY, DO THE FOLLOWING:

```
R
LIST/L/S
```

THESE ARE NOT ALL THE FILES ON THE DISK, ONLY THE ONES WHICH ARE NOT PERMANENT (ALL THESE ARE "DELETE"–ABLE). TO GET A LIST OF "ALL" FILES ON THE DISK, APPEND THE /A GLOBAL SWITCH TO THE LIST COMMAND.

```
R
LIST/L/S/A
```

NOTE THAT NEW FILES HAVE APPEARED, NOTABLY SOME IN THE BEGINNING THAT START WITH A DOLLAR SIGN ($). THESE ARE THE RDOS NAMES OF THE DEVICES IN THE SYSTEM.

$CDR	CARD READER
$TTI	TELETYPE INPUT (ALSO CRT INPUT)
$TTO	TELETYPE OUTPUT (ALSO CRT OUTPUT)
$LPT	LINE PRINTER
$PTP	PAPER TAPE PUNCH
$PTR	PAPER TAPE READER

THE DEVICES THAT ARE LISTED ON YOUR OUTPUT MAY NOT AGREE WITH THE ONES I HAVE LISTED ABOVE. THEY DEPEND ON THE SYSGEN THAT WAS DONE FOR THIS PARTICULAR SYSTEM. IF YOU DON'T ASK FOR A CARD READER, $CDR WILL NOT APPEAR. THIS INSURES THAT THE RDOS SYSTEM YOU GENERATE IS THE SMALLEST POSSIBLE FOR A GIVEN CONFIGURATION.

BY NOW YOU MUST BE WONDERING WHAT THE OTHER NUMBERS AND LETTERS ARE ON YOUR OUTPUT. LET'S LOOK AT ONE IN PARTICULAR. THE LINE PRINTER.

```
$LPT          0          RAP
```

$LPT — THE NAME OF THE FILE (OF COURSE)

0 — BYTE COUNT (MEANINGLESS FOR A DEVICE)

RAP — THE ATTRIBUTES OF THE FILE

- R – READ PROTECTED. RDOS WILL PREVENT YOU FROM READING THE LINE PRINTER
- P – PERMANENT. THE $LPT CANNOT BE DELETED FROM THE SYSTEM
- A – ATTRIBUTE PROTECTED. NORMALLY, THE ATTRIBUTES CAN BE CHANGED FROM THE CLI. THE "A" ATTRIBUTE PREVENTS CHANGING ANY OF THE ATTRIBUTES.

RDOS KNOWS MORE ABOUT EACH FILE THAN IT IS TELLING YOU ABOUT. TO DETERMINE EVERYTHING RDOS KNOWS ABOUT A FILE, APPEND THE /E GLOBAL SWITCH TO THE LIST COMMAND.

```
R
LIST/L/S/A/E
```

THIS COMMAND LISTS ALL FILES ON DISK, SORTED ALPHABETICALLY, LISTED ON THE LINE PRINTER, AND TELLS YOU EVERYTHING RDOS KNOWS ABOUT THE FILE. LET'S LOOK AT ONE IN PARTICULAR, THE BOOTSYS.SV. THE NUMBERS THAT ARE GIVEN BELOW MAY NOT CORRESPOND EXACTLY TO THE ONES ON YOUR LIST COMMAND.

```
BOOTSYS.SV     9430     SD   05/15/75 09:34 001204   0
```

BOOTSYS.SV	THE NAME OF THE FILE. THE .SV EXTENSION MEANS IT IS AN EXECUTABLE "SAVED" FILE.
9430	BYTE COUNT. THERE ARE 9430 BYTES ON THE DISK USED TO STORE THE FILE.
SD	S=SAVED FILE (IGNORE THE D FOR RIGHT NOW)
05/15/75	FILE WAS CREATED MAY 15, 1975 AT
09:34	9:34 AM. IT WAS LAST ACCESSED ON
05/15/75	MAY 15, 1975.
001204	THE STARTING DISK BLOCK ADDRESS
0	USE COUNT. NO ONE IS CURRENTLY USING THE FILE. LOOK AT USE COUNT FOR CLI.SV. IT IS IN USE BY YOU RIGHT NOW.

MANY TIMES YOU ARE NOT INTERESTED IN ALL THE FILES ON THE DISK, ONLY CERTAIN ONES. YOU CAN DO THIS BY PASSING AN ARGUMENT TO THE LIST COMMAND.

```
R
LIST/A BOOTSYS.SV
```

WHAT HAPPENED?____

WHAT HAPPENS WHEN YOU DO THE FOLLOWING?

```
R
LIST/A BOOTSYS
```

____

THE DASH (–) CONVENTION IS USEFUL IN THIS CASE BECAUSE IT MATCHES ANY SEQUENCE OF ASCII CHARACTERS. DO THE FOLLOWING:

```
R
LIST/A CLI.–
```

WHICH FILES ARE LISTED?____________________________________

____

DO THE FOLLOWING

R
LIST/A C–.–

WHICH FILES ARE LISTED?____________________________________

____

HOW WOULD YOU GET A LIST OF ALL "SAVED" (EXECUTABLE) FILES?

____________________________. TRY IT. DOES IT WORK?

HOW ABOUT ALL FILES THAT HAVE THE LETTER "R" IN THEM?

___________________________________. TRY IT. DOES IT WORK?

TAKE SOME TIME TO TRY SOME OTHER FILE SEARCHES USING THE – AND * TEMPLATES.

'XFER'ING TO MAG TAPE

IF YOU HAVE A BLANK MAG TAPE (MAKE SURE IT IS BLANK AND HAS THE WRITE RING IN IT), LOAD IT ON THE SYSTEM AND DO THE FOLLOWING:

```
R
INIT MT0
```

(NOTE: MT0:0 IS MT ZERO : ZERO)
THIS NOTIFIES THE SYSTEM THAT YOU HAVE A MAG TAPE LOADED.

NOW

```
R
XFER/A S200XXX MT0:0
```

THE TAPE SHOULD MOVE, FINALLY

```
R
XFER/A MT0:0 $LPT
```

IT SHOULD PRINT ON THE LINE PRINTER.

DO ONE LAST THING. TRY TO XFER TO S200XXX AGAIN.

```
R
XFER/A $TTI S200XXX
```

WHAT ERROR MESSAGE DO YOU GET?______________________________

WHAT DOES THIS TELL YOU?____________________________________

USE THE XFER/A COMMAND TO CREATE ANOTHER TEST FILE

R
XFER/A $TTI TEST2 (TEST2F and $TTI1 in F/G)

TYPE IN SOMETHING AND HIT A CONTROL Z.

DO A LIST/E COMMAND ON THE FILE TEST2.

WHAT IS THE BYTE COUNT?________________ ATTRIBUTES?________

HOW MANY CHARACTERS DID YOU TYPE?__________INCLUDING CR.

NOW DELETE THE FILE

R
DELETE/V/C TEST2

WHAT IS THE /V SWITCH FOR?________________________________

WHAT IS THE /C SWITCH FOR?________________________________

NOW DO A LIST/E TEST 2

WHAT HAPPENED?____

CREATE THE FILE TEST2 AGAIN IN THE SAME MANNER.

DO ANOTHER LIST/A ON IT. YOU SHOULD GET WHAT YOU GOT BEFORE (PERHAPS WITH A DIFFERENT BYTE COUNT)

RENAMING ALLOWS YOU TO CHANGE THE NAME OF A FILE. IT IS DONE WITH THE "RENAME" COMMAND. LET'S RENAME TEST2 TEST3.

R
RENAME TEST2 TEST 3 (TEST2F TEST3F for F/G)

DO A LIST TEST–.– TO VERIFY THAT TEST2 HAS DISAPPEARED.

NOW CREATE TEST2 AGAIN BY TRANSFERRING TEST3 TO TEST2

R
XFER/A TEST3 TEST2

DO A LIST/E ON BOTH. THEY SHOULD HAVE THE SAME BYTE COUNT.

TO MAKE SURE THEY ARE THE SAME, DO A FILE COMPARISON ON THEM.

R
FILCOM TEST2 TEST3

VERIFY THAT THEY ARE THE SAME. IF THERE ARE ANY DIFFERENCES, THEY WILL BE PRINTED ON THE CONSOLE. IF NO RESPONSE, THEY ARE THE SAME. TO TEST THIS OUT, DELETE TEST3 AND TYPE INTO IT (XFER/A $TTI TEST3) THE SAME MESSAGE, VARIED BY ONE (1) LETTER, THEN REPEAT FILCOM.

THE FOLLOWING INTEROGATIVE COMMAND WILL DISPLAY THE REVISION NUMBER OF ANY EXECUTABLE PROGRAM FOR WHICH A REVISION HAS BEEN DEFINED. TRY.

R
REV CLI

WHAT REV OF CLI ARE YOU USING?________________________________

CONSTRUCTIONS USING PARENTHESIS AND ANGLE BRACKETS CAN LEAD TO CONFUSION. THE MESSAGE COMMAND CAN BE USED TO EXPLOIT THE INTERPRETATION OF THESE CHARACTERS; TYPE THE FOLLOWING:

R
MESSAGE < 1,2,3,4 >(A,B,C,D)

WHAT HAPPENED?____

PLAY WITH THE CONSTRUCTIONS BY VARYING COMBINATIONS.

WHEN YOU'VE HAD ENOUGH, TERMINATE THIS SESSION.

WHEN YOU'RE DONE FOR THE DAY, TYPE ENDLOG.

YOU CAN THEN PRINT OUT A COPY OF YOUR LOG BY:

PRINT LOG.CM (FLOG.CM for F/G)

RELEASE THE MASTER DIRECTORY WITH THE FOLLOWING:

RELEASE %MDIR%

PLEASE POWER DOWN ALL EQUIPMENT AND DISCARD ANY EXTRA LINE PRINTER PAPER; YOUR INSTRUCTOR WILL HAVE A VERY LARGE SMILE . . .

S200 RDOS User Laboratory Exercise

CLI LAB 2 – FILE TYPES

INTRODUCTION:

RDOS HAS THREE (3) DIFFERENT KINDS OF FILE STRUCTURES – SEQUENTIAL, RANDOM AND CONTIGUOUS. WHEN A USER WISHES TO CREATE HIS OWN FILES, THE DECISION HE MAKES FOR WHICH TYPE HE WILL USE CAN GREATLY AFFECT HOW HIS SYSTEM WILL PERFORM. THIS SECTION WILL TEACH ALL THREE TYPES, HOW THEY ARE IMPLEMENTED BY RDOS, AND WHEN TO USE EACH ONE.

THE SMALLEST UNIT ON THE DISK THAT THE DISK CONTROLLER CAN ACCESS IS CALLED A "BLOCK". THIS PHYSICALLY CORRESPONDS TO A SECTOR ON THE DISK (IF YOU DON'T KNOW WHAT THIS MEANS, IT DOESN'T MATTER). A BLOCK CONTAINS 256 (DECIMAL) = 400 (OCTAL) 16-BIT WORDS, OR 512 DECIMAL, 1000 OCTAL 8-BIT BYTES.

RDOS MAINTAINS A MAP DIRECTORY WHICH CONTAINS 1 BIT FOR EVERY BLOCK ON THE DISK TO INDICATE THAT PARTICULAR BLOCK'S STATUS.

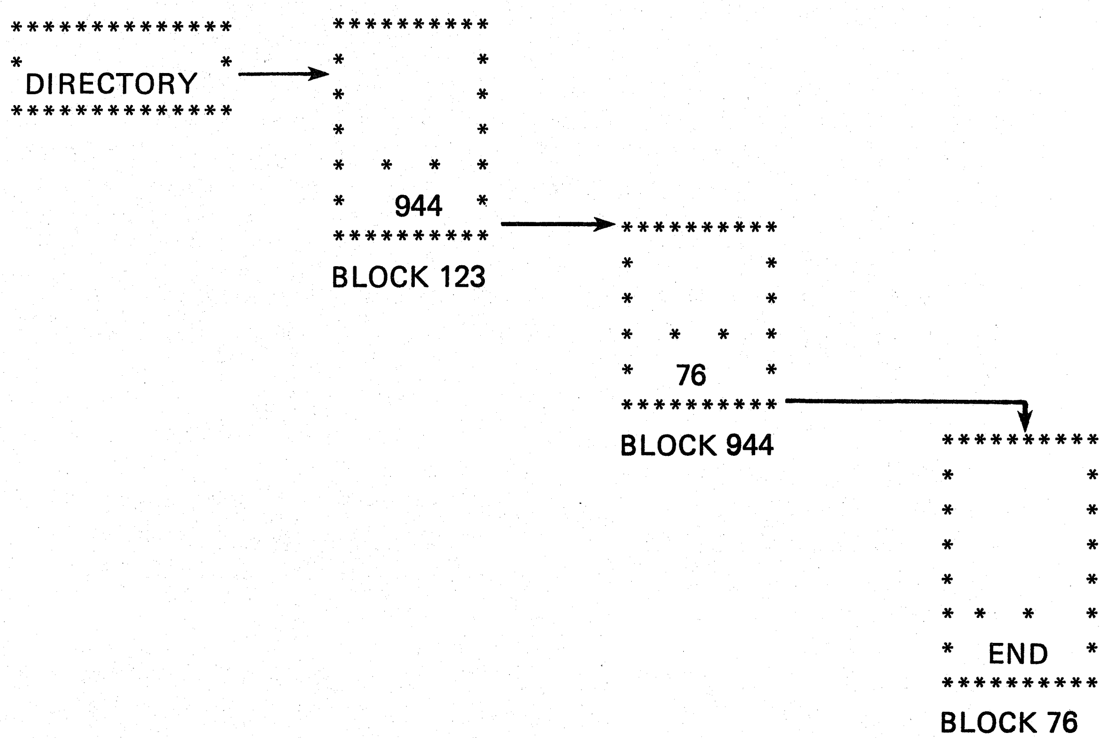
0 = FREE
1 = IN USE

IN FACT, THE "DISK" COMMAND JUST COUNTS THE NUMBER OF 1's and 0's IN THE MAP DIRECTORY, AND OUTPUTS THE ANSWER. THUS, SCATTERED THROUGHOUT THE DISK ARE FREE AND USED BLOCKS.

DISK BLOCKS ARE NUMBERED 0 THROUGH N WHERE N IS THE NUMBER OF BLOCKS ON THE DISK. THEREFORE THE USER DOES NOT HAVE TO THINK IN TERMS OF CYLINDER/HEAD/SECTOR, ONLY A NUMBER FROM 0 TO N.

SEQUENTIAL FILES

THE FIRST FILE STRUCTURE WE WILL LOOK AT IS SEQUENTIAL FILES. SEQUENTIAL FILES ARE LIKE MAG TAPE. THAT IS, IF YOU WANT TO GET TO ANY PART OF THE FILE, YOU MUST READ ALL DATA UP TO THAT POINT. THERE IS A LOW OVERHEAD IN THIS FILE STRUCTURE MAKING IT IDEALLY SUITED FOR TRANSACTION LOGGING, WHICH COMES IN SEQUENTIALLY AND IS PROCESSED SEQUENTIALLY. ALSO, SOURCE PROGRAM FILES FOR ASSEMBLY, FORTRAN, ALGOL, COBOL, RPG, ETC. ARE GOOD CANDIDATES FOR THIS FILE STRUCTURE. SEQUENTIAL FILES ARE STORED RANDOMLY ON THE DISK, AS SHOWN BELOW.



AS CAN BE SEEN FROM ABOVE, THE DIRECTORY POINTS TO THE FIRST BLOCK, AND EACH BLOCK'S LAST WORD POINTS TO THE NEXT BLOCK IN THE CHAIN. THERE IS A FLAG IN THE LAST WORD OF THE LAST BLOCK TO INDICATE THE END OF THE CHAIN. ALSO NOTE THAT THE BLOCKS ARE RANDOMLY SCATTERED THROUGHOUT THE DISK.

QUESTIONS:

IS THIS FILE EXPANDABLE?______________________________

HOW COULD RDOS DO IT?______________________________

____

GIVEN THE FACT THAT THE LINK TO THE NEXT BLOCK IS A 1-WORD ADDRESS HOW MANY USABLE BYTES AND WORDS ARE THERE IN EACH DISK BLOCK?

________________ BYTES OR ________________ WORDS.

WHY WOULD THIS FILE STRUCTURE NOT BE USEFUL FOR KEEPING A REAL-TIME INVENTORY SYSTEM? ________________________________

____

SEQUENTIAL FILE EXERCISE

THERE ARE TWO WAYS TO CREATE A SEQUENTIAL FILE. THE FIRST WAY IS THE "CREATE" COMMAND.

```
R
CREATE TEST4     (TEST4F on the F/G)
```

NOW DO A LIST COMMAND AND RECORD THE ATTRIBUTES OF THE FILE. ________

THE WAY YOU CAN TELL THIS IS A SEQUENTIAL FILE IS THAT IT HAS NO ATTRIBUTE THAT SAYS WHAT THE FILE STRUCTURE IS. THAT MEANS THE FILE IS SEQUENTIAL "BY DEFAULT".

NEXT, DELETE TEST4 AND VERIFY THAT IT IS GONE.

WE WILL USE THE SECOND METHOD FOR CREATING SEQUENTIAL FILES. BEFORE YOU DO, THOUGH, DO A DISK COMMAND AND DETERMINE HOW MANY BLOCKS HAVE BEEN USED.

________________ FREE ________________ USED

THE "XFER" COMMAND CREATES SEQUENTIAL FILES BY DEFAULT, SO WE WILL USE IT TO CREATE A SEQUENTIAL FILE

```
R
XFER/A $TTI TEST4     ($TTI1  TEST4F)
```

NOW TYPE IN A TEST MESSAGE FOLLOWED BY A CONTROL Z

DO A LIST COMMAND AND RECORD THE ATTRIBUTES ________________

NOW DO A DISK COMMAND ________________ FREE ________________ USED

HOW MANY DISK BLOCKS WERE USED? ________________________

DOES THIS MAKE SENSE? ________________________

CONTIGUOUS FILES

THE SECOND KIND OF DISK FILE STRUCTURE WE WILL EXAMINE ARE CONTIGUOUS FILES. CONTIGUOUS FILES ARE SO NAMED BECAUSE THE DISK BLOCKS WITHIN THEM ARE ARRANGED CONTIGUOUSLY ON THE DISK AS SHOWN BELOW.

```
*************     ***************************************
*           *     *      *       *        *        *      *
* DIRECTORY * ---->*      *       *        *        *      *
*************     *      *       *        *        *      *
                  *      *       *        *        *      *
                  *      *       *        *        *      *
                  ***************************************
```

PHYSICAL BLOCK #	555	556	557	558	559
LOGICAL BLOCK #	0	1	2	3	4

ALTHOUGH THIS FILE PHYSICALLY RESIDES AT BLOCK # 555 - 559, THE USER DOESN'T KNOW OR CARE WHERE ON THE DISK IT IS. THE USER ONLY REFERENCES BLOCK #0, 1, 2, 3, OR 4. RDOS DOES THE REST. THIS IS EASY FOR RDOS, BECAUSE IF THE USER WANTS BLOCK #3 IT IS AN EASY CALCULATION FOR RDOS TO DETERMINE WHERE ON THE DISK IT IS.
555+3=558.

FOR THIS REASON, CONTIGUOUS FILES OFFER THE FASTEST DISK ACCESS POSSIBLE. CRITICAL REAL-TIME SYSTEMS USUALLY USE THIS FILE STRUCTURE.

A DISADVANTAGE OF CONTIGUOUS FILES IS THAT THEY ARE NOT EXPANDABLE. THEIR SIZE IS FIXED AT TIME OF CREATION. WHY DO YOU SUPPOSE THIS IS TRUE?

____

AN ADVANTAGE OF CONTIGUOUS FILES IS THAT THERE IS NO EXTRA DISK SPACE WASTED IN OVERHEAD POINTING TO THE NEXT BLOCK AS IS TRUE WITH SEQUENTIAL FILES. THUS IN A CONTIGUOUS FILE OF 3 BLOCKS, HOW MANY WORDS OF STORAGE ARE AVAILABLE. HOW ABOUT IN A SEQUENTIAL FILE OF THE SAME LENGTH?

CONTIGUOUS______________________ SEQUENTIAL______________________

CONTIGUOUS FILES ARE USEFUL IN SITUATIONS WHERE FAST ACCESS IS REQUIRED, BUT THE DATA BASE DOES NOT GROW. IN RDOS, OVERLAYS ARE ALWAYS CONTIGUOUS FILES. ANOTHER USER APPLICATION MIGHT BE WHERE A PROGRAM KEEPS TRACK OF 100 DIFFERENT ELECTRIC UTILITY STATIONS, WHERE EACH STATION REPORTS CIRCUIT BREAKER STATUS (ON OR OFF) AND VARIOUS VOLTAGES AND CURRENTS WITHIN THE STATION.

WHAT MIGHT BE OTHER APPLICATIONS OF CONTIGUOUS FILES?

____

____

CONTIGUOUS FILE EXERCISE

THE COMMAND TO CREATE A CONTIGUOUS FILE IS "CCONT". WE WILL CREATE A CONTIGUOUS FILE OF 10 BLOCKS LONG. BEFORE WE DO, LET'S DO TWO THINGS.

FIRST, DELETE TEST4 FROM BEFORE

```
R
DELETE/V/C TEST4
```

NOW DO A DISK COMMAND TO SEE HOW MANY BLOCKS ARE IN USE.

```
R
DISK
```

FREE__________IN USE__________

TO CREATE A CONTIGUOUS FILE, DO THE FOLLOWING:

```
R
CCONT TEST4 10
```

DO A DISK COMMAND. HOW MANY BLOCKS DID IT USE?

THE ATTRIBUTE THAT SAYS IT IS A CONTIGUOUS FILE IS "C".
DO A LIST/E ON TEST4.

ATTRIBUTES ____________________ STARTING DISK BLOCK__________

IT WAS STATED BEFORE THAT ALL RDOS OVERLAYS FILES ARE CONTIGUOUS. ALL OVERLAY FILES HAVE A .OL EXTENSION. DO A

```
R
LIST/A/E  –.OL
```

AND VERIFY THAT THEY ALL HAVE A "C" AS AN ATTRIBUTE.

IN ORDER FOR YOU TO BE ABLE TO CREATE A CONTIGUOUS FILE, THERE MUST BE ENOUGH CONTIGUOUS BLOCKS AVAILABLE.

XFER CAN ALSO BE USED TO CREATE A CONTIGUOUS FILE BY USE OF THE "/C" LOCAL SWITCH. TRY TO CREATE A CONTIGUOUS FILE AND NOTE THE ERROR THAT OCCURS:

XFER/A $TTI S200XXX.3/C

REMEMBER THAT THE SIZE OF A CONTIGUOUS FILE IS DETERMINED AT THE TIME OF FILE CREATION AND CANNOT BE ALTERED. WHEN YOU ATTEMPT TO CREATE A CONTIGUOUS FILE BY XFER'ING FROM $TTI, THE CLI (HENCE RDOS) CANNOT TELL HOW MUCH CONTIGUOUS DISK SPACE TO ALLOCATE FOR THE FILE; RDOS DOES NOT KNOW HOW MUCH DATA YOU'RE GOING TO TRANSFER. THEREFORE, A CONTIGUOUS FILE CAN ONLY BE CREATED BY XFER WHEN THE XFER IS MADE FROM AN EXISTING DISK FILE. SO TO CREATE YOUR CONTIGUOUS FILE WITH XFER:

```
XFER/A       $TTI       DUMMY

XFER/A       DUMMY      S200XXX.3/C

DELETE/V     DUMMY
```

DISPLAY BOOKKEEPING INFO ABOUT THIS FILE.

WHAT ARE ITS ATTRIBUTES/CHARACTERISTICS? ______________________________

____

****

OMIT UNLESS INSTRUCTOR APPROVES

****

DO A DISK COMMAND AND RECORD THE NUMBER OF FREE BLOCKS.

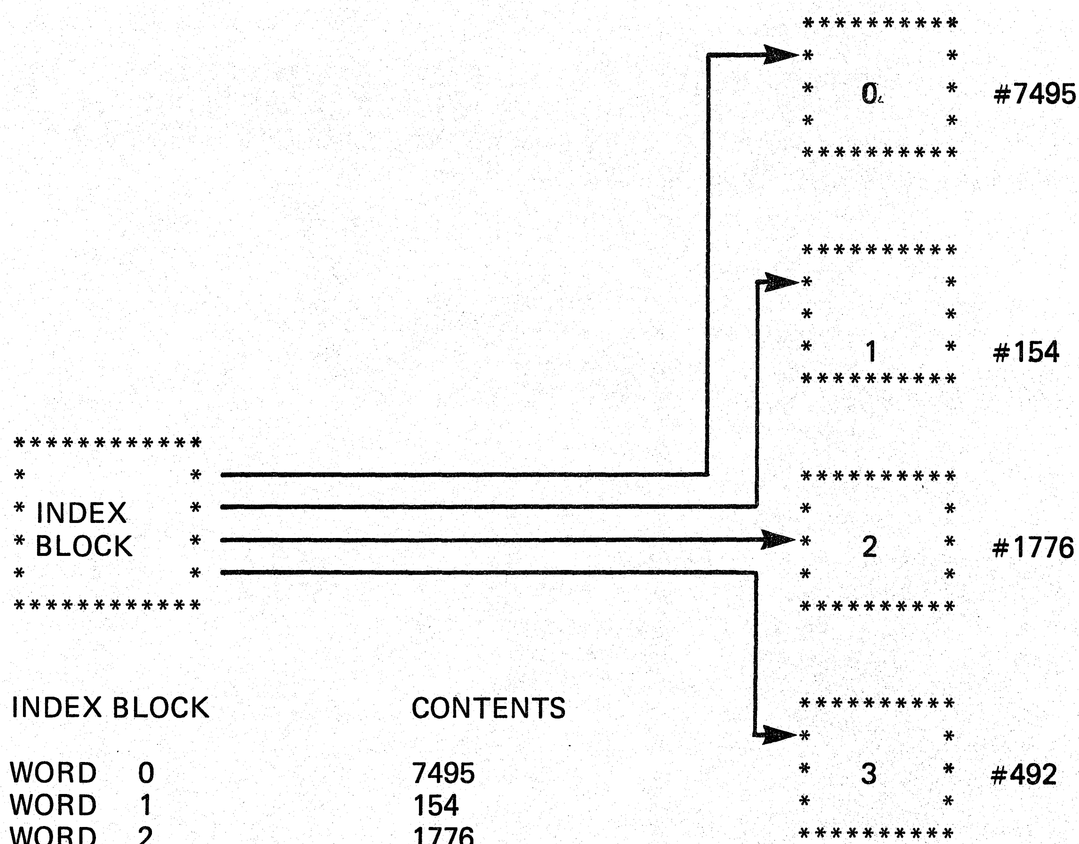
NOW TRY TO CREATE A CONTIGUOUS FILE WITH THE SIZE OF FREE BLOCKS YOU JUST RECORDED MINUS 10 (TO BE SAFE).

WHAT ERROR MESSAGE DID YOU GET?

THIS MEANS THAT WHENEVER YOU WANT TO CREATE A LARGE CONTIGUOUS FILE YOU SHOULD DO IT RIGHT AFTER YOU CREATE THE PACK AFTER A FULL INITIALIZATION. OTHERWISE YOU WON'T HAVE ENOUGH CONTIGUOUS BLOCKS TO DO IT LATER.

RANDOM FILES

THE LAST FILE STRUCTURE IS CALLED RANDOM. IT IS SOMEWHERE BETWEEN CONTIGUOUS FILES AND SEQUENTIAL FILES AND HAS SOME NICE FEATURES OF BOTH.



INDEX BLOCK		CONTENTS
WORD	0	7495
WORD	1	154
WORD	2	1776
WORD	3	492
WORD	4 & UP	−1

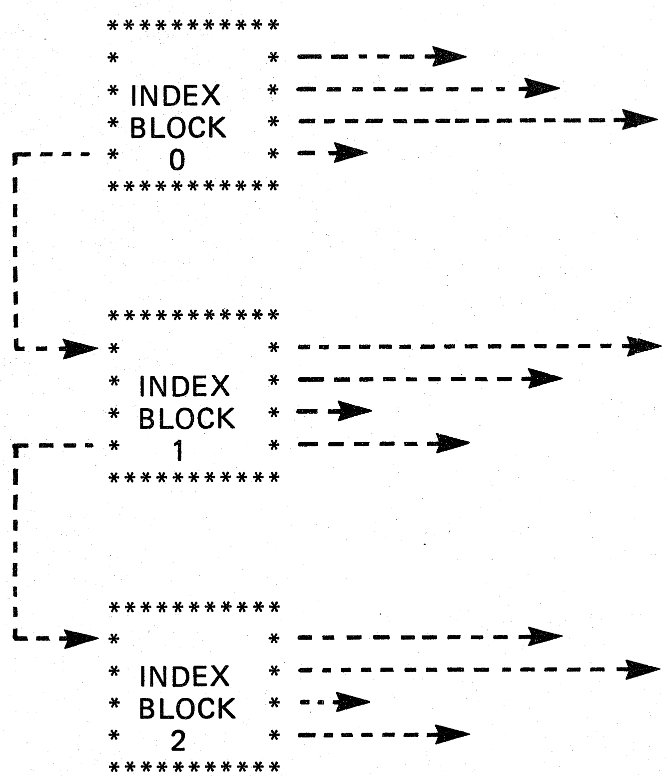
AS CAN BE SEEN FROM ABOVE, A RANDOM FILE HAS AN INDEX BLOCK (CALLED THE "RANDOM FILE INDEX BLOCK") WHICH POINTS TO THE VARIOUS BLOCKS WITHIN THE FILE. FOR THIS REASON, THE FILE CAN BE ACCESSED RANDOMLY AS WITH CONTIGUOUS FILES. UNLIKE CONTIGUOUS FILES, AND LIKE SEQUENTIAL FILES, A RANDOM FILE IS EXPANDABLE, AND CAN GROW TO THE USER'S ENVIRONMENT.

THE DIAGRAM ABOVE SHOWS THAT LIKE A CONTIGUOUS FILE, THE USER ONLY NEEDS TO KNOW THE RELATIVE BLOCK NUMBER, NOT THE PHYSICAL LOCATION ON THE DISK. THUS, THE USER ACCESSES BLOCK 2, AND NOT PHYSICAL BLOCK NUMBER 1776.

RDOS WILL TAKE THE LOGICAL BLOCK NUMBER, DISPLACE OFF OF THE INDEX BLOCK, AND LOOK AT THAT DISK BLOCK FOR THE INFORMATION.

THUS TO FIND BLOCK #2, LOOK IN INDEX BLOCK WORD 2 FOR THE PHYSICAL BLOCK NUMBER (=1776).

GIVEN THE FACT THAT A BLOCK IS ONLY 256 (DECIMAL) WORDS LONG, ONLY 255 ENTRIES WILL FIT IN AN INDEX BLOCK. WHEN AN INDEX BLOCK BECOMES FULL, THE SYSTEM AUTOMATICALLY CHAINS INDEX BLOCKS TOGETHER AS A SEQUENTIAL FILE AS SHOWN BELOW:



THIS SHOWS THAT 255 ENTRIES CAN FIT IN EACH INDEX BLOCK (THE LAST WORD POINTS TO THE NEXT INDEX BLOCK RATHER THAN A DATA BLOCK).

THE DISADVANTAGES OF RANDOM FILES ARE THAT THEY REQUIRE AT LEAST 1 EXTRA BLOCK OF DISK SPACE TO STORE THE INDEX BLOCK AND THAT AT LEAST 1 EXTRA DISK SEEK IS REQUIRED TO ACCESS A BLOCK OF DATA.

THE ADVANTAGES ARE THAT THEY ARE EXPANDABLE, AND THAT BLOCKS CAN BE ACCESSED IN A RANDOM ORDER.

RANDOM FILES ARE USEFUL IN KEEPING SMALL INVENTORY FILES THAT ARE DYNAMIC AND MUST BE ACCESSED QUICKLY. THE REASON I SAY SMALL IS BECAUSE AFTER A WHILE (SAY 700 BLOCKS) THE SEQUENTIALLY STRUCTURED INDEX BLOCKS BECOME A FACTOR IN DISK ACCESS TIME. <u>LARGER FILES THAT MUST BE ACCESSED QUICKLY SHOULD PROBABLY BE CONTIGUOUS, AND ENOUGH SPACE LEFT FOR EXPANSION.</u>

WHAT ARE THE OTHER USES FOR RANDOM FILES?____________________________

____

RANDOM FILE EXERCISE

JUST AS A SEQUENTIAL FILE CAN BE CREATED IN TWO WAYS, SO CAN A RANDOM FILE. FIRST, THOUGH, CLEAN UP BY DELETING TEST4.

```
R
DELETE/V/C TEST4
```

ALSO DO A DISK COMMAND. FREE__________________USED________

THE COMMAND "CRAND" IS USED TO CREATE A RANDOM FILE.

```
R
CRAND TEST4    (TEST4F)
```

DO THE ABOVE COMMAND AND THEN A DISK COMMAND   FREE____ USED____

HOW MANY DISK BLOCKS WERE USED? WHY?________________________________

DO A LIST COMMAND. THE ATTRIBUTE "D" INDICATES A RANDOM FILE.

ALL RDOS EXECUTABLE SAVED FILES ARE RANDOMLY ORGANIZED.

DO A

```
R
LIST/A/E –.SV
```

AND VERIFY THAT THIS IS SO. (INCIDENTALLY, THE STARTING BLOCK NUMBER IS THE BLOCK NUMBER OF THE RANDOM FILE INDEX BLOCK.) NOTE ESPECIALLY THAT ALL SAVED FILES HAVE AN "S" ATTRIBUTE TO INDICATE THAT THEY ARE SAVED FILES.

NOW DELETE TEST4, AND WE WILL USE THE OTHER METHOD OF CREATING A RANDOM FILE.

```
R
XFER/A $TTI TEST4/R
```

THE /R INDICATES THAT YOU WANT IT TO CREATE A RANDOM FILE INSTEAD OF THE DEFAULT SEQUENTIAL FILE. THE /R IS CALLED A "LOCAL SWITCH" AND MODIFIES A PARAMETER TO THE COMMAND RATHER THAN THE COMMAND ITSELF. (THE /A IS A GLOBAL SWITCH).

DO A LIST COMMAND TO VERIFY THAT IT CREATED A RANDOM FILE, AND A DISK COMMAND TO SEE HOW MANY BLOCKS WERE USED UP. WHY WERE THOSE BLOCKS USED?________________________________

RDOS SAVED FILES CAN ONLY BE RANDOMLY ORGANIZED. LET'S TEST THIS OUT WITH THE EDITOR. FIRST WE WILL MAKE IT SEQUENTIAL.

```
R
INIT UTIL
R
XFER UTIL:EDIT.SV MYEDIT.SV    (MYEDITF.SV in F/G)
```

NOTE NO GLOBAL /A IS USED BECAUSE THIS IS A BINARY TRANSFER, NOT AN ASCII TRANSFER. WE MUST ACCESS EDIT FROM THE UTILITY DIRECTORY BECAUSE THAT'S WHERE IT RESIDES.

SINCE WE ARE TRYING TO FAKE A SAVED FILE, WE NEED TO ADD THE ATTRIBUTE THAT SAYS THAT THIS IS A SAVED FILE. SO

```
R
CHATR MYEDIT.SV S
```

TRY TO EXECUTE MYEDIT BY TYPING IN

```
R
MYEDIT    (MYEDITF in F/G)
```

WHAT HAPPENED?____

NOW DELETE MYEDIT.SV AND CREATE A NEW ONE, RANDOMLY ORGANIZED.

```
R
XFER UTIL:EDIT.SV MYEDIT.SV/R
```

NOW CHANGE ITS ATTRIBUTES.

```
R
CHATR MYEDIT.SV S
```

EXECUTE IT IN THE SAME WAY.

```
R
MYEDIT
```

AN ASTERISK PROMPT MEANS YOU ARE IN THE EDITOR. TYPE AN H$$ TO GET OUT. THE DOLLAR SIGNS ARE REALLY ESC CHARACTERS. THIS CHARACTER IS ON A SPECIAL KEY LABELED "ESC" AND IS LOCATED SOMEWHERE ON YOUR KEYBOARD. IF THAT DOESN'T WORK, HIT A CONTROL C.

THAT'S IT FOR EDITING, FOR NOW, SO DELETE MYEDIT.SV

S200 RDOS User Laboratory Exercise

CLI LAB 3 – DIRECTORIES, LINKS

This section concerns disk files again and also links, secondary partitions, and subdirectories. Files, directories, and links to be created are referred to as "S200XXX." Please substitute your own names for them. You'll be using many of the commands and techniques introduced yesterday, so you may use the previous lab exercise as a reference. The new CLI commands used today are:

CHATR	GDIR
CHLAT	INIT
CPART	LINK
DIR	RELEASE
CDIR	UNLINK

CHANGING FILE ATTRIBUTES

Attributes are features of a file which can be set and changed by the user. They are most commonly used for file protection.

Copy S200SHOW To S200XXX.1

Now make your file read-protected.

CHATR S200XXX.1 R

Display bookkeeping info about this file. What are its attributes/characteristics?

____

Display the file's contents with the type command. What happened?

____

Remove the restriction.

CHATR S200XXX.1 0 (zero)

Can you display the file now?____________________

Use CHATR to apply the "D" characteristic to S200XXX.1
What happened?____________________________

Remember that characteristics are distinctive features of files which are set by RDOS and cannot be changed by the user. D = RANDOM FILE.

Make your file permanent (P Attribute).

Can you display file bookkeeping about the file now?____________

____

To display bookkeeping about a permanent file, you must use the "/A" global switch in the list command.

Try to delete your file.

CREATING A LINK

Links are a means whereby one file can be referenced by one or more alias names.

Create a link to S200XXX.1

LINK S200XXX.1L S200XXX.1

TYPE S200XXX.1L

TYPE S200XXX.1

What was different between the two outputs?______________________________

____

Display bookkeeping info about your link.

A link is merely a UFD which points to another UFD.

PROTECTION THROUGH LINKS

Links can also be used to protect a file. An extra set of attributes can be placed on a file that will be used whenever that file is accessed by any link.

Apply the link access attribute read-protect to S200XXX.1

CHLAT S200XXX.1 R

Note that the real (resolution) UFD determines the attributes added through links -- not the alias (link) UFD!!

How does list report the new attribute?___________________________________

Now, TYPE S200XXX.1

TYPE S200XXX.1L

What happened?____

____

REMOVING A LINK

DELETE   S200XXX.1L

What happened?____

Remove the P attribute on S200XXX.1 and try the delete again. Then do a LIST on both files to see what's gone.

Accessing a file through a link takes you to that file (the resolution file) and then performs the CLI operation: DELETE, TYPE, XFER, etc. List is the exception.

Notice that when accessing a file through a link, two sets of attributes apply: Link access and file. But when accessing a file by its real name only the file attributes apply.

To remove a link use the unlink command.

UNLINK   S200XXX.1L

Try to list S200XXX.1L

SUBDIVIDING DISKS

The total of all the space on a disk is a primary partition and has the same name as the disk unit (DP0, DP3F, DK1, for example). Parts of the disk may be sectioned off as secondary partitions or subdirectories. Primary partitions, secondary partitions, and subdirectories are all directories containing access information (UFD's) of files.

Create A Secondary Partition.

Secondary partitions are a fixed, contiguous set of blocks taken from a primary partition.

CPART   YOUR–PARTITION   (Use an original filename here)

Note that size is a required argument. Once created, a secondary partition can't be expanded.

CPART   YOUR–PARTITION 96

List bookkeeping about YOUR–PARTITION.

LIST/E   YOUR–PARTITION.DR

The DR extension is automatically appended on partitions and subdirectories.

What are the secondary partition's attributes/characteristics? ______________________

____

What is its size? ____

Is this the size you created? ___________________________________

(1 block = 512 BYTES)

Create a Subdirectory

Subdirectories are variable in size and randomly organized. They may be carved out of either primary or secondary partitions.

CDIR YOUR–SUBDIRECTORY (Use an original filename here)

No size needed here. List the bookkeeping on this subdirectory. What are its attributes/characteristics?

Changing Directories

Directories are used to isolate groups of files (i.e. all the work done for one client). Whenever you access a file, RDOS searches the current "default directory" for the UFD of that file.

Find the default directory --

GDIR

The list command reports only on files in the current directory. Is S200SHOW in this primary partition?

____

Make your subdirectory the current default --

DIR YOUR–SUBDIRECTORY

Verify the change by finding the current default directory.

File access in directories

Now try to display S200SHOW.

What happened? ____

LIST/A –.– These are the only files in your subdirectory.

LINKS AND DIRECTORIES

Links provide not only additional protection to files, but also an easy way to access files across directory boundaries.

Create a link in your subdirectory to DPO's copy of S200SHOW.

LINK S200XXX.2L DP0:S200SHOW

Now,

TYPE S200XXX.2L

A link can be used to span directories for many commands. Create S200XXX.2 by XFERing from S200XXX.2L. Did you get S200SHOW's contents?

____

Links can even be used to create their own resolution files!

LINK S200XXX.3L S200XXX.3

Check the bookkeeping to see if the link was created.

Well, was it?____

Try to type the link to display it.

What happened?____

Create a text file by XFERING from $TTI to S200XXX.3L Now, "TYPE" the link. Make DP0 the default and verify that S200XXX.3 has been created.

Ain't links wonderful?__________________________________

INITIALIZING DIRECTORIES

Let's try that last trick with your secondary partition (The contiguous type of directory). Link to it from the primary --

LINK S200XXX.4L YOUR–PARTITION: S200XXX.4

List it. OK so far?____

Now, XFER from S200SHOW to S200XXX.4L.
What happened? ____

____

RDOS can remember the names of only a limited number of directories. To let RDOS access files in a directory, it must be introduced to RDOS thru initialization. DP0 was INIT'ed when RDOS started up. Your subdirectory was INIT'ed as a part of "DIR". You haven't INIT'ed YOUR–PARTITION so RDOS doesn't recognize its name.

INIT   YOUR–PARTITION

Now,

XFER S200SHOW   S200XXX.4L

Display the contents of the link.

RELEASING DIRECTORIES

Once a directory is "INIT"ed it stays that way till "RELEASE"ed. Only one directory at a time is the default, but many can be INIT'ed at once. The exact number depends on the RDOS you're using. To allow room for new INIT'ed directories, release those you're done with.

RELEASE YOUR–PARTITION

Now,

TYPE   S200XXX.4L
(That link resolves to a file in the partition you just released)

You should get the same error message that started this section of INIT'ing directories.

DELETING DIRECTORIES

When you delete a directory, you also delete all the files that live in that directory.

How many free blocks are there? ____________________

DELETE/V YOUR–PARTITION.DR

Try to get rid of YOUR–SUBDIRECTORY.DR.

Your subdirectory is still INIT'ed so RDOS figures you're still using it; tell RDOS to let go of the directory with the RELEASE command. Now delete it (Remember the .DR extension).

How many free blocks now?____

S200 RDOS User Laboratory Exercise

DISK EDITOR LAB

Note

This exercise is optional because of its general applicability and degree of difficulty. So continue from this point providing that you have time and with your instructor's approval.

You'll be using the disk editor to trace thru directories and find a file. Then you'll use DSKED to recover a deleted file. Be careful when using DSKED – – it is a very powerful tool.

Create a secondary partition.
Create a sub-directory within that partition.
Create a file in that sub-directory.

Now invoke DSKED and trace through your file. Use the diagram on page D-39 as a reference.

Boot up DSKED. by - - -
BOOT DP0
Answer to "FILENAME?" -- DSKED
(DSKED runs instead of RDOS, not under RDOS)
Answer to "Disk Type?" --
4234 for top loader
4047 for front loader

Answer to "Disk Unit?" – DP0

(FS) Determine the frame size for this disk:

3:6/

(SPHV) Determine the hash value for your secondary partition. The hash value is the word of SYS.DR's index pointing to the block that will contain a UFD:

FS;your partition.DR=

(SPUFD) Now, what is the address of the block containing the secondary partition's UFD?

6:SPHV/

Find the UFD for your partition in this block. That is, search SPUFD:1, SPUFD:2 looking for the UFD describing your-partition.DR. SPUFD:0 will tell you how many UFD's are currently held in the block.

When you find the UFD, determine the starting address of your secondary partition (SPADD). It is the 12th octal offset within the UFD which points to the SYS.DR of the secondary partition.

(SDHV) Now determine the hash value for your sub-directory:
FS;your-sub-directory.DR=

(SDUFD) What is the address of the sub-directory's UFD block?

SPADD:SDHV/________________________________

Find the UFD for your sub-directory in this block; search from SDUFD:1

(SDADD) When you find the UFD, determine the starting block address of your sub-directory.

Determine the hash value for your file.

(FHV) FS;your file name =

What is the address of the block that contains the UFD for your file?

(FUFD)

SDADD:FHV/

Find the UFD for your file in this block.

(FADD) When you find the UFD, determine the starting block address of your file (FADD).

Go to that address and verify the contents of the file. (Display from FADD:0 several words. Is this the contents of your file?)

To go back to CLI hit the escape key (ESC), then Z and re-boot the system.

RECOVER A DELETED FILE

Once your back in CLI, create a file in your secondary partition. Now delete the file.

To recover this file we're going to invoke the disk editor and rebuild the file's UFD. We must also increment the first word of the block containing the file's UFD by 1. The first word keeps a count of the current number of UFD's in a block. Once we have recovered the file we should be able to print it. Remember the blocks comprising the recovered file are free according to MAP.DR so you should disable spooling (SPDIS $LPT) so the spooler does not grab these blocks.

Boot Up DSKED

Recover your File

Find your file's UFD. Remember your file is in a secondary partition, so you'll first have to trace to the secondary partition and then to the file.

SPHV → SPUFD → SPADD → FHV → FUFD

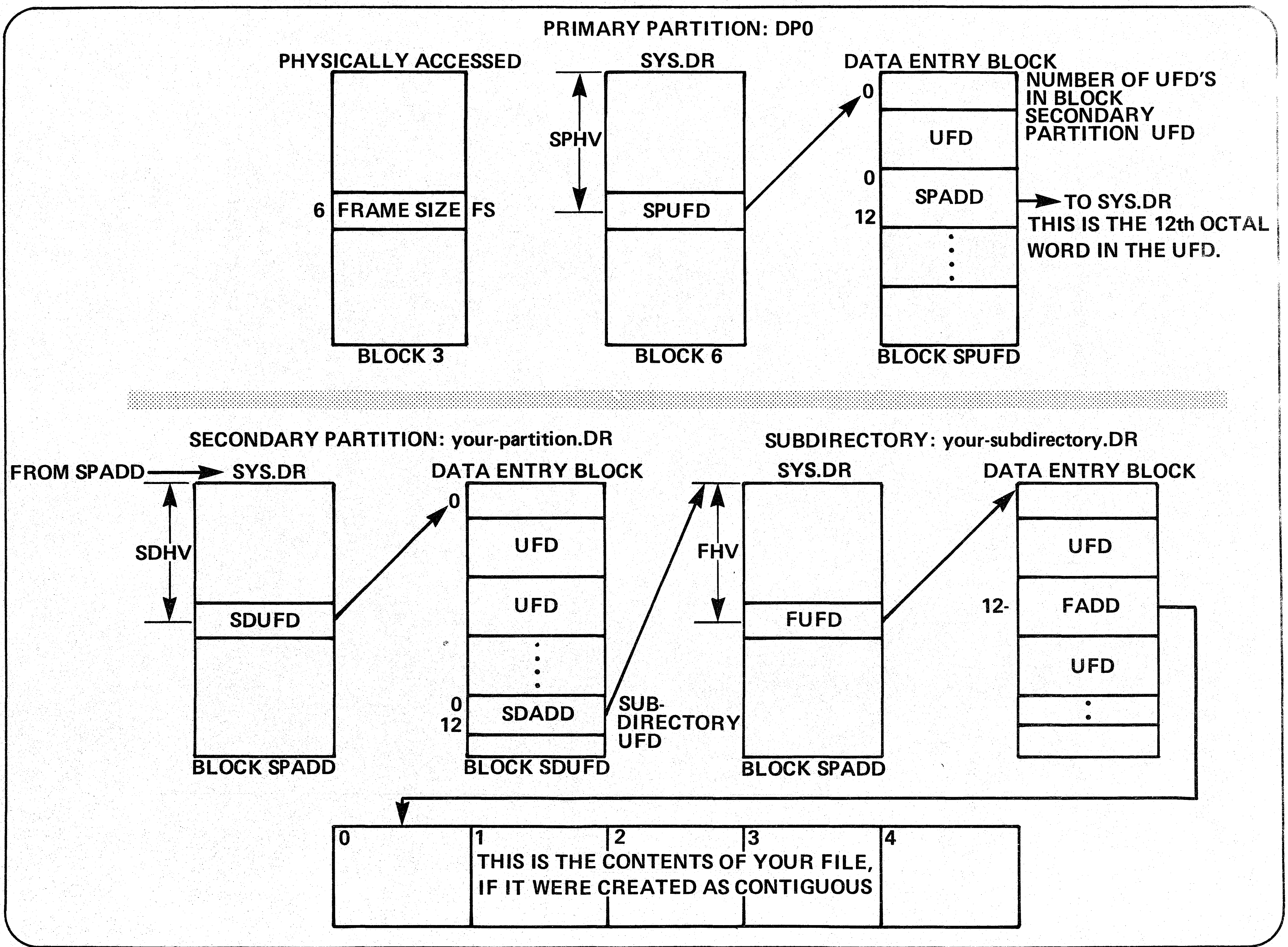
When you find your file's UFD, put the first two characters of the file name back into it. Then increment the first word of the block (containing the UFD) by 1.

Go back to CLI.

Disable Spooling

SPDIS $LPT

And Print Your File.


PRIMARY PARTITION: DP0
PHYSICALLY ACCESSED
6 FRAME SIZE FS
BLOCK 3
SYS.DR
SPHV
SPUFD
BLOCK 6
DATA ENTRY BLOCK
0
UFD
NUMBER OF UFD'S IN BLOCK
SECONDARY PARTITION UFD
0
SPADD
12
TO SYS.DR
THIS IS THE 12th OCTAL WORD IN THE UFD.
BLOCK SPUFD
SECONDARY PARTITION: your-partition.DR
FROM SPADD
SYS.DR
SDHV
SDUFD
BLOCK SPADD
DATA ENTRY BLOCK
0
UFD
UFD
0
12
SDADD
SUB-DIRECTORY UFD
BLOCK SDUFD
SUBDIRECTORY: your-subdirectory.DR
SYS.DR
FHV
FUFD
BLOCK SPADD
DATA ENTRY BLOCK
UFD
12-
FADD
UFD
0
1
2
3
4
THIS IS THE CONTENTS OF YOUR FILE,
IF IT WERE CREATED AS CONTIGUOUS

FPRINT/Z SYS.DR

THE FOLLOWING SHOW UFD'S WITHIN DATA ENTRY BLOCKS

```
   0 000003 041117 047524 051531 051400 000000 047514 040012 ..BOOTSYS...OL@.
  10 000000 000065 001000 000021 000001 000001 000000 000000 ...5...........
  20 000000 000000 000033 051104 047523 047400 000000 000000 ......RDOSO.....
  30 [illegible] 000004 000000 000123 000270 002055 006336 005515 LB....S...-.^.M
  40 007030 000000 000000 000000 000033 050101 051106 000000 ..........PARF..
  50 000000 000000 051522 000000 000000 000030 000015 003003 ....SR..........
  60 005365 005365 000003 000000 000000 000000 000033 000000 ................
  70 030063 050122 047502 031000 046123 000004 000000 000004 03PROB2.LS......
 100 000242 005312 006362 005425 000014 000000 000000 000000 .".J............
 110 000033 000000 030515 047000 000000 000000 051524 000004 ....1MN.....ST..
 120 000000 000020 000000 006104 006362 006362 007012 000000 .......D........
 130 000000 000000 000033 000000 000000 000000 000000 000000 ................
 140 000000 000000 000000 000000 000000 000000 000000 000000 ................
***
 370 000000 000000 000000 000000 000000 000000 000005 000000 ................
 400 000003 040514 046523 050104 000000 000000 051102 000004 ..ALMSPD....RB..
 410 000000 000000 000734 002214 005514 005514 005444 000000 .....\...L.L.$..
 420 000000 000000 000033 051531 051400 000000 000000 000000 ......SYS.......
 430 046123 000004 000000 000045 000370 002423 006336 006336 LS.....%......^.^
 440 004413 000000 000000 000000 000033 044115 050131 042000 ...........HMPYD.
 450 000000 000000 046102 000004 000000 000000 000234 002660 ....LB.........0
 460 005266 005266 006050 000000 000000 000000 000033 000000 .6.6.(..........
 470 046105 030400 000000 000000 000000 000000 000000 000000.LE1............
 500 000032 005265 006571 006571 006426 000000 000000 000000 ...5............
 510 000033 000000 030523 031000 000000 000000 051102 000004 ....152.....RB..
 520 000000 000001 000234 006063 006362 006362 007011 000000 .......3........
 530 000000 000000 000033 000000 031115 047000 000000 000000 ........2MN.....
 540 051102 000004 000000 000000 000000 006137 006362 006362 RB..........._...
 550 007020 000000 000000 000000 000033 000000 031123 030400 ..............251.
 560 000000 000000 051102 000004 000000 000001 000746 006127 ....RB.........W
 570 006362 006362 007041 000000 000000 000000 000033 000000 .....!..........
 600 031115 047000 000000 000000 047514 000010 000000 000003 2MN.....OL......
 610 001000 006317 006362 006362 007051 000000 000000 000000 ...O.....)......
 620 000033 000000 000000 000000 000000 000000 000000 000000 ................
 630 000000 000000 000000 000000 000000 000000 000000 000000 ................
***
 770 000000 000000 000000 000000 000000 000000 000010 000000 ................
1000 000002 051104 047523 041000 000000 000000 046102 000004 ..RDOSB.....LB..
1010 [illegible] 000014 000326 001717 006336 005514 004442 000000 .....V.O.^.L."..
1020 [illegible] 000000 000033 043111 047104 052506 042000 000000 ......FINDUFD...
1030 [illegible] 000000 000000 000000 000035 004740 006571 006571 ................
1040 [illegible] 000000 000000 000000 000033 [illegible] 031123 031000 .8..........252.
1050 [illegible] 000000 000000 000004 000000 [illegible] 000431 004715 ...............M
1060 [illegible] 006362 007036 000000 000000 [illegible] 000033 000000 ................
1070 [illegible] 031000 000000 000000 051522 [illegible] 000000 000000 252.....SR......
1100 [illegible] 006300 006362 006362 007020 [illegible] 000000 000000 ...@............
1110 [illegible] 000000 000000 000000 000000 [illegible] 000000 000000 ................
1120 [illegible] 000000 000000 000000 000000 [illegible] 000000 000000 ................
***
```

NUMBER OF UFD'S IN THE BLOCK

THE STARTING BLOCK OF THE FILE

THE UFD AFTER FILE DELETION

```
   0 000003 041117 047524 051531 051400 000000 047514 040012 . . BOOTSYS. . . OL@.
  10 000000 000065 001000 000021 000001 000001 000000 000000 . . . 5. . . . . . . . . .
  20 000000 000000 000033 051104 047523 047400 000000 000000 . . . . . . RDOSO. . . . .
  30 046102 000004 000000 000123 000370 002055 006336 005515 LB. . . . . S. . . -. ^. M
  40 007030 000000 000000 000000 000033 050101 051106 000000 . . . . . . . . . . PARF. .
  50 000000 000000 051522 000000 000000 000030 000015 003003 . . . . SR. . . . . . . . . .
  60 005365 005365 000003 000000 000000 000000 000033 000000 . . . . . . . . . . . . . . .
  70 030063 050122 047502 031000 046123 000004 000000 000004 03PROB2. LS. . . . . .
 100 000242 005312 006362 005425 000014 000000 000000 000000 . ". J. . . . . . . . . . .
 110 000033 000000 030515 047000 000000 000000 051524 000004 . . . . 1MN. . . . ST. .
 120 000000 000020 000000 006104 006362 006362 007012 000000 . . . . . . . D. . . . . . .
 130 000000 000000 000033 000000 000000 000000 000000 000000 . . . . . . . . . . . . . . .
 140 000000 000000 000000 000000 000000 000000 000000 000000 . . . . . . . . . . . . . . .
***
 370 000000 000000 000000 000000 000000 000000 000005 000000 . . . . . . . . . . . . . . .
 400 000003 040514 046523 050104 000000 000000 051102 000004 . . ALMSPD. . . . RB. .
 410 000000 000000 000734 002214 005514 005514 005444 000000 . . . . . \. . . L. L. $. .
 420 000000 000000 000033 051531 051400 000000 000000 000000 . . . . . . SYS. . . . . .
 430 046123 000004 000000 000045 000370 002423 006336 006336 LS. . . . . %. . . . ^. ^
 440 004413 000000 000000 000000 000033 044115 050131 042000 . . . . . . . . . . HMPYD.
 450 000000 000000 046102 000004 000000 000000 000234 002660 . . . . LB. . . . . . . . . 0
 460 005266 005266 006050 000000 000000 000000 000033 000000 . 6. 6. (. . . . . . . . . .
 470 046105 030400 000000 000000 000000 000000 000000 000000 LE1. . . . . . . . . . . .
 500 000032 005265 006571 006571 006426 000000 000000 000000. . . . 5. . . . . . . . . .
 510 000033 000000 030523 031000 000000 000000 051102 000004 . . . . 1S2. . . . . RB. .
 520 000000 000001 000234 006063 006362 006362 007011 000000 . . . . . . . 3. . . . . . .
 530 000000 000000 000033 000000 031115 047000 000000 000000 . . . . . . . . 2MN. . . . .
 540 051102 000004 000000 000000 000000 006137 006362 006362 RB. . . . . . . . . . _. . .
 550 007020 000000 000000 000000 000033 000000 031123 030400 . . . . . . . . . . . . 2S1.
 560 000000 000000 051102 000004 000000 000001 000746 006127 . . . . RB. . . . . . . . . W
 570 006362 006362 007041 000000 000000 000000 000033 000000 . . . . . !. . . . . . . . .
 600 031115 047000 000000 000000 047514 000010 000000 000003 2MN. . . . . OL. . . . .
 610 001000 006317 006362 006362 007051 000000 000000 000000 . . . O. . . . . ). . . . . .
 620 000033 000000 000000 000000 000000 000000 000000 000000 . . . . . . . . . . . . . . .
 630 000000 000000 000000 000000 000000 000000 000000 000000 . . . . . . . . . . . . . . .
***
 770 000000 000000 000000 000000 000000 000000 000010 000000 . . . . . . . . . . . . . . .
1000 000001 051104 047523 041000 000000 000000 046102 000004 . . RDOSB. . . . . LB. .
1010 000000 000014 000326 001717 006336 005514 004442 000000 . . . . . V. O. ^. L. ". .
1020 [illegible] 000000 000033 000000 047104 052506 042000 000000 . . . . . . . . NDUFD. . .
1030 [illegible] 000000 000000 [illegible] 000035 004740 006571 006571 . . . . . . . . . . . . . . .
1040 [illegible] 000000 000000 [illegible] 000033 000000 031123 031000 . 8. . . . . . . . . . 2S2.
1050 [illegible] 000000 000000 [illegible] 000000 000002 000431 004715 . . . . . . . . . . . . . . . M
1060 [illegible] 006362 007036 000000 000000 000000 000033 000000 . . . . . . . . . . . . . . .
1070 [illegible] 031000 000000 000000 [illegible] 000004 000000 000000 2S2. . . . . SR. . . . .
1100 [illegible] 006300 006362 006362 [illegible] 000000 000000 000000 . . . @. . . . . . . . . . .
1110 [illegible] 000000 000000 000000 [illegible] 000000 000000 000000 . . . . . . . . . . . . . . .
1120 [illegible] 000000 000000 000000 [illegible] 000000 000000 000000 . . . . . . . . . . . . . . .
***
```

NUMBER OF UFD'S COUNT DECREMENTED

FIRST TWO CHARACTERS OF NAME ARE NULLED

S200 RDOS User Laboratory Exercise

TEXT EDITOR LAB

Introduction:

In order to easily enter and modify ASCII text, Data General has created several text editors. Each editor is similar in command structure for ease in learning; the differences occur as extensions providing additional capabilities. The most fundamental editor is EDIT.SV; once learned, it provides a sturdy foundation to grasp the extended editors. The first phase of this lab is to correct the text in a file within the UTIL directory called GETTYSBURG.

The second phase of this lab will be program development of a "canned" FORTRAN program. You will merely enter in the fortran statements, compile the program, load the program and execute it.

The multi-editor, MEDIT.SV will be used to allow editing over a multi-terminal line.

So, let's get started:

YOUR INSTRUCTOR HAS ALREADY BROUGHT UP THE MULTI-TERMINAL EDITOR. YOU SHOULD ESTABLISH A CHANNEL TO YOUR COPY OF THE GETTYSBURG ADDRESS. TO DO THIS, FIND THE TAG ON YOUR TERMINAL THAT SAYS "LINE #n". THE n IS THE HARDWARE LINE FROM THE COMPUTER. YOU WANT TO EDIT A FILE CALLED "GETTYn" WHERE "n" IS THE LINE NUMBER.

USE UYGETTYn$$ TO OPEN AND YANK IN GETTY (BAD COPY).

CORRECT THE ERRORS.

CLOSE WITH US$$.

DON'T USE THE "G" COMMANDS FOR THIS EXERCISE (GR,GW,GC).

EDIT COMMANDS

. FILE ASSOCIATION COMMANDS

– GRfilename$: GET A DISK FILENAME OR DEVICE FOR READING INPUT

– GWfilename$: GET A DISK FILENAME OR DEVICE FOR OUTPUT

– GOfilename$: CLOSE CURRENT OUTPUT FILE GET ANOTHER FILENAME

– GC$: GET FOR CLOSING THE CURRENT INPUT & OUTPUT FILENAMES

. INPUT / OUTPUT COMMANDS

– Y$: YANK THE NEXT PAGE INTO THE CHARACTER BUFFER: A NUMBER MAY PREFACE THIS COMMAND AND THAT NUMBER OF LINES WILL BE PUT INTO THE BUFFER. PREVIOUSLY THE BUFFER IS CLEARED AND IS A POTENTIAL PROBLEM AREA FOR NEW USERS FOR DATA LOSS, BE CAREFUL . . .

– A$: APPEND THE NEXT PAGE TO THE CURRENT PAGE IN THE BUFFER A NUMBER MAY PRECEDE THE COMMAND TO APPEND A NUMBER OF LINES TO THE CURRENT PAGE.

– P$: PUT THE CURRENT PAGE TO THE OUTPUT FILE. IF A NUMBER PREFACES THE COMMAND THAT NUMBER OF LINES FROM THE CHARACTER POINTER POSITION (CP) IS OUTPUT.

– Itext$: INSERT TEXT FROM THE CURRENT POSITION OF CP UNTIL ESCAPE. A COMMON ERROR HERE IS TO CONFUSE CARRIAGE RETURN FOR ESCAPE AND THEREBY INCORPORATE COMMANDS IN WITH THE TEXT. A VERY SEVERE MALADY IS TO INCORPORATE THE PUT COMMAND INTO TEXT AND THEN ACTUALLY YANK IN THE NEXT PAGE, LOSING THE PREVIOUS PAGE OF TEXT. WATCH OUT . . .

. DELETE

– K$: A NUMBER PRECEDES THE COMMAND WHICH WILL DELETE THAT NUMBER OF LINES FROM CURRENT CP POSITION.

– D$: A NUMBER PRECEDES THE COMMAND WHICH WILL DELETE THAT NUMBER OF CHARACTERS FROM THE CURRENT CP POSITION.

. CP POSITIONING

– B$: REPOSITIONS THE CP TO THE BEGINNING OF THE BUFFER.

– Z$: REPOSITIONS THE CP TO THE END OF THE BUFFER.

– J$: USUALLY PRECEDED WITH A NUMBER INDICATING THE ABSOLUTE LINE USED TO POSITION THE CP.

– L$: AGAIN, USED WITH A PRECEDING NUMBER WHICH INDICATES A RELATIVE NUMBER OF LINES TO MOVE THE CP OVER. THE L COMMAND DEFAULTS TO A PRECEDING 0 WHEN USED ALONE.

– M$: A NUMBER PRIOR TO THE M COMMAND MOVES THE CP THAT NUMBER OF CHARACTERS FROM THE CURRENT POSITION.

. SEARCHING

– Stext1$: A SEARCH IS CONDUCTED FROM THE CURRENT CP POSITION UNTIL TEXT1 IS LOCATED, THE CP IS POSITIONED JUST AFTER TEXT1.

. CHANGE COMMAND

– Ct1$t2$: A SEARCH IS CONDUCTED FROM THE CURRENT CP POSITION UNTIL T1 IS LOCATED, T1 IS REPLACED WITH T2 AND THE CP IS POSITIONED JUST AFTER THE T1 TEXT.

. DISPLAY

– T$: DISPLAYS THE ENTIRE BUFFER; IF A NUMBER PRECEDES THE T COMMAND THAT NUMBER OF LINES ARE DISPLAYED FROM THE CURRENT POSITION OF THE CP.

– U? : DISPLAY CURRENT FILE OPEN FOR INPUT AND OUTPUT.

– . : DISPLAY THE LINE NUMBER THAT CP IS LOCATED ON.

– : : DISPLAY TOTAL NUMBER OF LINES WITHIN THE TEXT BUFFER.

. MACRO IMPLEMENTATION

-- XMcommand$: THE COMMAND STRING FOLLOWING THE XM COMMAND SERVES TO DEFINE A MACRO COMMAND STRING FOR REPETITIVE EXECUTION. REDEFINITION WILL ALSO REWRITE THE MACRO BUFFER.

-- XD : DELETES THE CURRENT MACRO IMPLEMENTATION.

– X$: EXECUTES THE CURRENT MACRO COMMAND STRING; IF THE COMMAND IS PRECEDED BY A NUMBER, THE MACRO IS EXECUTED THAT NUMBER OF TIMES.

– X? : DISPLAYS THE CURRENT MACRO STRING.

COMMON PROBLEMS

– THE SEVEREST ERROR WILL OCCUR WHEN STUDENTS FORGET TO PUT A BUFFER OUT TO DISK, AND ACCIDENTLY YANK ANOTHER PAGE WHICH CLEARS THE BUFFER.

– THE STUDENT WILL FORGET THE INSERT COMMAND "T", SUCH THAT INSERTION WILL OCCUR ONLY WHEN THE EDITOR STUMBLES UPON AN "I" IN THE TEXT.

– IF THEY ARE USING THE 6012 TERMINAL, CERTAIN CONTROL CHARACTERS WILL CHANGE THE CHARACTER INTERPRETATION BY THE EDITOR. OUTPUT WILL LOOK LIKE " <S <M". THE REMEDY IS TO STRIKE THE SAME KEY WHICH CAUSED THE MISINTERPRETATION.

NEVER IGNORE ERRORS – JUST WHEN YOU THINK YOU'RE SECURE, EVERYTHING IS LOST . . .

S200 RDOS User Laboratory Exercise

PROGRAM DEVELOPMENT LAB

The next phase involves program development of a FORTRAN program, contained below. This details the phases of development that all higher level language programs must proceed through and the code representation at each phase. Several errors have been imbedded in the program for error diagnosis. The errors are syntactical in nature, except for one logical error and for the FORTRAN neophyte the program is written correctly in an appendix in the student handout. So enter the following version as the first step in program development – ASCII source file creation.

```
C          THIS IS A FORTRAN TEST FILE, THE PROGRAM DEMONSTRATES
C          THE CODE REPRESENTATION AT EACH PHASE OF PROGRAM DEVELOPMENT.
C          THE ALGORITHM ACCEPTS : LOWER LIMIT, UPPER LIMIT, AND INCRE-
C          MENT TO SUM A GROUP OF NUMBERS OVER. THE RESULTS ARE
           PRINTED OUT.

10         ACCEPT "LOWER LIMIT, UPPER LIMIT, INCREMENT ", LL, LH,
C
           SUM = 0.0
           DO 100 I = LL, LH INC

               SUM = SUM = FLOAT(I)
               TYPE "I = ",I,"          SUM = ",SUMPTUOUS

100        CONTINUE

           GO TO 100
           END
```

Note: This program is intended to have errors.

Now compile your program with the following command:

```
R
FORT   your-program-filename   $LPT/L
```

Did it compile?____________________

What came out on the line printer?____________________

Do you have any syntax errors?____________________

Where are they? Circle them on your line printer listing. Now go back to the edit stage and fix your program and recompile it. (Correct program listing is on page B-1.)

If you have a good compile, proceed, otherwise consult your student handout appendix B for proper syntax.

How many words are generated for your program?____________________________

What is the first octal value generated for your program?

_________________________, and the last?________________________________

Now load your program with the following Relocatable Loader Command:

```
R
RLDR/P your - program - file   FORT.LB  $LPT/L
```

What is the FORT.LB file used for?____

____

____

Did you get any errors (yes, no). If you did check with your instructor, you shouldn't have any errors during the load phase.

What came out at the line printer?______________________________

What does the list tell you?_________________________________

Save both the source compilation listing and the program load map, there are questions about these later.

Now execute your program:

```
R
your-program-file
```

Did it perform as expected?________________________

If not, you have a logical error, which you should because we put one in the source program; but only one. So eliminate it and edit it from your source.

OPTIONAL:

If you can read Fortran code you know that the program is an endless loop. Type Control C to terminate your program. This causes a core image which was executing to be copied into (F) BREAK.SV. Print this file on the line printer up to location 2000 octal:

```
BREAK
R
FPRINT/L BREAK.SV 2000/T
```

The FPRINT command shows you exactly the core resident code loaded during execution. The load map documents this; using the assembly source, the load map and the BREAK.SV file make the following comparisons:

Where does the load map say your program starts?____________________
What is the first octal value in your source program?____________________
Now, look at the start location of your program in the BREAK.SV listing, what octal value is located there?____________________

Go through the same procedure for the last location in your program.

Usually errors are traced back to the source program using load maps denoting absolute core locations and relative offsets from the beginning of module starts.

For example, what routine contains the Data General Copyright?

Feel free to play in the Fortran Realm from this point on; otherwise release RDOS, power down all equipment, and discard your line printer paper.

S200 RDOS User Laboratory Exercise

BACKUP LAB 1

In this lab you'll learn how to back up an RDOS system and install RDOS on an initialized disk. The new CLI commands are:

DUMP

LOAD

MOVE

FDUMP

FLOAD

So, after a little practice with these commands you'll create a backup tape macro, execute it, verify it, and ultimately destroy the disk information and restore the system from your backup mag tape. After the system installation you'll generate your own tailored RDOS system with tuning and obtain a tuning file report. So let's begin with a little practice.

Boot up the system.

Record this session in "LOG.CM"

BACKING UP THE DISK INFORMATION

There are two kinds of information stored on disk –

File data (seen in XFER Commands)

RDOS bookkeeping (seen in LIST Commands)

To protect this info from loss by a disk crash, you should copy it from the disk to some other storage place -- a spare disk, mag tape, or paper tape.

XFER isn't designed to do this kind of copy. It would involve too much typing for an entire disk (XFER doesn't accept – or * templates). But more important than operator laziness is that XFER only copies the contents of a file, not its bookkeeping.

The DUMP and LOAD commands, however, are intended for back up. "DUMP MT0:0" will squish all the contents and bookkeeping of all the files on the entire disk into a single mag tape file (first file on the first unit, in this example). The argument could also be a single file on another disk or $PTP. If you ever need to restore the disk, a command like "LOAD MT2:0" would be all you'd input. That one command reverses the dump process and rebuilds many disk files from a single dump file on tape (in this case, the tape has been remounted on the third unit).

User XFER to create two files, S200XXX.6 and S200XXX.7

Mount a scratch tape on mag tape unit 0. Be sure there is a write-enable ring in the reel. Tell RDOS you've done this by

INIT MT0

USING DUMP/LOAD TO BACKUP

Back up one file you've created by

DUMP/V MT0:0 S200XXX.6

The additional argument overrides DUMP's default action of copying an entire disk. And it only backs up the files you describe. DUMP also accepts switches and templates that make it easy to describe groups of files, like "all save files created since April 1, 1977."

Now delete S200XXX.6.

Now try

XFER MT0:0 S200XXX.8.

And then try typing S200XXX.8.

What happened? ______________________________

Is the printout the same as the data you had originally inserted into the file? ______________

____

Now try

LOAD S200XXX.8

And then try typing S200XXX.6.

What happened? ______________________________

Remember the DUMP command dumped the contents of S200XXX.6 and its bookkeeping into one tape file. When you XFER'ed it back, you transferred file data and bookkeeping into one file called S200XXX.8. In order to make the file meaningful, the LOAD command separated the file data and bookkeeping again, thus recreating S200XXX.6. So to restore files backed up with the DUMP command, it's easier to use the LOAD command directly. Delete both S200XXX.6 and S200XXX.8.

Then

LOAD/V MT0:0

And try typing S200XXX.6. What happened? ______________________________

USING XFER TO BACK UP

XFER S200XXX.7 MT0:0

Now delete S200XXX.7 and

LOAD/V MT0:0

What happened?____

Remember MT0:0 was created with an XFER command, so no file bookkeeping is contained in it; MT0:0 is not in "DUMP" format. The dump file format is detailed in an appendix of the CLI manual.

Try

XFER MT0:0 S200XXX.7

This should work. Display S200XXX.7 and verify that it is the same as the file you originally created. The file data was transferred, and new bookkeeping was generated for the file. XFER, then, can accomplish the same objective as DUMP/LOAD (i.e., backing up files), but DUMP/LOAD is more efficient and easier to use.

USING MOVE TO BACK UP

MOVE, like DUMP/LOAD transfers both file data and bookkeeping, but maintains each as separate entities. That is, it doesn't put both file data and bookkeeping into one file.

File data is transferred as a separate file; the bookkeeping is moved into an RDOS system directory (SYS.DR) structure. For this reason MOVE cannot be used to create tape back ups (tapes have no SYS.DR), but it is very useful for disk to disk backups or directory to directory backups.

Create a subdirectory called DIRXXX.

MOVE DIRXXX S200XXX.6 S200XXX.7

DIR into DIRXXX and verify that the files were moved. Notice that they're still in the original directory as well. To use MOVE as a disk to disk backup you'd merely substitute the disk name (DP0F, DP1, for example) for the directory name.

USING FDUMP TO BACKUP

The FDUMP & FLOAD programs (residing in UTIL) allow a backup mag tape to be created in a more condensed format and more quickly. The speed is a result of multi-tasking programming; the condensed information is a result of storing bookkeeping and file contents together in separate mag tape files. Let's create a backup tape of the entire system; this will use mag tape files; MT0:0, MT0:1, MT0:2. If multiple copies of the disk are desired the second must be written to MT0:3 given that the first three tape files are used. So let's perform the back-up . . . (you may have to link to the FDUMP.SV program in the UTIL directory).

```
R
FDUMP/L  MT0:0
```

The FLOAD command is the reciprocal program to restore the disk information from tape. Note the condensed line printer listing, each indented line of asterisks denotes an entered directory to access the files there.

Again, you should have noticed a faster operation with the mag tape unit.

S200 RDOS User Laboratory Exercise

SYSGEN LAB

You will need the HOW TO LOAD AND GENERATE YOUR RDOS SYSTEM (93-188xx).

In this lab you will build a tailored operating system for the purpose of getting the machine to run more efficiently.

From DP0, initialize the UTIL directory and then:

(on B/G):DIR into the directory:GEN.DR

(on F/G):DIR into FGEN.DR

From the GEN directory, create a link to RLDR.SV and RLDR.OL in UTIL (these are used during the system generation).

Refer to chapter 6 in the manual "How to Load and Generate Your RDOS System" as an aid to answering the SYSGEN questions. Your instructor can fill in any particular hardware details you'll need for the system you are working on.

Begin the system generation by entering the SYSGEN command:

*SYSGEN YOUR-SYSTEM-NAME.</S SG/V LM/L>

where * is replaced by: nothing for NOVAs
N for NOVA 3s and 4s
B for ECLIPSEs

The /S will give the new operating system name to the .SV and .OL files.
The dialogue taking place will be recorded under YOUR-SYS-NAME.SG and the Load Map will have the .LM extension.

When you've answered all the questions, the system will load in (using RLDR) all the modules you have requested. This will take a few minutes.

PATCHING – UPDATING TO THE CURRENT REVISION

When you get your R prompt back, find the revision of the current operating system by:

REV %GSYS%

Now find the revision of your new system (REV YOUR-SYS-NAME).
Your new system was created at the most recent major revision level. You can bring it up to the current rev by patching your system.

Still in the GEN directory:

```
INIT DP0:PATCH.DR
LINK RDOS.PF PATCH:*RDOS.PF
```

where * is replaced with:
A   for ECLIPSE
N   for NOVA 3 or 4
M   for mapped NOVA
U   for unmapped NOVA

and then:

```
PATCH YOUR-SYS-NAME/S YOUR-SYS-NAME.LM/L  RDOS.PF/P
```

After you have installed the patches, verify that your new system is at the current rev level.

Now DIR into DP0 and create links to your operating system:

```
LINK YOUR-SYS-NAME.(OL,SV) GEN:YOUR-SYS-NAME.(OL,SV)   (FGEN ON F/G)
```

When both the users at the background and foreground have created these links, then bring down the foreground (CTRL F) and bring up one of the new operating systems by typing:

BOOT YOUR-SYS-NAME

TUNING

Now invoke tuning, exercise the system, and get a tuning file report:

```
TUON YOUR-SYS-NAME

R
.
.
.    Use some CLI commands to exercise the system
.
.
R
```

Turn off tuning and print the tuning report:

```
TUOFF YOUR-SYS-NAME
TPRINT/L/O YOUR-SYS-NAME
```

OPTIONAL:

Analyze the tuning report and, if called for, generate a new operating system with more appropriate requests for stacks, cells, and buffers either by:

1. Fully accepting the results of the tuning report:

   *SYSGEN NEWSYS.</S LM/L SG/V> YOUR-SYS-NAME.<SG/A,TU/T>

   or

2. Going through all the questions again but having the tuning report recommendations displayed for the questions on stacks, cells, and buffers:

   *SYSGEN NEWSYS.< /S,LM/L,SG/V> YOUR-SYS-NAME.TU/T

In either case, patch and boot this new system.

Finally get any printouts you'll want (the .SG file is useful) and boot up the original operating system, unlink your systems in DP0, and delete all the files that you created in the GEN and FGEN directories.

S200 RDOS User Laboratory Exercise

SPOOLING LAB 1

By now, you should feel fairly inundated with laboratory material and exercises. Today's lab is devoted to some finalization of previous material and some short studies on lesser important facilities talked about on Friday. The new material exercised in lab will involve a SPOOL'ing exercise and a Foreground/Background familiarization to allow the student an opportunity to work with RDOS's full capabilities.

At this point, take it upon yourself to wrap up any unfinished lab assignment. If you are up to date with lab, conclude your inspection of RDOS with the following two exercises.

Spooling affords CPU optimization by temporarily storing information on disk as the output device is initiated. Further information is requested when needed by the device via interrupt. This technique allows the CPU to work on independent of the transfer.

Although subtle, we can realize spooling stages from output at the master console . . . During a long print, the system will go away, initiate the device transfer and, when spool files are created on disk, an "R" prompt will be given. At this point CLI may be controlled, and any device may be used in parallel while the line printer maintains the output transfer. For example, create the following macro:

```
R
XFER/A %GCIN% DSK.MC
DISK
DSK.MC
```

Hit ↑Z to terminate data entry mode. The above macro allows disk inquiries to be made on RDOS. If this is employed during SPOOLing of output data to disk, we can be shown intermediate disk storage used for spool files.

So print the file PARU.SR within the UTIL directory and, as soon as you have an "R" prompt, submit the DSK.MC macro by typing its name. The repetitive execution of the disk command is like watching a movie of disk block usage. You should be able to watch spool files come from disk and print on the line printer.

S200 RDOS User Laboratory Exercise

BACKUP LAB 2 (OPTIONAL)

This lab involves:

- Dumping all files to tape
- Full initialization of the disk
- Installing the RDOS starter system
- Loading back files from tape

Now you have the necessary mechanics to create an effective backup macro. Remember, the XFER command transfers to tape an executable format of disk files. So either use the editor or transfer from the master console the following backup macro under a name with the .MC extension.

```
MESSAGE "BACKUP IN PROGRESS – DO NOT DISTURB"
/
/STANDALONE FACILITIES
XFER TBOOT.SV MT0:0
DUMP/A MT0:1 CLI.<SV, ER, OL> , BOOTSYS.SV,BOOT.SV
XFER BOOTSYS.SV MT0:2
DUMP/A MT0:3 BOOTSYS.OL
XFER DKINIT.SV MT0:4
XFER BOOT.SV MT0:5
/DUMP THE CURRENT OP SYS SV+OL FILES
DUMP MT0:6 OP-SYS.<SV, OL>
             replace with
             actual name
/
MESSAGE "BACKUP COMPLETE"
```

Ordinarily here you would Dump 2 copies of your entire disk:
 Dump MT0:(6,7)

Verify the standalone facilities by releasing DP0 and initiating execution for each standalone program directly from tape. For example, if TBOOT.SV is installed on the tape properly, you should see:

RELEASE %MDIR%
MASTER DEVICE RELEASED

.

.

.

set the number switches to 100022

depress STOP, RESET, PRGM LD

.

.

.

FROM MT0:

At this point TBOOT.SV has performed sufficiently to indicate that it is on MT0:0 and is bootstrap'able. So, check mag tape files: 0, 2, 4, and 5 as all the standalone files are on tape at these relative positions.

Now it's a go – no go situation. If you are unsure of any portion of the backup tape ask your instructor, later is not the time to ask questions . . .

Refer to the How to Load & Generate Your RDOS System for details around the system load; this information is contained in Chapter Three. The sections to execute in sequence are:

3 – 3: The Disk Initializer
3 – 6: Installing the Disk Bootstrap
3 – 6: Installing the RDOS Starter System

So install the RDOS Starter System with aid from the above sections. Modify the remaining section, Transferring the Remaining Files, because yours isn't a true starter tape.

When you bootstrap BOOTSYS.SV from MT0:2, it will afford you only two directories initialized at the same time. Therefore, you must boot a more appropriate operating system. So execute the following:

R
LOAD/A/V MT0:6

Then boot a system affording more directories to be initialized simultaneously.

R
BOOT NEWSYSTEM (name of your original operating system)
MASTER DEVICE RELEASED

.

.

.

S200 RDOS User Laboratory Exercise

SPOOLING LAB 2 (OPTIONAL)

Note: If optional Lab 1 was performed, see your instructor for the back-up tape before proceeding.

The purpose of this lab is to demonstrate the advantage of making a secondary partition the master directory.

The plan is to create a secondary partition containing a minimum of software, links to the operating system, and some spare blocks for spooling. Then bring up the system in that secondary partition, thereby making it the master directory. Once you get an R prompt from CLI, DIR to DP0 and continue operating there. Then, if the system crashes while spooling, the secondary partition can be deleted, allowing the disk blocks which were used for spooling to be recaptured.

1. Create a secondary partition with enough space for CLI files and some spare blocks.

   CPART SECPART 288

2. Set up the bookkeeping for it.

   INIT SECPART

3. Move copies of the CLI files into that partition.

   MOVE/A/V SECPART CLI. <SV, ER, OL>

4. DIR into that partition and create links to the .SV and .OL files for the system you will be operating under.

   DIR SECPART
   R
   LINK opsysname. (SV, OL)/2 (Replace opsysname with the name of your operating system)
   R

5. From this point, you may bring up the system in SECPART using the device specifier format. So, in response to FILENAME?, you may specify SECPART: opsysname. Or you may use the CLI command BOOT (that is, BOOT SECPART:opsysname).

   At this time, boot up the system in the secondary partition using one of the methods described above.

6. Next, execute the following CLI commands:

   GDIR Current default directory = ______________________
   R
   MDIR Master directory = ______________________

R
DISK ____________LEFT ________USED
R
DIR DP0
R
DISK ____________LEFT ________USED
R
INIT UTIL; PRINT UTIL:PARU.SR
R
DISK ____________LEFT ________USED
R
DIR SECPART
R
DISK ____________LEFT ________USED
R
SPKILL $LPT
R
DISK ____________LEFT ________USED

Notice that the disk blocks for the spool queue were taken from the master directory.

7. This time, repeat all the CLI commands in step 6 up to the SPKILL command. Instead of issuing the SPKILL command, toggle STOP and RESET on the front panel. This will cause the system to cease operations in a less-than-graceful manner. It will then be necessary to re-home the disk heads. This may be done by toggling the LOAD/RUN switch on the disk drive. When the Ready light comes on, boot the system in the normal manner and make DP0 the master directory.

8. Next, clear the file use counts in both DP0 and SECPART.

CLEAR/A/V/D;CLEAR/V CLI.<ER, OL>, opsysname.OL
R
DIR SECPART
R
CLEAR/A/V/D;CLEAR/V CLI.<ER, OL>

9. Now execute the following CLI commands:

DISK	____________ LEFT ________ USED
R	
DIR SECPART	
R	
DISK	____________ LEFT ________ USED
R	
DIR DPO	
R	
RELEASE SECPART	
R	
DELETE SECPART.DR	
R	
DISK	____________ LEFT ________ USED

You have re-captured the disk blocks which were lost when the system crashed.

Data General

Data General Corporation, 4400 Computer Drive, Westboro, MA 01580

(617) 366-8911