UNISYS

U Series MAPPER® Advanced Run Design

Student Guide

March 1990

Printed in U S America

AL 2801

UE 8498

Contents

Course Description

Audience

Experienced run designers who build and maintain MAPPER runs.

Prerequisites

SE 1564 - MAPPER Basic Run Design, and at least six months current run design experience.

Objectives

Upon successful completion of this course, the student should be able to

- Understand internal MAPPER run processing; especially I/O handling concepts and their effect on MAPPER runs and the system.
- Analyze various efficiency methods and apply them to the MAPPER system.
- Use advanced run design techniques to design and maintain more efficient MAPPER runs.

Description

This course provides the experienced MAPPER run designer with tools and techniques to design and maintain complex but efficient runs. Emphasis is placed on efficiency technqiues; lectures are enhanced by hands-on exercises to apply methods of analyzing and improving run efficiency.

Topics

- Internal MAPPER Overview
- Run Efficiency Analysis Tools
- Run Recovery
- Screen Design
- Advanced Work With Variables
- Unisys System V Interface
- Networking

Duration

5 days

Agenda

Day	Module	Topics
1		Introduction and Welcome
1	1	MAPPER Overview o System sign on o User sign on o PreRun o MAPPER database files o Drawer table o RPT table o Page table o Summary o Exercise
1	2	Basics of RCR Execution o RCR structure o Control bank o Memory allocation o Memory manager o Buffers, blocks, and I/Os o Output area o Run interpreter o Function processing o RCR characteristics o Report size calculation o @CAH statement o Summary o Exercise
1	3	Efficiency and Analysis o I/Os o LLPs o DLPs o Analysis tools o Accounting log o @LOG statement
2		o Interim vs. noninterim displays o LOGLA o RUNA o Summary o Exercise

Day	Module	Topics
2	4	Recovery Concerns and Techniques o System recovery o Update functions o Report size o Results o Duplex files o System restoration o Purges o Checkpoint reports o Deferred updates o @DFU statement o Summary o Exercise
3	5	Advanced Variable Concepts o Variable review o Standard names o @USE statement o Variable table o Variable stacks o STACK$ o @PSH and @CLV statements o @POP, @PEK, and @POK statements o @RMV and @XCH statements o BVT o VARIABLE run o Summary o Exercise
3	6	Efficient Run Structure o Modular Run structure o Label table o Label definition lines o @BLT statement o @CLT statement o @GTO statement o Computation GTO o GTO RPX o Subroutine review o @RSR, @CSR, and @ESR statements o @CALL and @RETURN statements o @RUN and @LNK statements o @XQT statement

Day	Module	Topics
		o Abort routines o @RAR statement o Error routines o @RER statement o Summary o Exercise
4	7	Screen Design Technqiues o INPPUT$ o INVAR$ o INVR1$ o INSTR$ o @LFC and @SFC statements o @OUM and @FMT statements o @CHD and @KEY statements o Forced transmit o Summary o Exercise
4	8	Basic Screen Control Techniques o @OUT statement with options o 4-to-1 o 5-to-1 o @SC statement o Cursor control commands o Screen editing commands o Field attribute commands o Emphasis commands o Special commands o Summary o Exercise
5	9	Indexing Schemes o Binary find o Sorted data reports o @BFN statement o Creating an index report o Using an index report o Number generation o Summary o Exercise

Day	Module	Topics
5	10	Interfacing with UNIX o UNIX overview o Types of UNIX files o UNIX file structure o UNIX filename syntax o MAPPER and UNIX o @FIL statement o Data control commands o @RET statement o @UNX statement o @XUN statement o Summary o Exercise

About This Course

Curriculum Information

This course, U Series MAPPER Advanced Run Design (AL 2801), is the fourth in a series of MAPPER training courses produced by the Unisys Education Center in Princeton. Other MAPPER courses in the MAPPER training path include:

- SE 1562 - MAPPER User Workshop for Programmers
- SE 1563 - MAPPER User Workshop for Business Users
- SE 1564 - MAPPER Basic Run Design

1

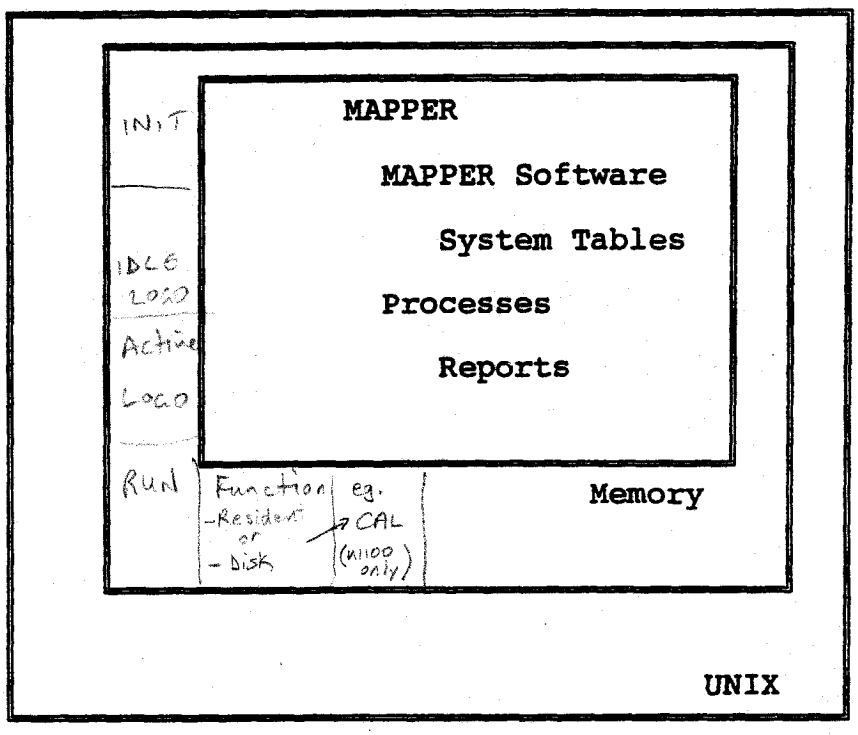
MAPPER Overview

Module Objectives

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

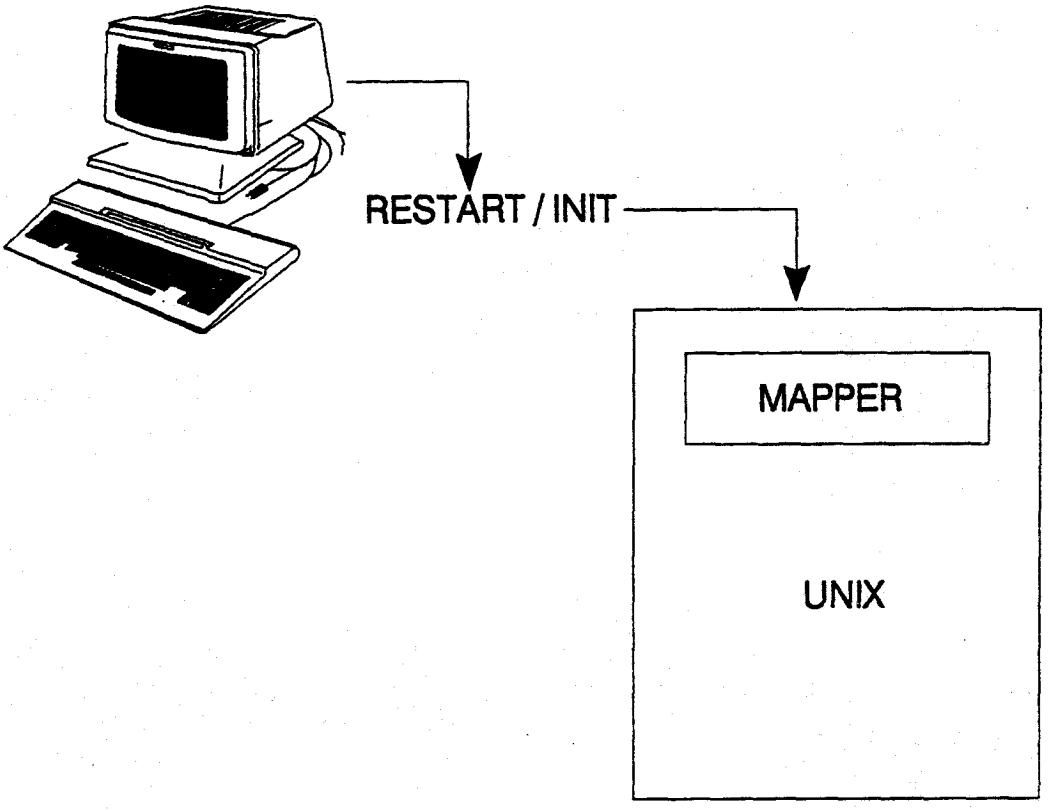
1. Identify the steps required on the MAPPER system for user sign-on.
2. Define the purpose and content of selected system tables.
3. List characteristics of the MAPPER database.

MAPPER Overview



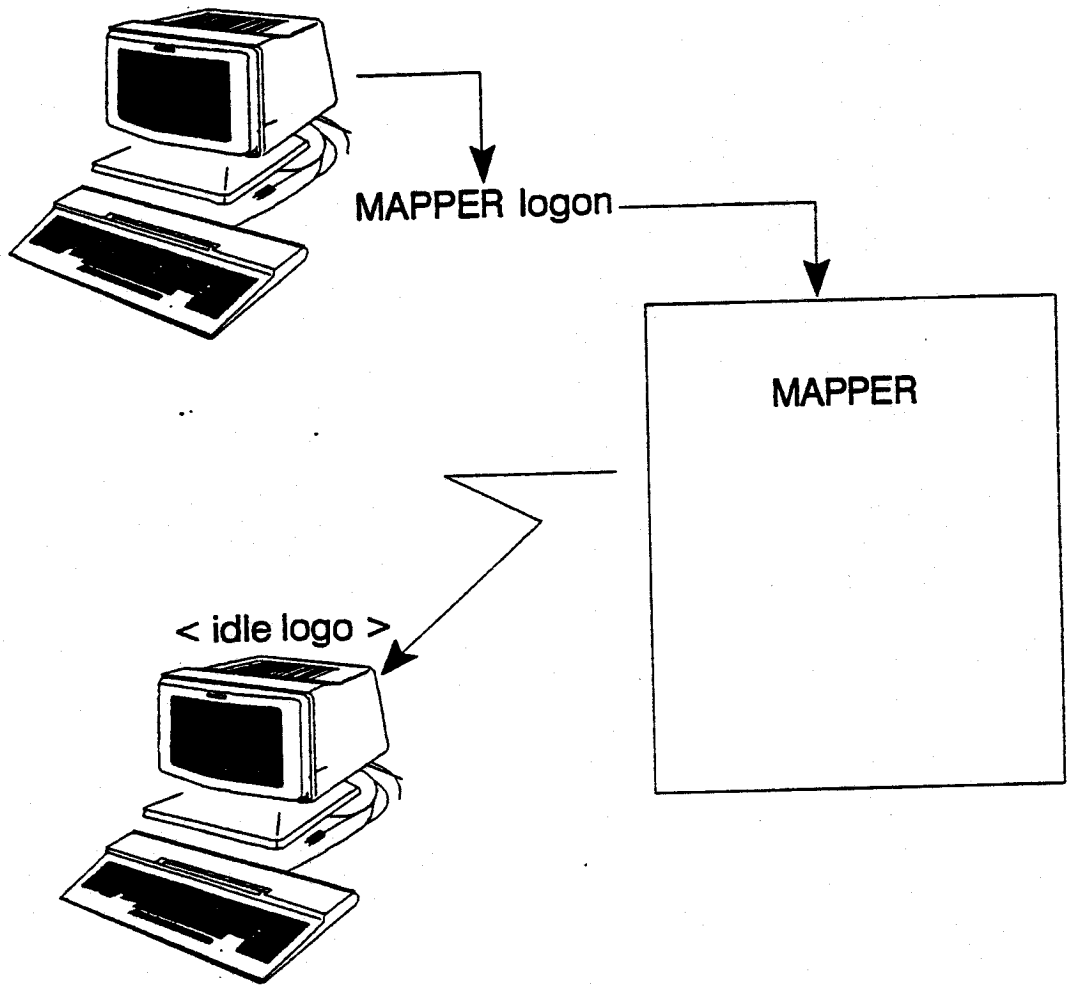
V1-1

Bringing MAPPER up on UNIX



V1-2

Executing MAPPER

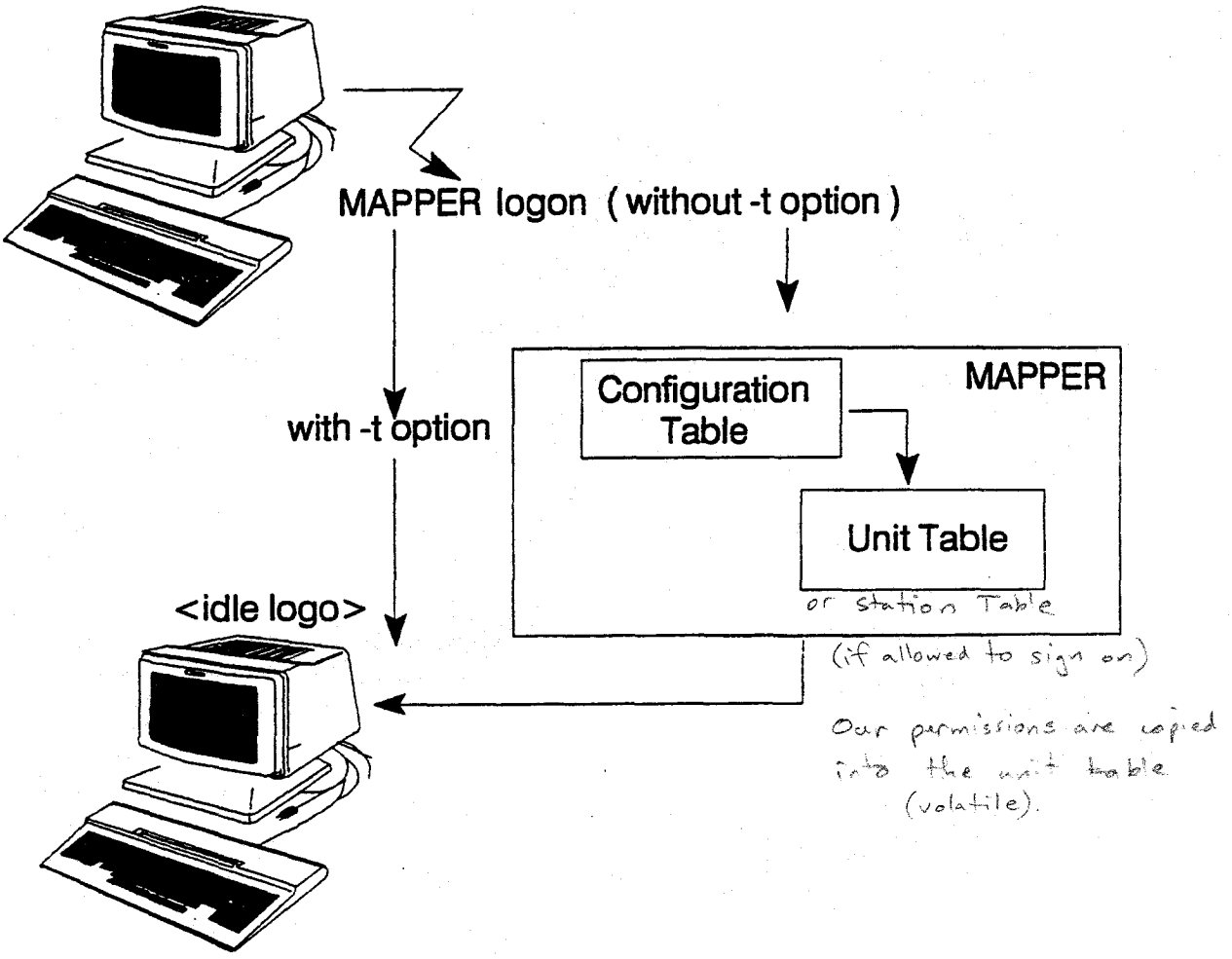


$$OPEN -----
^ (caret)

V1-3

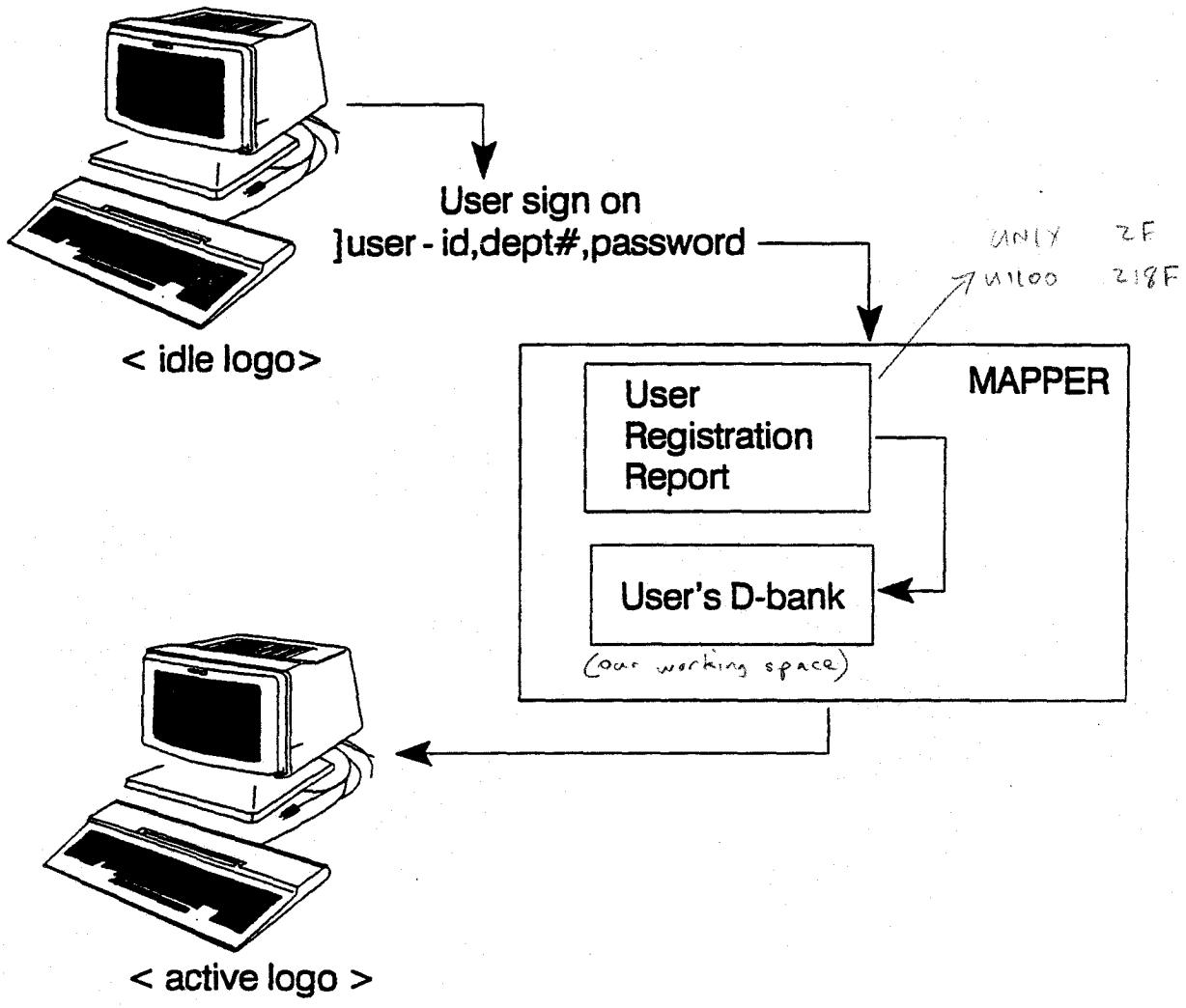
PC Designer Workbench
Script (Macro)
User-id

Executing MAPPER



V1-4

User Sign-on



V1-5

User Registration Report

```
.DATE   27 JUN 89  15:37:34  RID    7F   18 JUL 89  MAPCOORD
.USER REGISTRATION REPORT                                          F0032
.REPORT FOR DEPARTMENT: 'Dept.  name'
*           .     .    .RADRDDAAFSSSMMUDELCTAICDCCPELASSDPFLRSSOMBDRDVSBR ---->
*           . PASS .USER.UDPELRDDNRUOCUPEXOHORNAAAASXZUXQPRUNFETOSFSCWEYGEI---->
* USER-ID   . WORD . CAB.NRRPRWDTDHDRHDDLTCGTTDBTLUWIRXXQWTNGMNSKGNGRNRSRTL---->
*-----------.------.----.-------------------------------------------------->
 HUNNEMAN          2                                                    ---->
 JSTEWART          20                        X        X                 ---->
 BURGESS           20                                                   ---->
 CREED             20                        X        X                 ---->
 JDOE              0     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX---->
 QS                0                         X        X                 ---->
                    ..... END REPORT .....
```

X - means cannot execute function

Search Sequence

After the active logo appears, MAPPER must interpret what appears next on the control line:

- **Manual function a user may execute?**
  - checks your permissions
- **Department run?**
- **Global run?**
- **None of the above <Sorry, unable to process your request>** (TUF)

To register a run for more than one dept, register it separately for each department

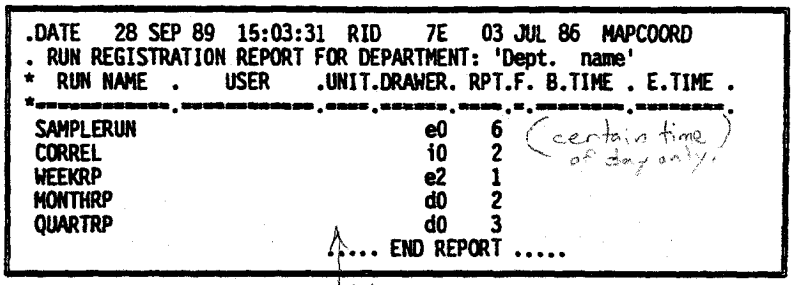
PreRun

- A MAPPER routine started if a manual function is not located in the user's D-bank.

- Determines if the entry is a valid run name.

- Checks department/master run registration report for applicable restrictions.

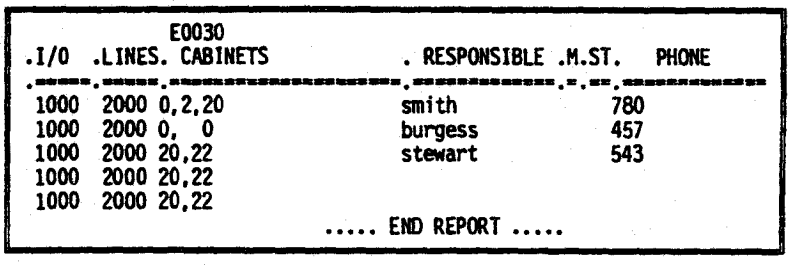
Run Registration Report



```
.DATE   28 SEP 89  15:03:31  RID    7E   03 JUL 86  MAPCOORD
. RUN REGISTRATION REPORT FOR DEPARTMENT: 'Dept.  name'
*  RUN NAME  .    USER    .UNIT.DRAWER. RPT.F. B.TIME . E.TIME .
*-----------.-----------.----.------.----.-.-------.-------.
 SAMPLERUN                          e0    6
 CORREL                             i0    2
 WEEKRP                             e2    1
 MONTHRP                            d0    2
 QUARTRP                            d0    3
                          ..... END REPORT .....
```

(certain time of day only)

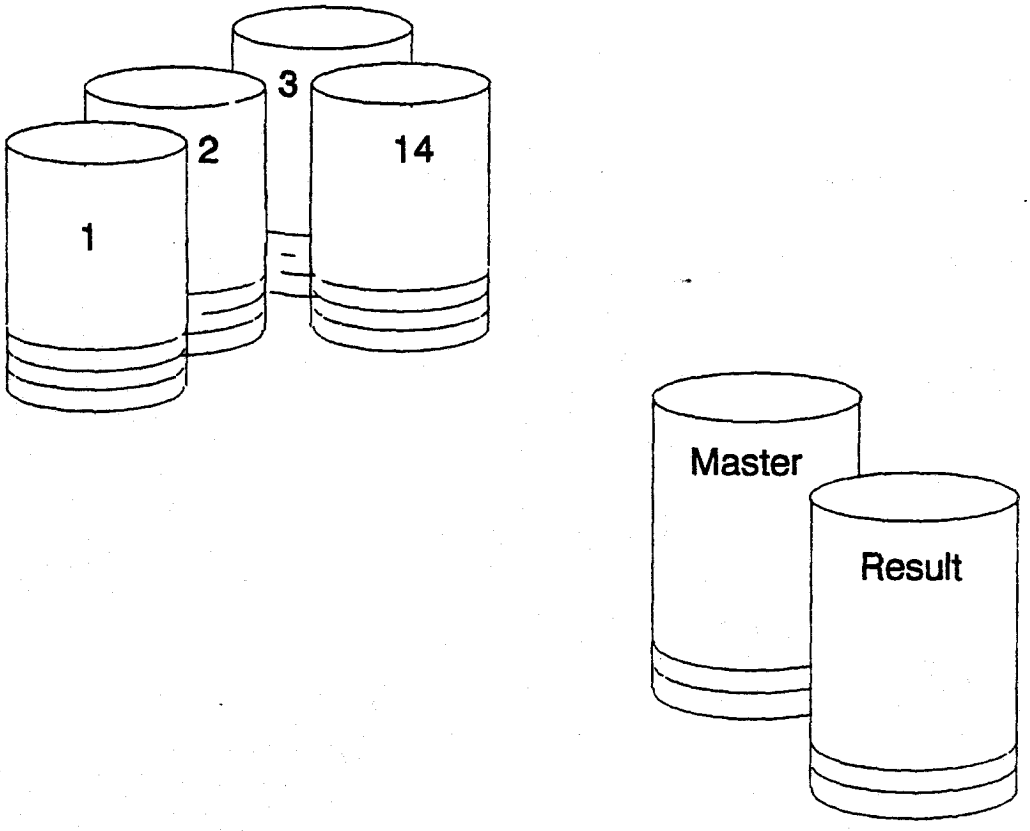
restriction to a particular terminal



```
                E0030
.I/O  .LINES. CABINETS            . RESPONSIBLE .M.ST.   PHONE
.-----.------.--------------------.-------------.-.--.-----------
 1000  2000 0,2,20                 smith              780
 1000  2000 0,  0                  burgess            457
 1000  2000 20,22                  stewart            543
 1000  2000 20,22
 1000  2000 20,22
                          ..... END REPORT .....
```

Register the run for more than one user by adding extra lines for each user.

MAPPER Database



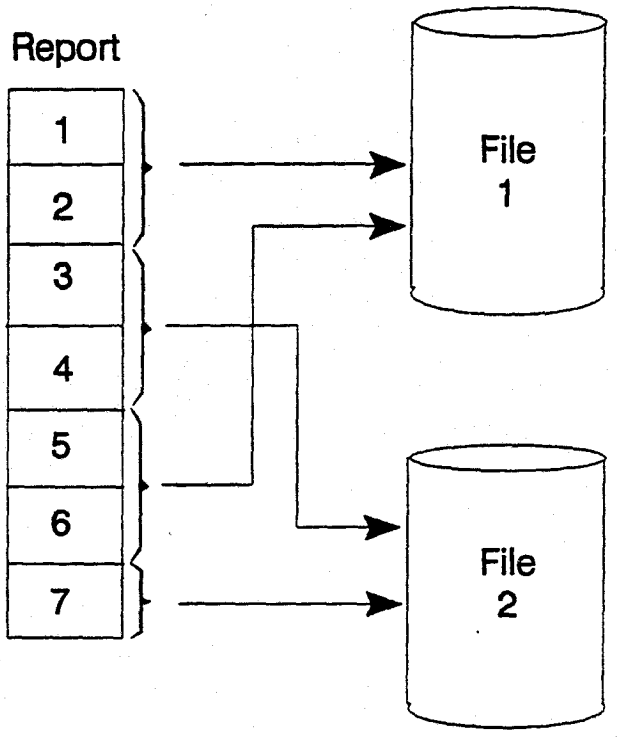
V1-8

MAPPER Database Files

- Maximum of 14.
- Standard UNIX or character special files.
- Maintained in "round robin" method.

MAPPER or UNIX can maintain I/O's.

MAPPER Database Files

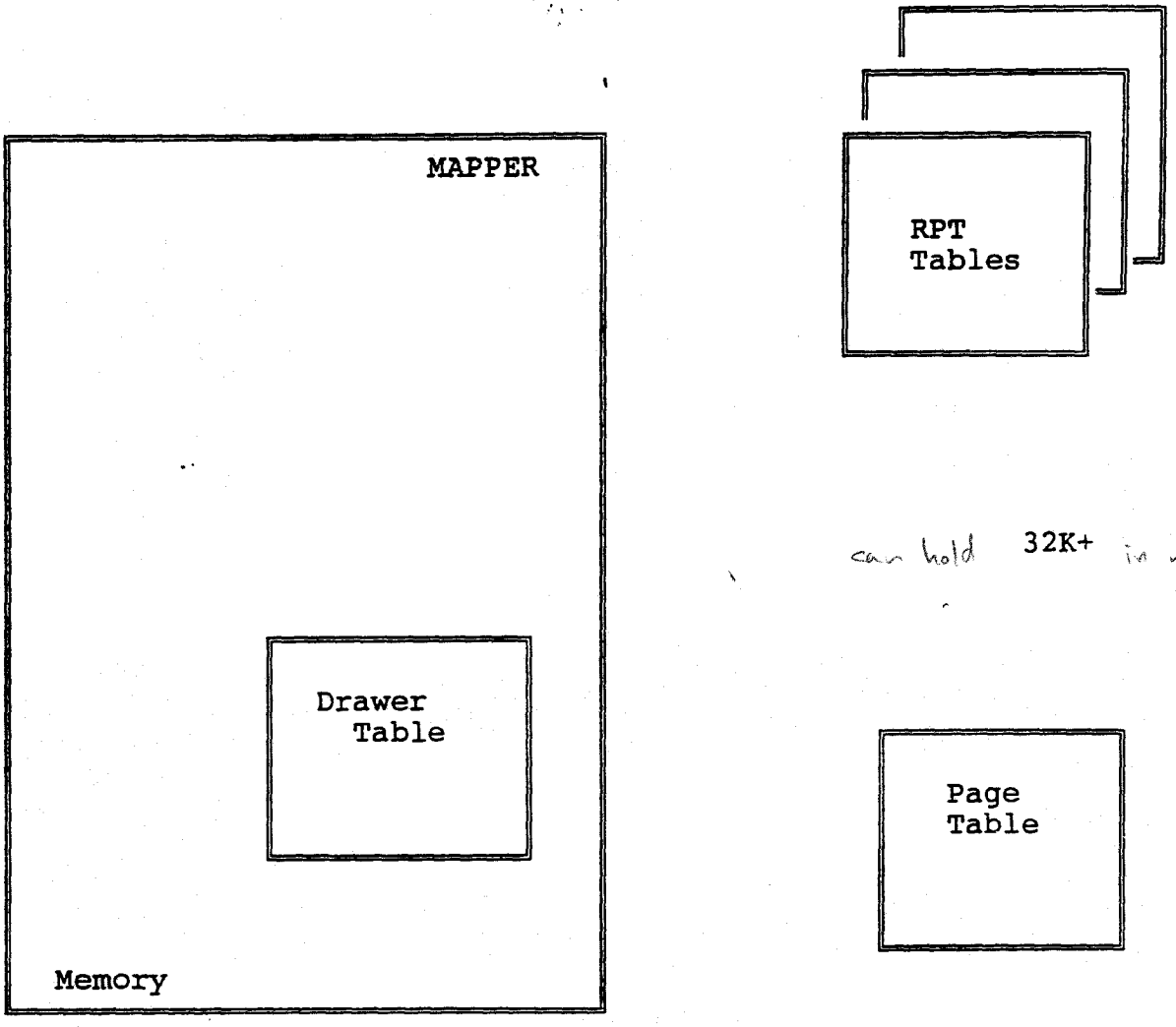


V1-9

Additional System Tables

- Drawer table
- RPT table
- Page table

Additional System Tables



V1-10

Drawer Table

- Created at system initialization
- One per MAPPER site; always resident in memory
- Points to the RPT tables

Drawer Table

****

* DRAWER TABLE *

****

R	P	DRWR	RVF	L	CHR	MXRD	RZ	RTF	DAD	RTCAD	RTU	PZ	PTF	DAD	PTCAD	PTU
1	1	1	0	0	88	1021	5	1	2159	1ae000	4000	1	2	4	1ad000	4000
1	1	2	0	0	88	1000	5	1	9388	0	0	1	1	245	0	0
1	1	4	0	0	88	18	1	1	346	0	0	1	1	353	0	0
1	1	6	0	0	140	10	1	2	340	0	0	1	1	362	0	0
1	1	10	0	0	88	46	1	2	346	0	0	1	2	348	0	0
1	1	12	0	0	264	13	1	2	705	0	0	1	121	947	0	0
1	1	16	0	0	88	21	1	1	441	0	0	1	1	443	0	0
1	1	20	0	0	264	27	1	2	439	0	0	1	2	440	0	0
1	1	24	0	0	88	4	1	1	500	0	0	0	0	0	0	0
1	1	26	0	0	140	2	1	1	503	1c4000	4000	0	0	0	0	0
1	1	30	0	0	140	2000	10	1	527	1b8000	4000	1	2	493	0	0
1	1	32	0	0	140	104	1	1	537	1c3000	4000	1	2	524	0	0

V1-11

RPT Tables

- Created at system initialization
- One maintained for each drawer on the system
- Information about each report in the drawer is contained in the associated table:
  - Number lines total
  - Number lines in header
  - Address on disk
- Can point to page table

RPT Tables

```
**************************************************
*                R P T   T A B L E               *
*          C A B - 2   DRAWER I - 040            *
**************************************************
```

CXWREV	rpt	lines	hdr	upd	bsz	dev	dad	link
001000	0	40	1	0	1	2	595	-1
001000	1	141	3	0	2	1	20812	111
					2	2	483	-2
001000	2	90	3	0	2	1	20814	-1
001000	3	246	3	0	2	2	597	0
					2	1	20816	112
					2	2	1789	-2
001000	4	175	3	0	2	1	20818	1
					2	2	599	-2
001000	5	32	3	0	1	2	603	-1
001000	6	51	3	0	2	1	20820	-1
001000	7	71	3	0	2	2	601	-1
001000	8	93	3	0	2	1	20822	2
					1	2	604	-2
001000	9	112	3	0	2	1	620	5
					1	2	607	-2
001000	10	13	3	0	1	1	622	-1
001000	11	41	3	0	1	2	608	-1
001000	12	54	3	0	2	1	623	-1
001000	13	55	3	0	2	2	609	-1
001000	14	25	3	0	1	1	20824	-1
001000	15	91	3	0	2	2	605	-1
001000	16	24	3	0	1	1	626	-1
001000	17	404	3	0	2	1	20825	3
					2	2	611	4
					2	1	20827	113
					2	1	20829	114
					1	2	438	-2

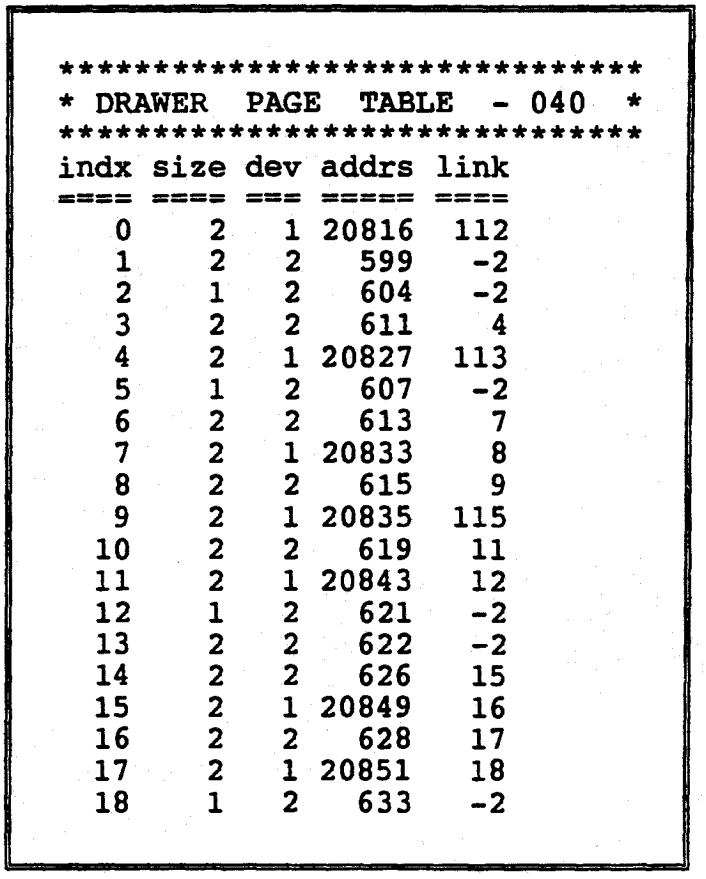
V1-12

Page Table

- Created only if a report contains more than 32K bytes

- Entries made up of pointers that point to areas of disk

Page Table



```
*********************************
* DRAWER  PAGE  TABLE  - 040  *
*********************************
indx size dev addrs link
==== ==== === ===== ====
   0    2   1 20816  112
   1    2   2   599   -2
   2    1   2   604   -2
   3    2   2   611    4
   4    2   1 20827  113
   5    1   2   607   -2
   6    2   2   613    7
   7    2   1 20833    8
   8    2   2   615    9
   9    2   1 20835  115
  10    2   2   619   11
  11    2   1 20843   12
  12    1   2   621   -2
  13    2   2   622   -2
  14    2   2   626   15
  15    2   1 20849   16
  16    2   2   628   17
  17    2   1 20851   18
  18    1   2   633   -2
```

Summary

- U Series MAPPER runs as an application or program under UNIX.
- MAPPER must be brought up on UNIX, MAPPER software executed, and user sign-on occur before functions or a run can be executed.
- MAPPER programs, system tables, processes, are all kept resident, and reports are brought into memory in their entirety if memory contention is not tight.
- The unit table contains information about the user-id, device type and status information.
- MAPPER interprets a user entry in the following order:
  - Manual function a user may execute? (user D-bank)
  - Department run? (department run registration report)
  - Global run? (master run registration report)
- PreRun is a process that validates run names and checks for user restrictions.
- The MAPPER database is a series of files (maximum 14) on disk.
  up to 1024 MAPPER files.
- Character special files are recommended as MAPPER handle I/O instead of UNIX, is immediate, and database integrity is maintained.
- Database files are initialized and updated in a "round robin" method.
- Additional system tables:
  - Drawer table
  - RPT tables
  - Page tables

Exercises

Matching: Letters may be used more than once.

1. ___ Run Registration Report
2. ___ PreRun
3. ___ Configuration Table
4. ___ Drawer Table
5. ___ User Registration Report
6. ___ Unit Table
7. ___ User D-bank
8. ___ Page Table
9. ___ RPT Table

A. Always resident; points to PRT tables.
B. Contains information about devices.
C. Contains information ablut the user-id, device name, status information.
D. An area of memory that keeps track of what the user is doing and what he is allowed to do.
E. Report that indicates what functions a user is allowed to execute.
F. Determines if an entry on the control line is a valid run name.
G. Lists line number limitations for a report.
H. Built only if a report is 32K or greater in size.
I. Maintained for each report on the system.

Step through the search sequence:

1.
2.
3.

Step through user sign on:

1.
2.
3.

Identify three MAPPER database characteristics:

1.
2.
3.

2

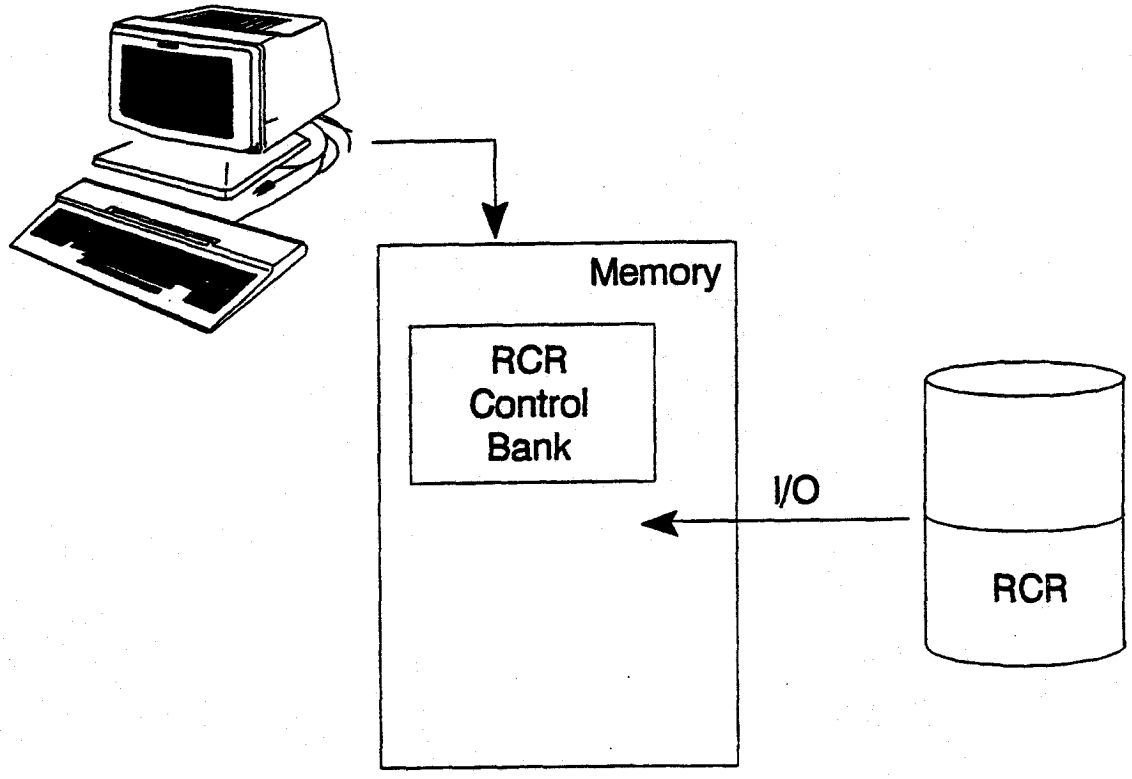
Basics of RCR Execution

Module Objectives

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

1. Identify how MAPPER locates an RCR and prepares memory for its execution.
2. Define the term buffer.
3. Identify the purpose of the run interpreter.
4. Identify the difference between RCR and function processing.

RCR Structure

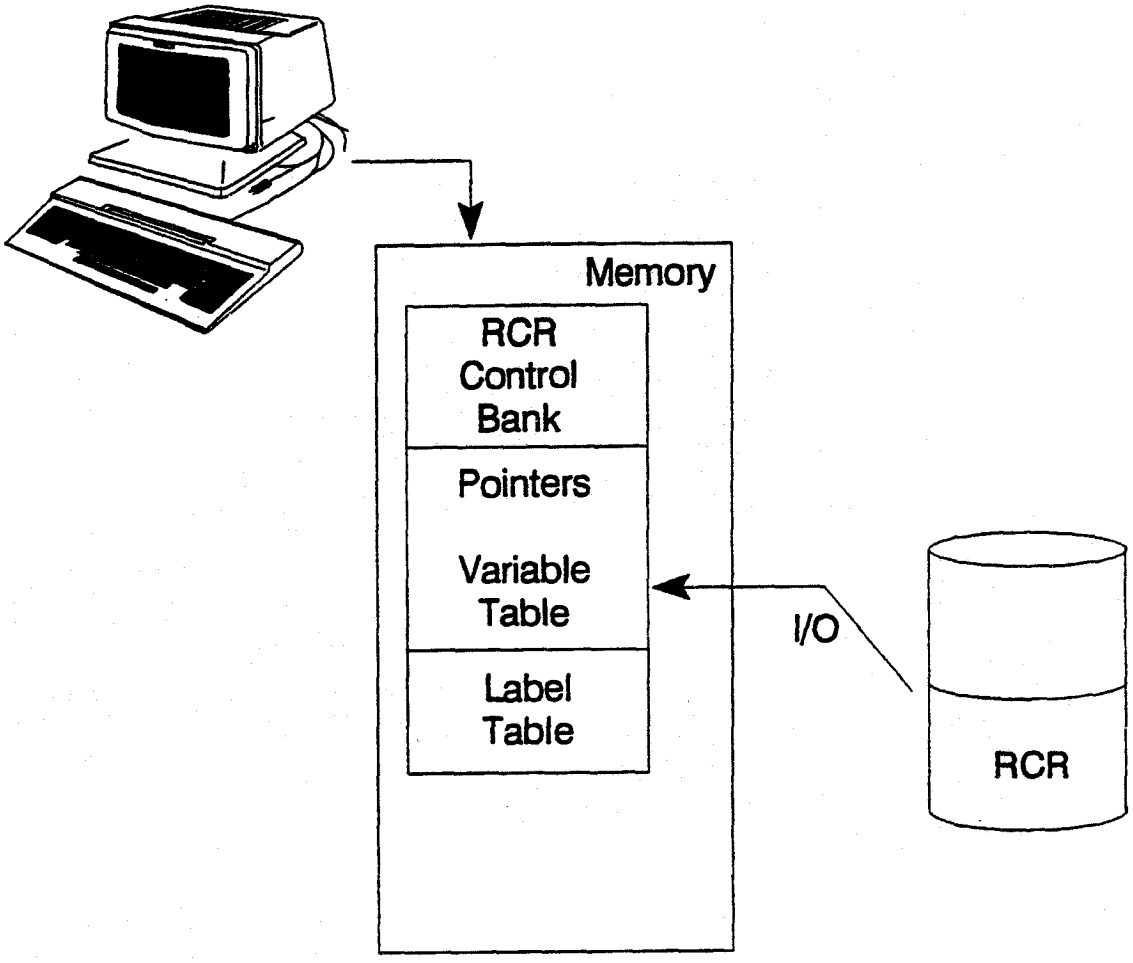


V2-1

Control Bank

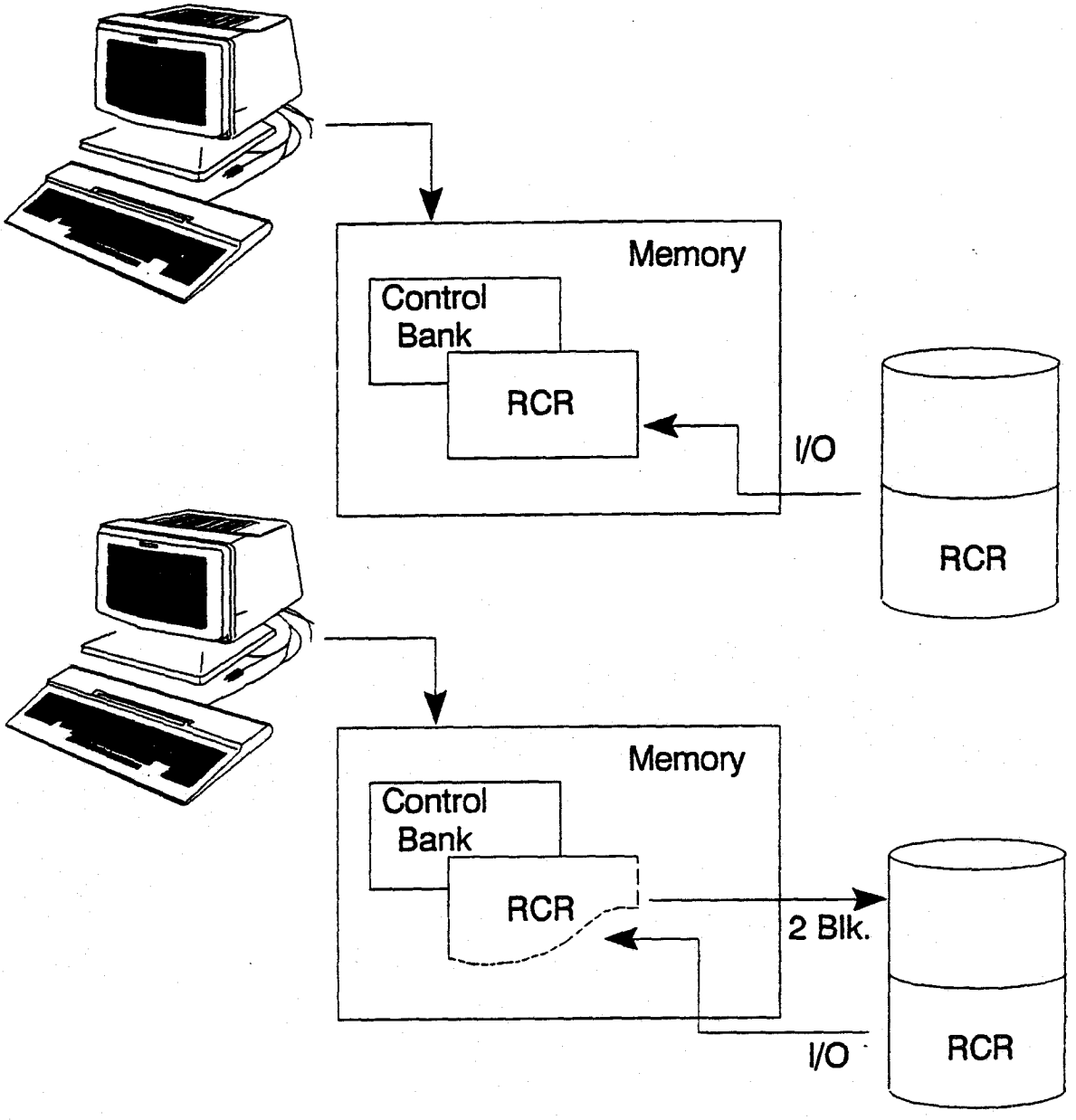
- A logical unit of memory that contains all necessary information for proper run execution to occur. May contain:
  - Pointers to sections of memory
  - Variable table
  - Label table
- Commonly referred to as the RCR's D-bank.
- Each RCR has its own control bank.

Control Bank



V2-2

Memory Allocation



V2-3

Memory Allocation

- Entire report will be pulled into memory if possible

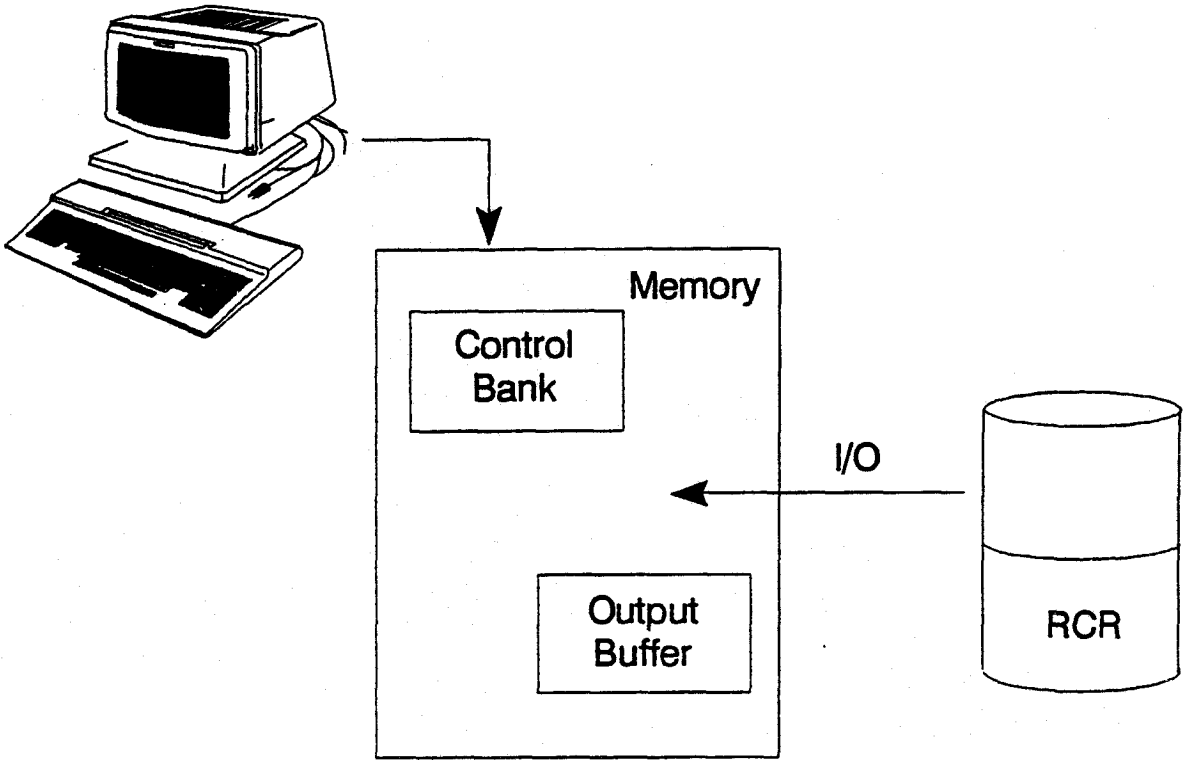
- Report loaded in two block increments

- Memory allocation handled by MAPPER's Memory Manager, which handles the following:
  - Initial allocation of memory
  - Determines which blocks are swapped out and which remain, if swapping is necessary
  - Swapping blocks in/out of memory

Buffers, Blocks, and I/Os

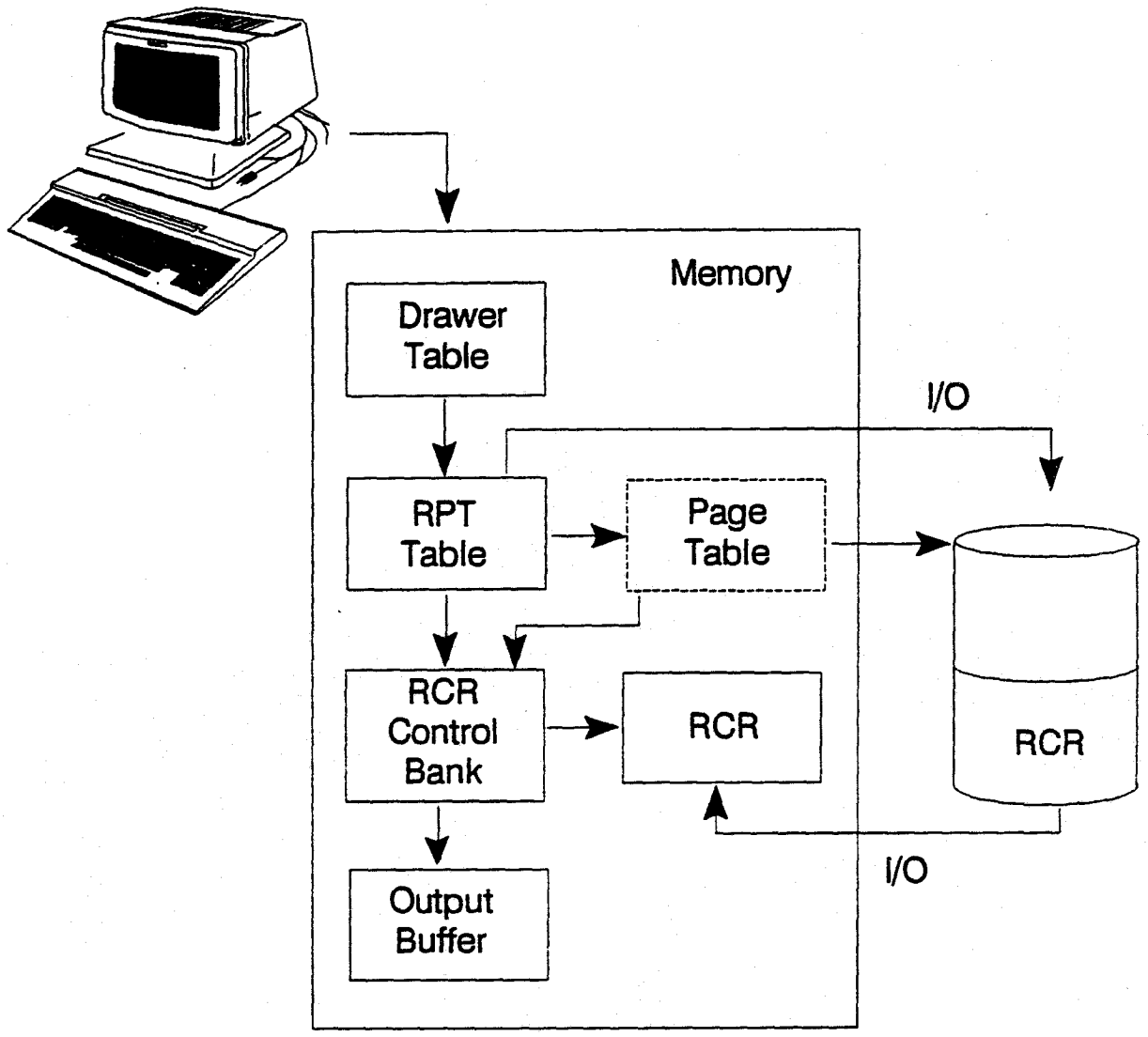
- One block = 4096 bytes
- One buffer = two blocks maximum
- One I/O = one buffer, two blocks maximum

Output Area



V2-4

RCR Setup Summary

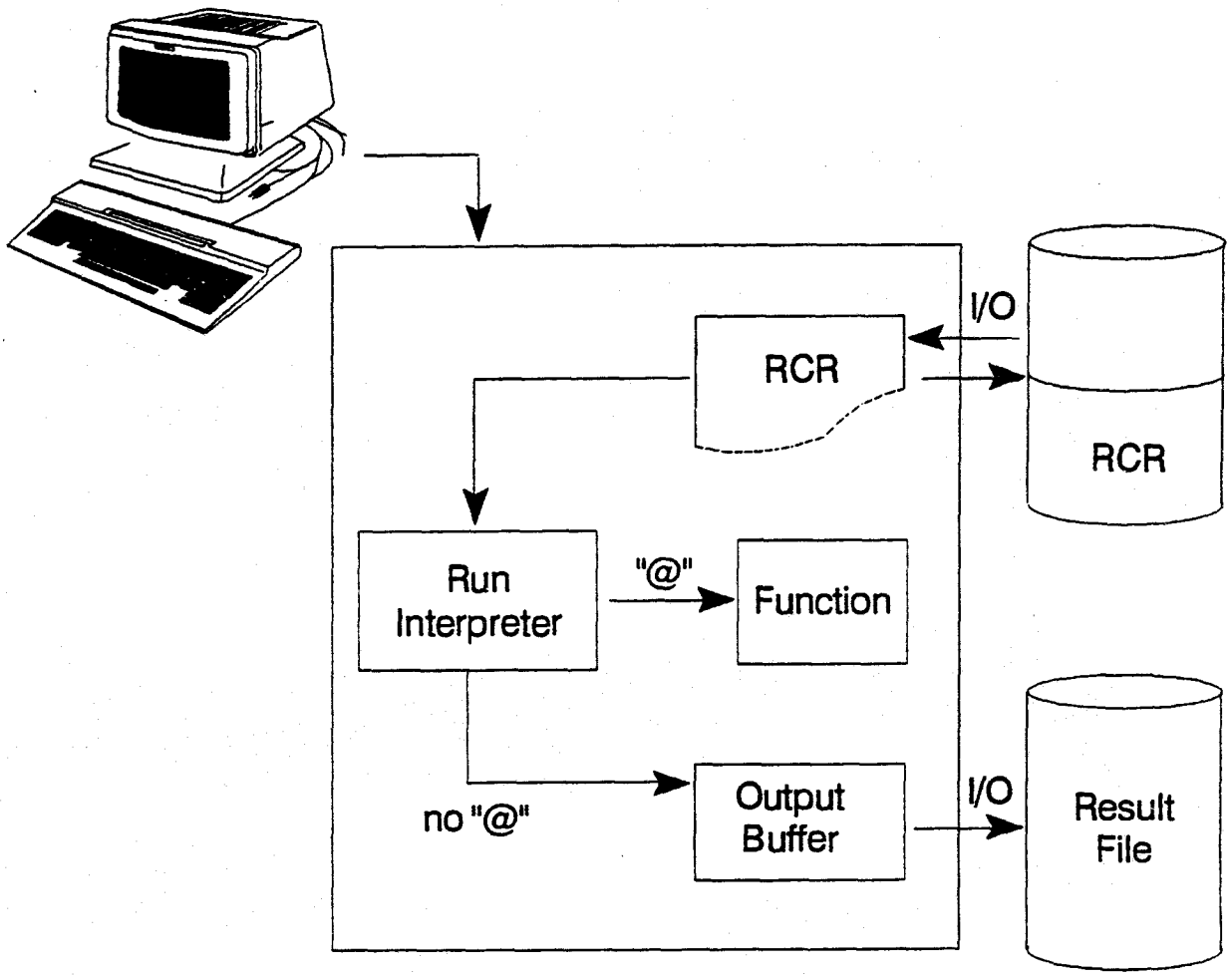


V2-5

Run Interpreter

- Interprets RCR in memory
- If "@" present in column one, control passes to the function
- If "@" not present in column one, the line is placed in the output area and the next line examined
- Any saved results are written to the result database file.

Run Interpreter



V2-6

Function Processing

- Run Interpreter encounters "@".
- Determines if the function is valid by checking the user's D-bank.
- All functions are resident.
- A function D-bank is created.

RCR Characteristics

- 40 columns minimum, 256 columns maximum.
- 64,000 lines maximum.
- ASCII/Full Character Set is used:
  - Use of both upper and lower case letters.
  - Numerals 0-9.
  - Special characters.
- Less than 93 lines, 80 columns are most efficient.

Report Size Calculation

$$\text{(Lines per Report)} = \frac{\text{(Number of Total Bytes)}}{\text{(Number of Bytes per Line)}}$$

(Number of Bytes per Line) = (Number of Characters per Line + 8)

V2-7

Block Number Calculation

$$\text{(Number of Blocks in Report)} = \frac{\text{(Number of Bytes per Report)}}{\text{4096 (1 Block)}}$$

(Number of Bytes per Report) = (Number of Lines x Bytes per Line)

(Number of Bytes per Line) = (Number of Characters per Line + 8)

CAH Statement

- The Cache Report (CAH) statement loads (caches all of) the specified report into memory for faster processing.

- If enough memory is not available to cache the report, the report is processed as if the CAH statement were not executed.

CAH Statement

Description

Loads all of the specified report into memory for faster processing.

Format

```
@CAH,c,d,r .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of the report to be processed.

Example

```
@cah,0,b,3 .
```

Cache report 3B in cabinet 0.

Summary

- MAPPER locates a report on the database via the drawer, RPT, and the page table.
- Before the report can be brought into memory, a control bank must be built and memory allocated.
- The control bank is an area of memory that contains pointers to other areas of memory, the variable table, and the label table.
- Memory size is determined by MAPPER's memory manager.
- The output buffer (output area) is allocated.
- The report is read in in its entirety if memory permits ... only one read. If memory constraints exist, the remainder of the report is swapped in when needed in maximum two-block increments.
- One block equals 4096 bytes, regardless of UNIX model.
- One buffer (a unit of memory) equals, at most, two blocks.
- MAPPER's run interpreter reads the RCR and places statements without "@" in column one in the output area; those with "@" are processed as functions.
- Functions without manual counterparts are faster.
- Report line length can be from 40 to 256 characters; 80 is most efficient.
- Report size can be 64,000 lines; 16,768,000 bytes maximum (256 x 64,000). Less than 8192 is optimum.

Exercises

Matching: Letters may be used more than once.

1. B,C RCR D-bank
2. F Buffer
3. ___ Memory Manager
4. F One block
5. ___ Output buffer
6. G Run Interpreter
7. A Function Processing
8. C CAH Statement

A. Checks user D-bank to see if run is valid.
B. Contains all necessary information for run execution to occur.
C. Loads a report into memory for faster processing.
D. Handles memory allocation.
E. Temporary storage area in memory to hold information.
F. 4096 bytes.
G. Interprets RCR in memory.
H. Stores processed information in memory.

Identify how MAPPER locates an RCR and prepares memory for its execution.

1.
2.
3.
4.

Define the term buffer.

Define the purpose of the Run Interpreter.

What is the difference between RCR and function processing?

3

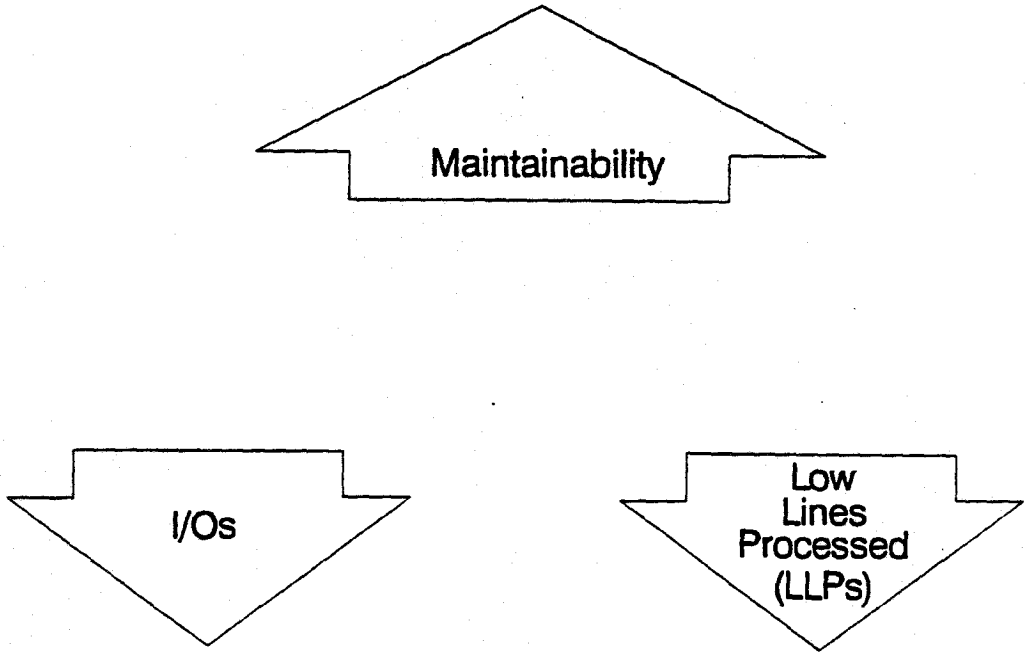
Efficiency and Analysis

Module Objectives

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

1. Define and distinguish the difference between I/Os, LLPs, and DLPs.
2. State the purpose of the Accounting Log.
3. Interpret and utilize LOG, LOGLA, and RUNA results.

What is Efficiency?

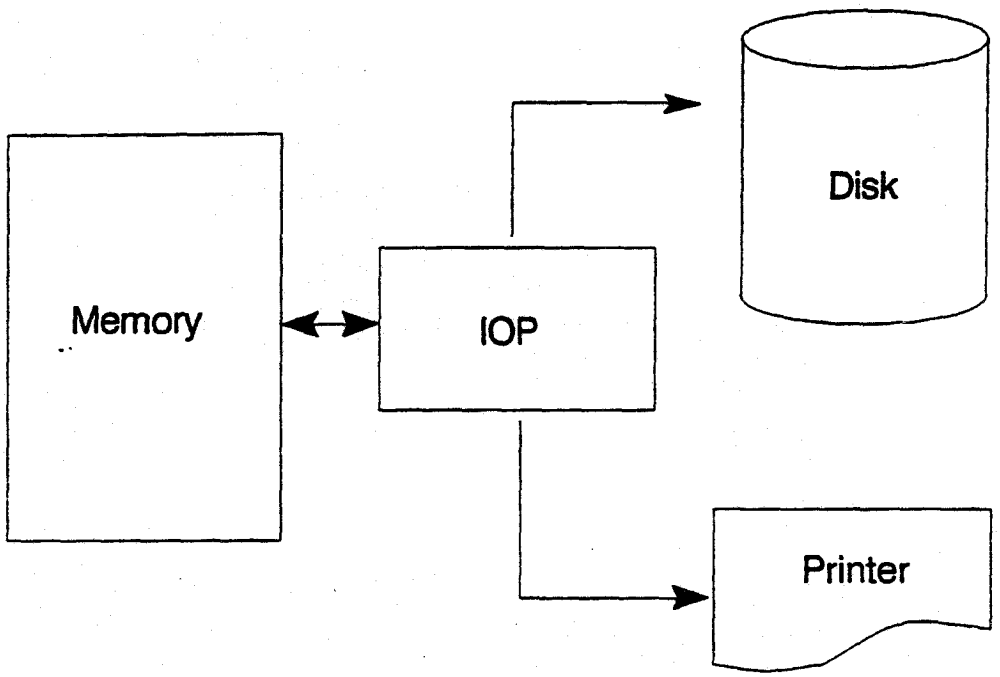


V3-1

I/Os

- Input/Output.
- Input/Output Processor retrieves and transfers data between peripheral devices and memory.
- Reserve word IO$ captures I/O count.
- Expensive in terms of time and money.

I/Os



V3-2

I/O Limits

- The Run Registration Report contains a field for I/O limits.
- Each run must have an I/O limit registered.
- <I/Os exceeded> will appear as an error message on the Control Line if I/O limit is exceeded.

I/O Limits

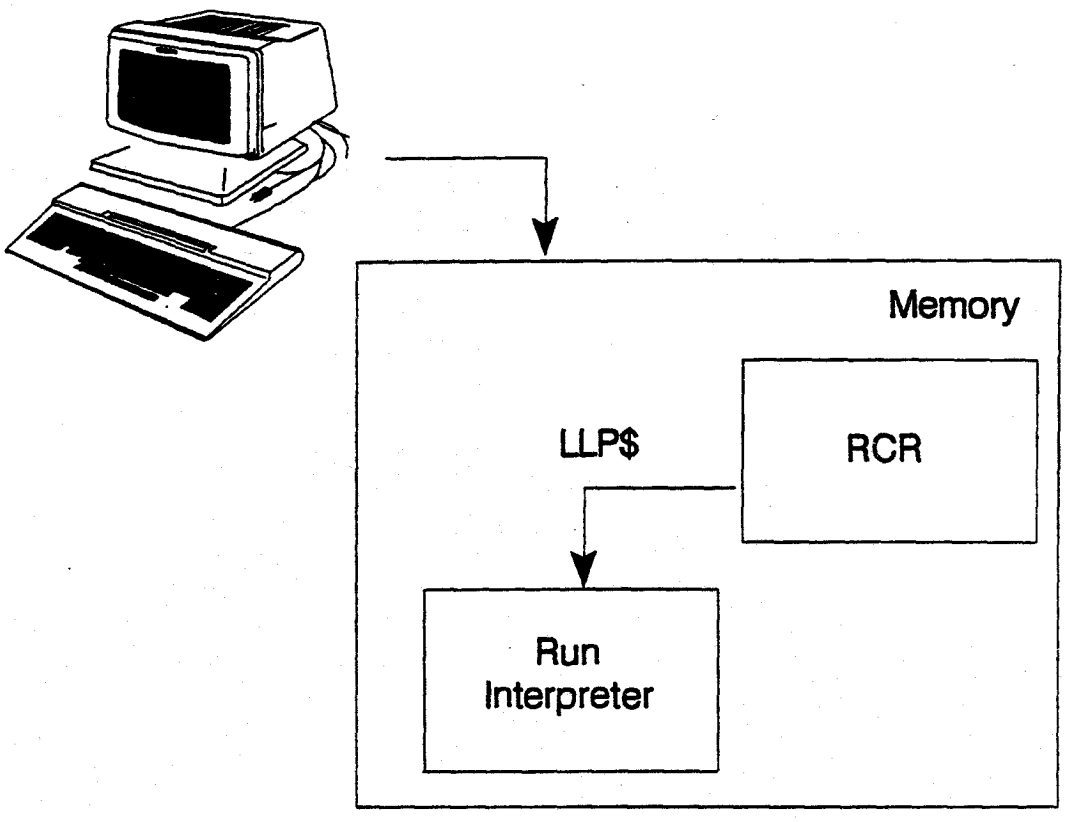
```
.DATE   28 SEP 89  15:03:31  RID    7E   03 JUL 86  MAPCOORD
. RUN REGISTRATION REPORT FOR DEPARTMENT: 'Dept.  name'                  E0030
*  RUN NAME  .    USER    .UNIT.DRAWER. RPT.F. B.TIME . E.TIME . I/O  LINES. CAB
*-----------.----------.----.------.----.-.-------.-------.-----.-----.----
 SAMPLERUN                      e0    6                        1000  2000 0,2,
 CORREL                         i0    2                        1000  2000 0,
 WEEKRP                         e2    1                        1000  2000 20,2
 MONTHRP                        d0    2                        1000  2000 20,2
 QUARTRP                        d0    3                        1000  2000 20,2
                          ..... END REPORT .....
```

V3-3

LLPs

- Logic Lines Processed (LLPs).
- Any line read/processed in the RCR by the Run Interpreter.
- Reserve word LLP$ captures LLP count.
- Can be expensive in terms of I/Os.

LLPs

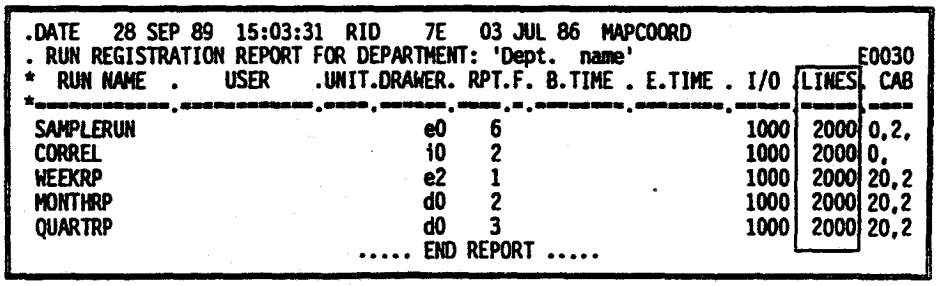


V3-4

LLP Limits

- The Run Registration Report contains a field for LLP limits.
- Each run must have an LLP limit registered.
- <Logic Line Processed Exceeded> will appear as an error message on the Control Line if LLP limit is exceeded.

LLP Limits



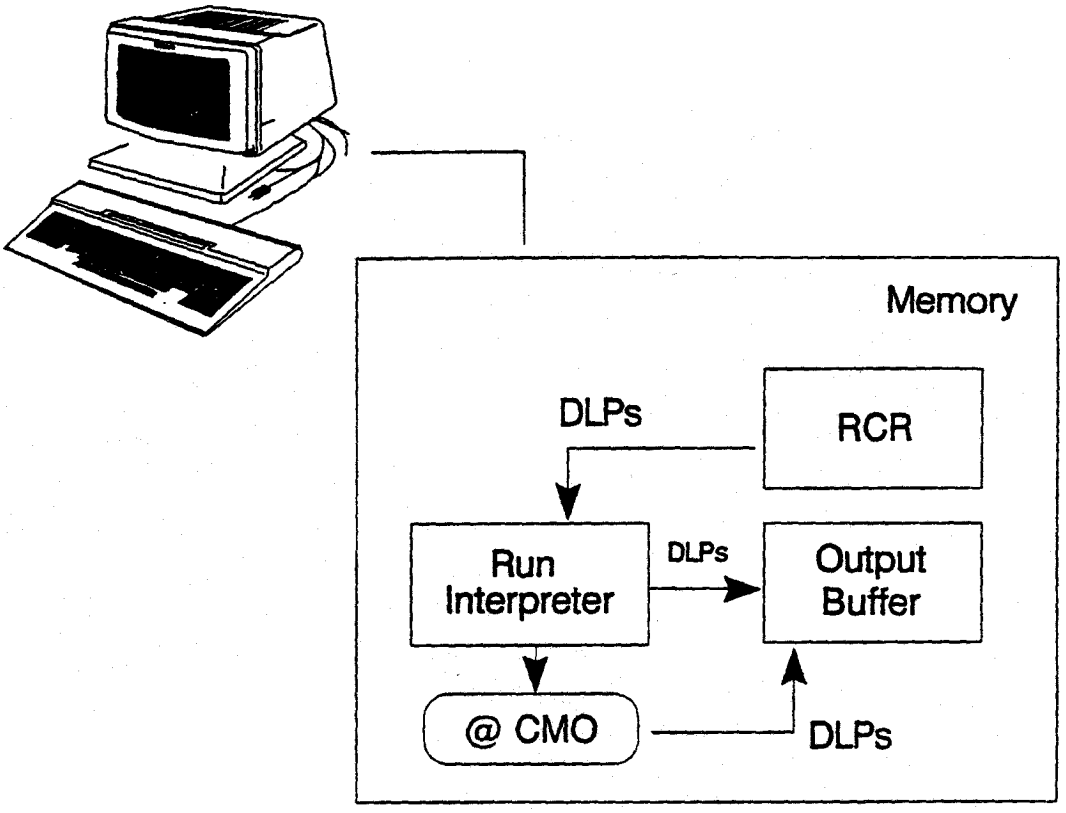
```
.DATE   28 SEP 89  15:03:31  RID    7E   03 JUL 86  MAPCOORD
. RUN REGISTRATION REPORT FOR DEPARTMENT: 'Dept.  name'                  E0030
*  RUN NAME  .    USER    .UNIT.DRAWER. RPT.F. B.TIME . E.TIME . I/O  LINES  CAB
*----------.----------.----.------.----.-.-------.-------.-----.------.----
 SAMPLERUN                      e0    6                      1000   2000  0,2,
 CORREL                         i0    2                      1000   2000  0,
 WEEKRP                         e2    1                      1000   2000  20,2
 MONTHRP                        d0    2                      1000   2000  20,2
 QUARTRP                        d0    3                      1000   2000  20,2
                          ..... END REPORT .....
```

V3-5

DLPs

- Data Lines Processed.
- Any line read/processed in the RCR by the Run Interpreter throughout the duration of the run.
- Reserve word DLP$ captures DLP count.
- Can be expensive in terms of I/Os.

DLPs


Memory
DLPs
RCR
Run Interpreter
DLPs
Output Buffer
@ CMO
DLPs

V3-6

I/Os, LLPs, and DLPs

RCR

```
1.  .Date 18 Jul 89     140:08:59     Rid 22E
2.  .Run Function Data:
3.  *=================================================
4.  @srh,0,8,9 DH 2-2 ,Sh .
5.  @ldv,w v1i3=IO$,v2i3=LLP$, v3i3=DLP$
6.        IO$ = v1        LLP$ = v2          DLP$ = v3
7.  @Gto end.
8.              ***** End Report *****
```

Source Report

```
1.  .Date 18 Jul 89        14:08:59            Rid 9B
2.  .                Corporate Production Status
3.  *St. Status.  By. Product.   Serial.  Produc. Order.  Cust.
4.  *Cd. Date.    In. Type.      Number.   Cost.  Number. Code.
5.  *=============================================================
6.  lp   831224   Ls  Blackbox1  436767           84389   Amco
7.  lp   831225   Ls  Blackbox1  436768           84390   Amco
8.  lp   831219   Ls  Blackbox2  637071           84353   Intr
9.  Or   840110   Ls  Blackbox4                   94754   Arco
10. Sh   840110   Ls  Blackbox5  675281           97441   Feds
11.                      ***** End Report *****
```

V3-7a

I/Os, LLPs, and DLPs

Search Result

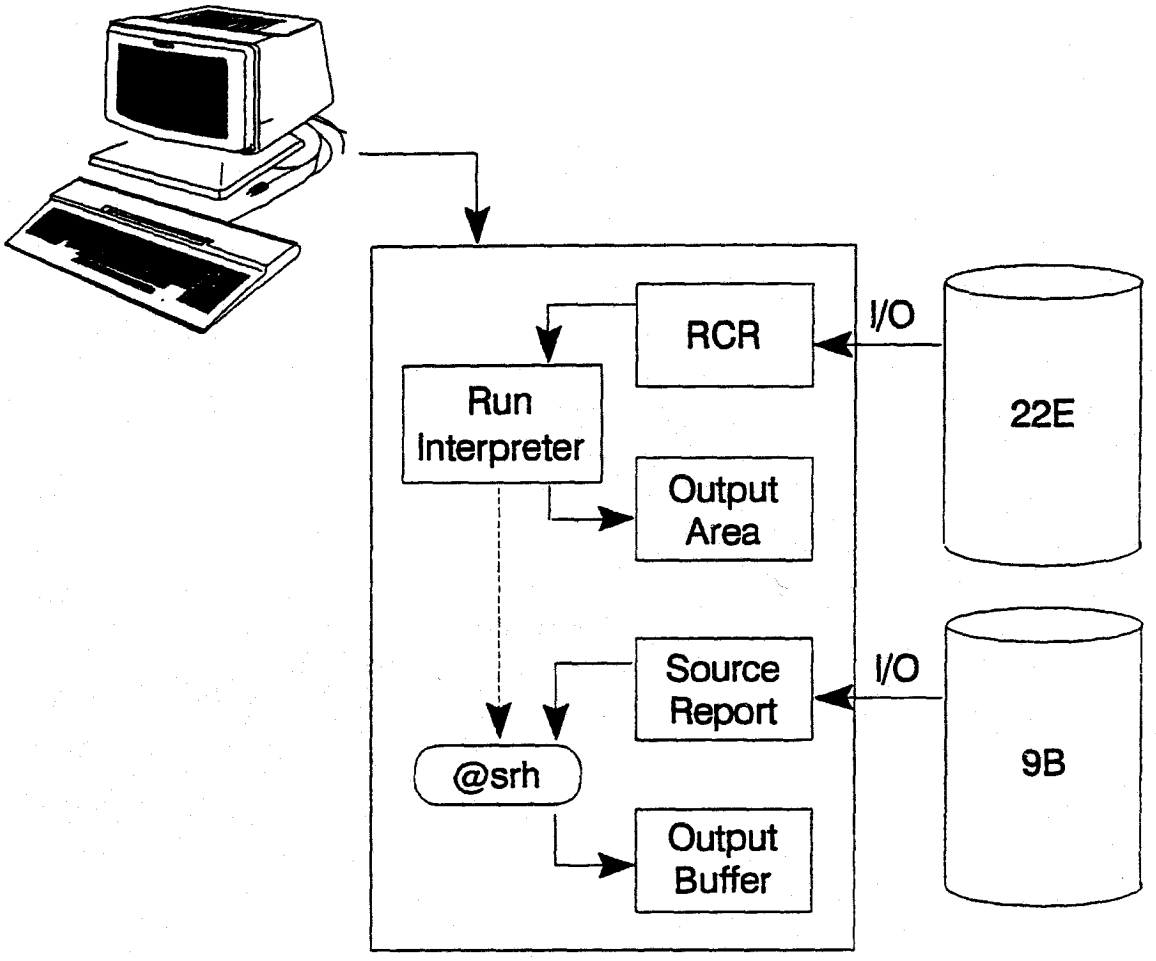
```
1.  .Date 18 Jul 89        14:08:59
2.  .                 Corporate Production Status
3.  *St. Status.  By. Product.   Serial.  Produc. Order.  Cust.
4.  *Cd. Date.    In. Type.      Number.   Cost.  Number. Code.
5.  *=============================================================
6.  Sh   840110   Ls  Blackbox5  675281          97441   Feds
7.                        ***** End Report *****
```

Output Area

```
1.  .Date 18 Jul 89        14:08:59        Report Generation
2.  *=============================================================
3.      IO$ = 2            LLP$ = 3            DLP$ = 23
4.                        ***** End Report *****
```

V3-7b

I/Os, LLPs, and DLPs



V3-8

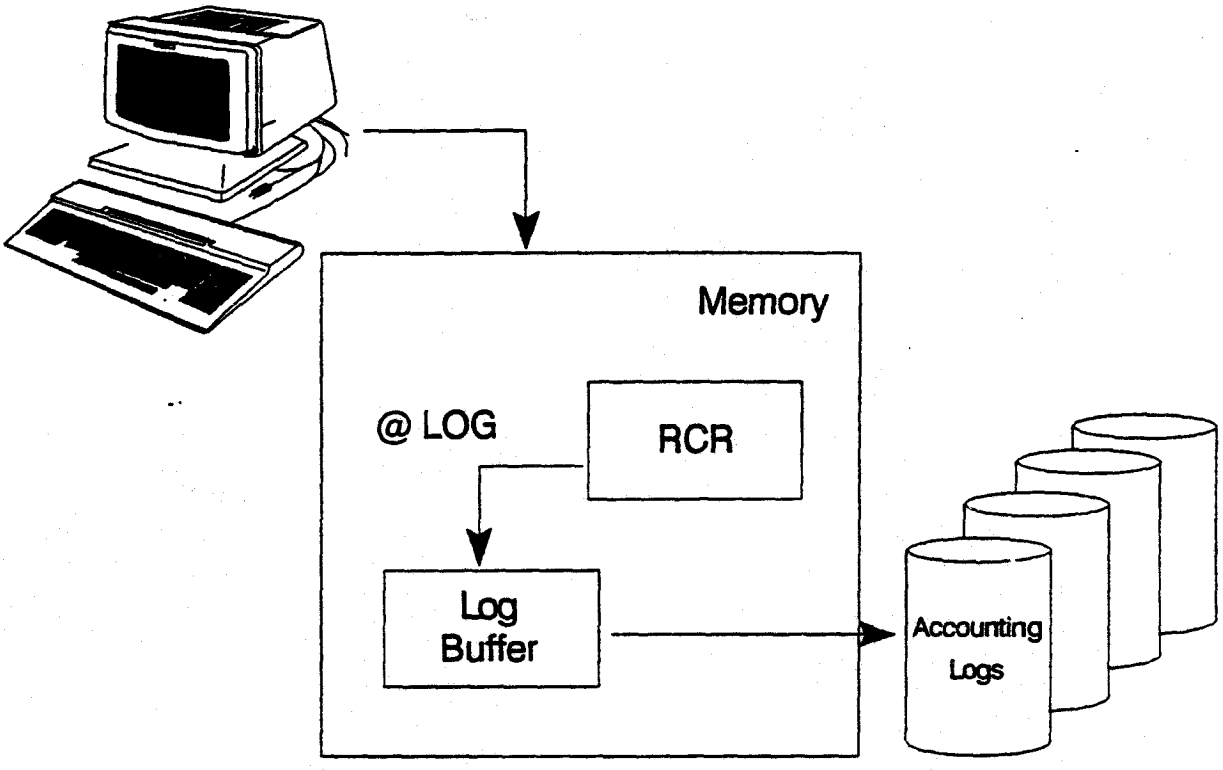
Analysis Tools

- LOG
- LOGLA
- RUNA

Accounting Log

- Maintained specifically for logging information.
- All run activities and transactions are kept here chronologically.
- Aids in recovery or restart of RCR if necessary.
- Located in /mapper/tmp directory.

Accounting Log



V3-9

@LOG

- Generates a log listing.
- Entered as first statement of RCR.
- Produces an entry in the accounting log file for each function executed in the run.
- Delete when run evaluation and debugging is complete.

@LOG Example

```
.DATE              15:08:18  RID   23E   28 SEP 89  MAPCOORD
.@       REV. 03.055W  880621     'MARK' EXAMPLE RUN                          E0010
*-------------------------------------------------------------------------------
@LOG .
@MCH,0,C,1,0,D,1 '' 'PRODUCT TYPE','RETAIL'  ,1,A \              MATCH FOR $
'PRODUCT TYPE','UNIT RETAIL'  ,1,A .
@CAL,-0 '' 'ORD QTY','UNIT RETAIL','EXTENDED RETAIL' \     QTY * COST = EXT$
 ,A,B,C C=A*B;QTY=VSUM(A);RETAIL=VSUM(C) <QTY>I3,<RETAIL>I8 .     SUM QTY & EXT$
@RNM -1 .
@JUV,C <RETAIL> .
@BRK .

 CORPORATE ORDER STATUS: RETAIL VALUE AND ORDER QUANTITIES
 GRAND TOTAL ORDER QTY: <QTY>, RETAIL VALUE: $<RETAIL>

 CUSTOMER   ORDER QTY    RETAIL VALUE
*--------  .--------   .-----------
@SUB,-1 J(L) 'ORD QTY','EXTENDED RETAIL','CUSTOMER(1-8)' \   SUBTOTAL
 ,+,+,S <QTY>I,<RETAIL>I,
<CUST>H . <CUST>      <QTY>          <RETAIL>
@BRK OUT,0,E,-0,2,23,1,0,Y,,,P .
@REL .
                        ..... END REPORT .....
```

```
.DATE              15:08:18   28 SEP 89  Report Generation
*-------------------------------------------------------------------------------
 CORPORATE ORDER STATUS:  RETAIL VALUE AND ORDER QUANTITIES
 GRAND TOTAL ORDER QTY:  16, RETAIL VALUE: $ 380,800

 CUSTOMER   ORDER QTY    RETAIL VALUE
*--------  .--------   .-----------
.AMERICAN  .     4     .      104650
.ARGENTIN  .     4     .       99575
.DIGITAL   .     1     .       25375
.FED SYST  .     4     .       75250
.INTERNAT  .     1     .       24150
.UNION ST  .     2     .       51800
```

V3-10a

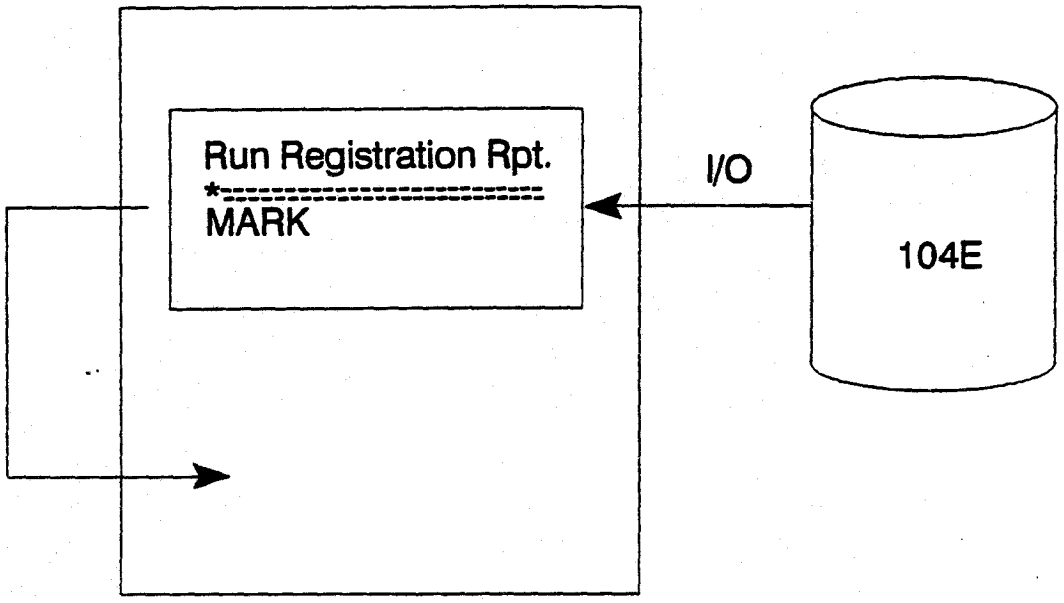
@LOG Example

```
.DATE              17:29:52  RID   10B   24 JUL 89  MAPCOORD
.LOG LIST: FROM  17:29:31  24 JUL 89  TO  17:29:35  24 JUL 89                B0102
.STA=9
```

* USER	FUNCTION	TIME	DRAWR.	RPT	LINES-IN.	LINE-OUT.	READS.	WRITES
MAPCOORD	PRE-RUN	17:29:31	30	104	78	0	1	0
MAPCOORD	RUN	17:29:31	10	3	0	0	0	0
MAPCOORD	MATCH*	17:29:31	6	-0	181	60	0	0
MAPCOORD	CALCULATE*	17:29:31	6	-0	34	20	0	0
MAPCOORD	TOTALIZE*	17:29:31	6	-1	48	6	0	0
MAPCOORD	RUN	17:29:31	10	3	20	13	0	0
MAPCOORD	DISPLAY	17:29:31	10	-0	13	0	0	0
MAPCOORD	RUN	17:29:33	10	3	1	2	0	0
MAPCOORD	DISPLAY	17:29:33	10	-0	2	0	0	0

V3-10b

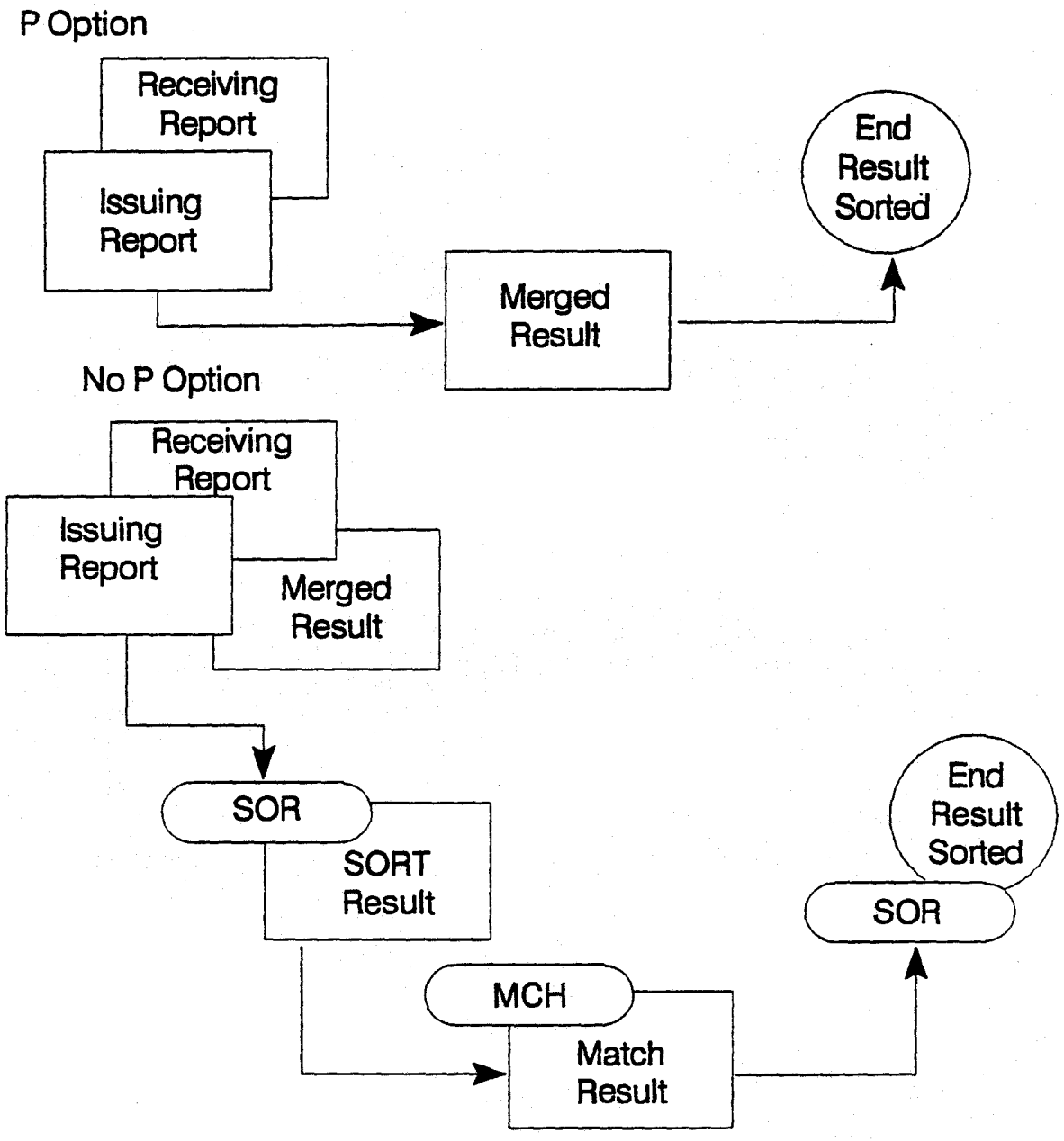
Log List
PreRun




```
.DATE              17:29:52  RID   10B   24 JUL 89  MAPCOORD
.LOG LIST: FROM  17:29:31  24 JUL 89  TO  17:29:35  24 JUL 89              80102
.STA=9
*   USER     . FUNCTION    .  TIME   .DRAWR. RPT .LINES-IN.LINE-OUT. READS. WRITES
*----------.-----------.--------.-----.-----.--------.--------.------.-------
 MAPCOORD     PRE-RUN      17:29:31    30   104       78        0      1       0
 MAPCOORD     RUN          17:29:31    10     3        0        0      0       0
 MAPCOORD     MATCH*       17:29:31     6    -0      181       60      0       0
 MAPCOORD     CALCULATE*   17:29:31     6    -0       34       20      0       0
 MAPCOORD     TOTALIZE*    17:29:31     6    -1       48        6      0       0
 MAPCOORD     RUN          17:29:31    10     3       20       13      0       0
 MAPCOORD     DISPLAY      17:29:31    10    -0       13        0      0       0
 MAPCOORD     RUN          17:29:33    10     3        1        2      0       0
 MAPCOORD     DISPLAY      17:29:33    10    -0        2        0      0       0
```

V3-11

Log List
Match Function



V3-12

Log List
Calculate Function

```
.DATE              15:08:18  RID   23E   28 SEP 89  MAPCOORD
.@       REV. 03.055W  880621      'MARK' EXAMPLE RUN                    E0010
*=============================================================================
@LOG .
@MCH,0,C,1,0,D,1 '' 'PRODUCT TYPE','RETAIL'  ,1,A \              MATCH FOR $
'PRODUCT TYPE','UNIT RETAIL'  ,1,A .
@CAL,-0 '' 'ORD QTY','UNIT RETAIL','EXTENDED RETAIL' \     QTY * COST = EXT$
 ,A,B,C C=A*B;QTY=VSUM(A);RETAIL=VSUM(C) <QTY>I3,<RETAIL>I8 .     SUM QTY & EXT$
@RNM -1 .
@JUV,C <RETAIL> .
@BRK .

 CORPORATE ORDER STATUS: RETAIL VALUE AND ORDER QUANTITIES
 GRAND TOTAL ORDER QTY: <QTY>, RETAIL VALUE: $<RETAIL>

 CUSTOMER   ORDER QTY    RETAIL VALUE
*========  .========   .============
@SUB,-1 J(L) 'ORD QTY','EXTENDED RETAIL','CUSTOMER(1-8)' \   SUBTOTAL
 ,+,+,S <QTY>I,<RETAIL>I,
<CUST>H . <CUST>       <QTY>           <RETAIL>
@BRK OUT,0,E,-0,2,23,1,0,Y,,,P .
@REL .
                        ..... END REPORT .....
```

```
.DATE              17:29:52  RID   10B   24 JUL 89  MAPCOORD
.LOG LIST: FROM  17:29:31  24 JUL 89  TO  17:29:35  24 JUL 89              B0102
.STA=9
*   USER     . FUNCTION    .  TIME  .DRAWR. RPT .LINES-IN.LINE-OUT. READS. WRITES
*============.============.========.======.=====.========.========.======.=======
MAPCOORD      PRE-RUN       17:29:31    30   104        78         0      1        0
 MAPCOORD     RUN           17:29:31    10     3         0         0      0        0
 MAPCOORD     MATCH*        17:29:31     6    -0       181        60      0        0
 MAPCOORD     CALCULATE*    17:29:31     6    -0        34        20      0        0
 MAPCOORD     TOTALIZE*     17:29:31     6    -1        48         6      0        0
 MAPCOORD     RUN           17:29:31    10     3        20        13      0        0
 MAPCOORD     DISPLAY       17:29:31    10    -0        13         0      0        0
 MAPCOORD     RUN           17:29:33    10     3         1         2      0        0
 MAPCOORD     DISPLAY       17:29:33    10    -0         2         0      0        0
```

V3-13

Log List
Display Functions

```
.DATE              15:08:18  RID   23E   28 SEP 89  MAPCOORD
.@       REV. 03.055W  880621      'MARK' EXAMPLE RUN                        E0010
*=============================================================================
@LOG .
@MCH,0,C,1,0,D,1 '' 'PRODUCT TYPE','RETAIL'  ,1,A \              MATCH FOR $
'PRODUCT TYPE','UNIT RETAIL'  ,1,A .
@CAL,-0 '' 'ORD QTY','UNIT RETAIL','EXTENDED RETAIL' \     QTY * COST = EXT$
 ,A,B,C C=A*B;QTY=VSUM(A);RETAIL=VSUM(C) <QTY>I3,<RETAIL>I8 .     SUM QTY & EXT$
@RNM -1 .
@JUV,C <RETAIL> .
@BRK .

 CORPORATE ORDER STATUS: RETAIL VALUE AND ORDER QUANTITIES
 GRAND TOTAL ORDER QTY: <QTY>, RETAIL VALUE: $<RETAIL>

 CUSTOMER   ORDER QTY    RETAIL VALUE
*========  .========   .===========
@SUB,-1 J(L) 'ORD QTY','EXTENDED RETAIL','CUSTOMER(1-8)' \   SUBTOTAL
 ,+,+,S <QTY>I,<RETAIL>I,
<CUST>H . <CUST>      <QTY>          <RETAIL>
@BRK OUT,0,E,-0,2,23,1,0,Y,,,P .
@REL .
                        ..... END REPORT .....
```

```
.DATE              17:29:52  RID   10B   24 JUL 89  MAPCOORD
.LOG LIST: FROM  17:29:31  24 JUL 89  TO  17:29:35  24 JUL 89              B0102
.STA=9
*   USER     . FUNCTION    .  TIME   .DRAWR. RPT .LINES-IN.LINE-OUT. READS. WRITES
*============.============.========.=====.=====.========.========.======.=======
 MAPCOORD     PRE-RUN       17:29:31    30    104        78         0      1       0
 MAPCOORD     RUN           17:29:31    10      3         0         0      0       0
 MAPCOORD     MATCH*        17:29:31     6     -0       181        60      0       0
 MAPCOORD     CALCULATE*    17:29:31     6     -0        34        20      0       0
 MAPCOORD     TOTALIZE*     17:29:31     6     -1        48         6      0       0
 MAPCOORD     RUN           17:29:31    10      3        20        13      0       0
 MAPCOORD     DISPLAY       17:29:31    10     -0        13         0      0       0
 MAPCOORD     RUN           17:29:33    10      3         1         2      0       0
 MAPCOORD     DISPLAY       17:29:33    10     -0         2         0      0       0
```

V3-14

Log List
Interim vs. Noninterim Displays

- Interim displays are information displayed on the screen for a given length of time; the run then continues automatically.

  Reserve word (IO$, LLP$, and DLP$) counts are maintained.

- Noninterim displays allow the user to perform some manual functions or some other processing during the display.

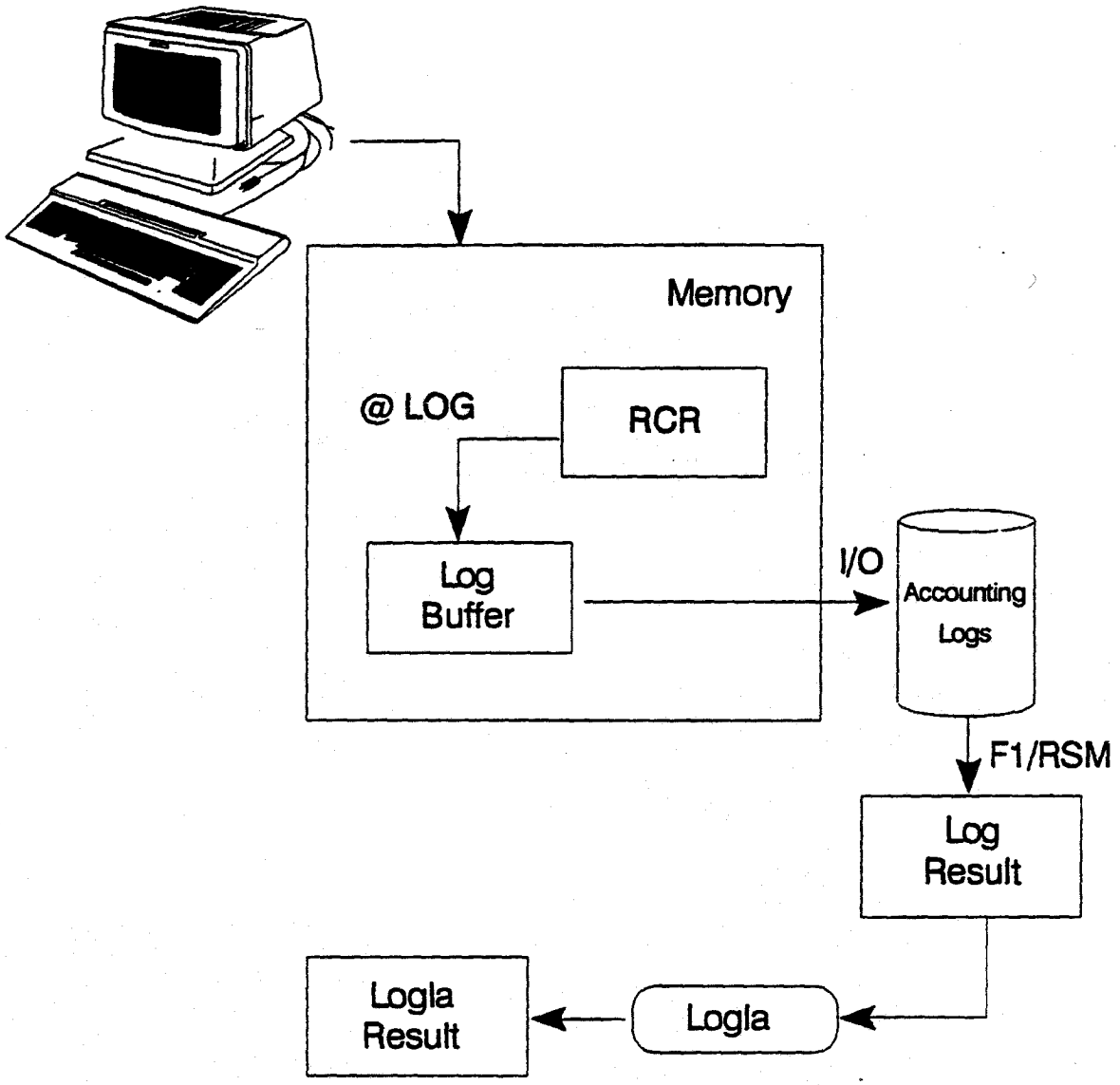
  CAUTION! Reserve word IO$, LLP$, and DLP$ are set back to zero! I/O and LLP limits are reset.

Logla

- Performs a further analysis of the log listing.

- Display log listing.
  - Home cursor.
  - Enter "logla" on the control line and transmit.

Logla


Memory
@ LOG
RCR
Log
Buffer
I/O
Accounting
Logs
F1/RSM
Log
Result
Logla
Result
Logla

V3-15

Logla Listing
Parts and Transactions

- Similar to log listing.

- Provides a detailed summary of the run.
  - Parts
  - Transactions

Logla Listing

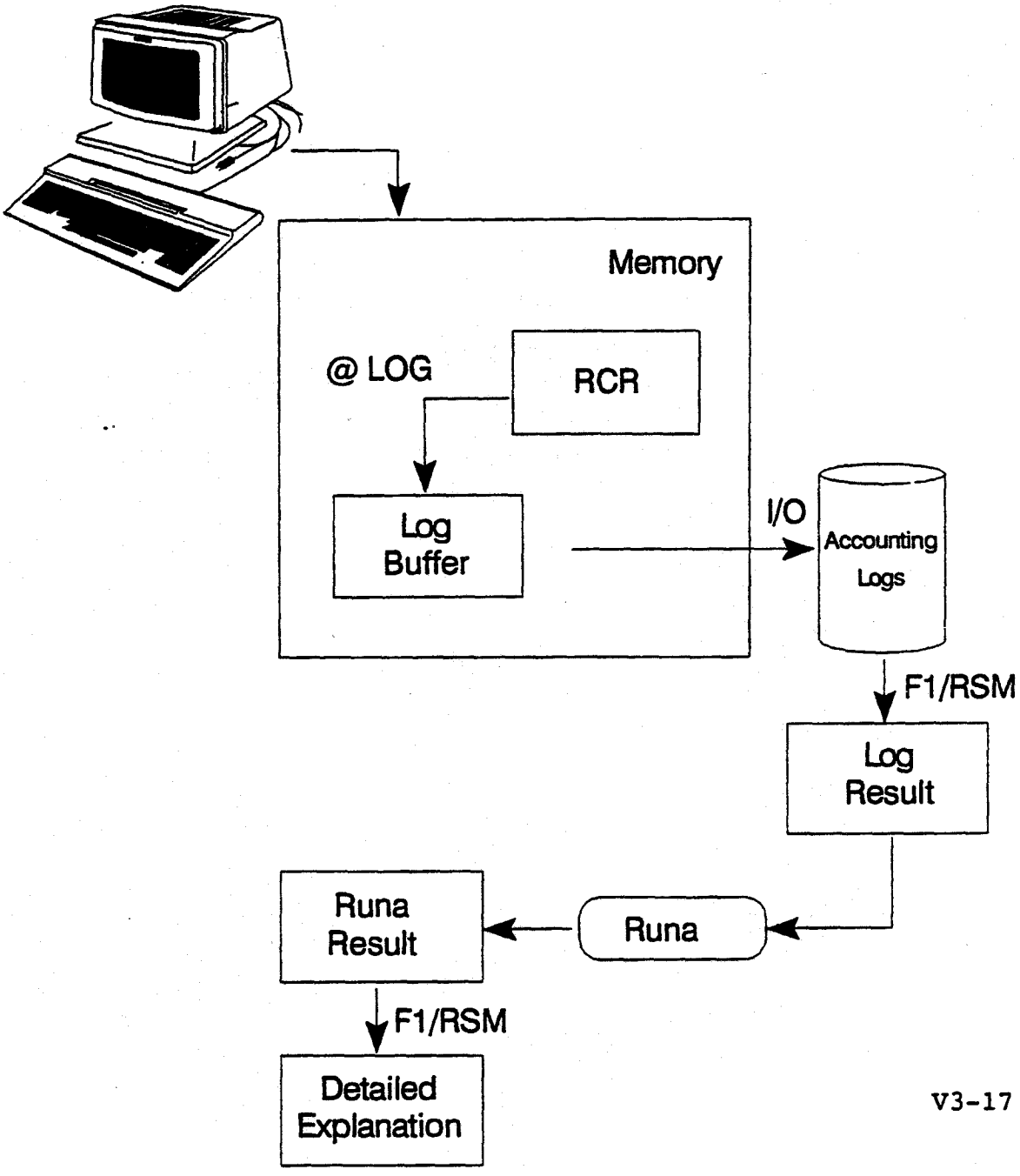
```
.DATE              14:08:35  RID   16A   20 JUL 89  MAPCOORD
*-------------------------------------------------------------------------------
 .LOG LIST: FROM  14:06:31  20 JUL 89  TO  14:06:37  20 JUL 89                B010
 FOR: MAPCOORD
       TRANSACTIONS        IO'S          LOGIC LINES        DATA LINES
       ------------ ---------------- ---------------- ----------------

PART  1 =         5                1               20               460
PART  2 =         2                0                1                16

TOTAL =           8                1               21               478


.
*   USER     .  FUNCTION  .   ACTIVE   .DRAWER.RPT .RD LINES.WR LINES.RD IO.WR IO
*-----------.------------.-----------.------.----.--------.--------.-----.-----
 MAPCOORD     PRE-RUN            0.020    30  104       78        0     1     0
 MAPCOORD     MATCH*             0.080     6   -0      181       60     0     0
 MAPCOORD     CALCULATE*         0.240     6   -0       34       20     0     0
 MAPCOORD     TOTALIZE*          0.060     6   -1       48        6     0     0
 MAPCOORD     RUN                0.040    10    3       20       13     0     0
PART  1 XCTNS =     5  LLP =       20   LINES =       460  I/O'S =      1
 MAPCOORD     DISPLAY            0.040    10   -0       13        0     0     0
 MAPCOORD     RUN                0.020    10    3        1        2     0     0
PART  2 XCTNS =     2  LLP =        1   LINES =        16  I/O'S =      0
 MAPCOORD     DISPLAY            0.000    10   -0        2        0     0     0
                          ..... END REPORT .....
```

V3-16

Runa

- Performs a further analysis of the log listing.
- Attempts to identify inefficient run design techniques.
- Displays log listing.
  - Home cursor.
  - Enter "runa" on control line and transmit.
  - For detailed explanations, F1/RSM again.

Runa


Memory
@ LOG
RCR
Log
Buffer
I/O
Accounting
Logs
F1/RSM
Log
Result
Runa
Runa
Result
F1/RSM
Detailed
Explanation

V3-17

Runa Listing

```
.DATE            14:48:05  RID   18A   20 JUL 89  MAPCOORD
**** U  SERIES   MAPPER   RUN   ANALYZER  *****

                  ANALYSIS OF RUN BIO
              RUN CONTROL RPT 3E IN CABINET PAIR 0/1

                        *** LOG LIST ***
  FUNCTION   .  TIME  .   ACTIVE   .DRAWR. RPT .LINES-IN.LINE-OUT.READS.WRITES
============.========.==========.=====.=====.========.========.=====.======
PRE-RUN       14:47:23      0.020    30   104       78        0     1      0
RUN           14:47:23      0.000    10     3        0        0     0      0
MATCH*        14:47:23      0.080     6    -0      181       60     0      0
CALCULATE*    14:47:23      0.220     6    -0       34       20     0      0
TOTALIZE*     14:47:23      0.060     6    -1       48        6     0      0
RUN           14:47:23      0.040    10     3       20       13     0      0
DISPLAY       14:47:23      0.020    10    -0       13        0     0      0
RUN           14:47:25      0.020    10     3        1        2     0      0
DISPLAY       14:47:25      0.000    10    -0        2        0     0      0

          *** DATA I/O DISTRIBUTION BY FUNCTION ***
      FUNCTION      NUMBER  DATA LINES    I/O'S      % OF I/O'S
      ------------  ------  ----------    ------     ----------
      PRE-RUN          1        78          1          100.00
      CALCULATE*       1        54          0            0.00
      DISPLAY          2        15          0            0.00
      MATCH*           1       241          0            0.00
      TOTALIZE*        1        54          0            0.00

                  *** LINE USE SUMMARY ***
                          LINES          I/O'S
                         -------        -------
     LOGIC LINES            21             0
     DATA LINES            457             1

     TOTALS                478             1

          THE APPROXIMATE COST FOR PROCESSING WAS $0.001
******* SOME GENERAL GUIDELINES FOR GOOD RUN DESIGN HAVE NOT BEEN MET *******

NUMBER
======

*01 THE RATIO OF DATA I/O'S TO LOGIC I/O'S IS  0.00 TO 1.
    (TRY FOR 10 OR GREATER)

******                  RUN REGISTRATION DATA                        ******
          IO'S ALLOWED -  1000      LLP'S ALLOWED -  2000
                      RUN DESIGNER -
                     - CABINETS REGISTERED -
                                0


   MULTI REGISTRATION -    RUN STATUS -

******                  USER REGISTRATION DATA                       ******
USER SIGN ON - XXXXXXXXXX NAME - XXXXXXXXXXXX      DEPT -   104
EXT-LOCATION - XXXXXXXXXX                      STATION NUMBER -     9
```

V3-18a

Runa Listing

```
THIS USER IS A RUN DESIGNER
********************************************************************************
*              G U I D E L I N E   E X P L A N A T I O N S                     *
********************************************************************************

01 LOGIC LINE ANALYSIS - RATIO OF DATA LINES TO LOGIC LINES PROCESSED

 TEST DESCRIPTION:

 THIS TEST COMPARES THE NUMBER OF DATA LINES TO THE NUMBER OF LOGIC LINES
 PROCESSED, THEN CALCULATES THE RATIO.

 PURPOSE:

 TO COMPARE THE NUMBER OF DATA LINES TO LOGIC LINES USED.  THIS RATIO, AS WELL
 AS I/O'S, MEASURE EFFECTIVE AND EFFICIENT RUN DESIGN.  DATA LINES ARE ANY
 LINES PROCESSED BY MAPPER FUNCTIONS (EXCEPT THOSE PROCESSED BY THE RUN
 FUNCTION).  LOGIC LINES ARE ANY LINES READ FROM THE RCR.  IF THE RATIO OF DATA
 LINES TO LOGIC LINES IS LESS THAN ONE, IT MEANS THAT MORE LINES OF LOGIC WERE
 PROCESSED THAN WERE LINES OF DATA AND THAT THE RUN IS NOT TAKING FULL
 ADVANTAGE OF MAPPER FUNCTION CAPABILITIES.

 ACCEPTABLE TEST RESULTS:

 A DATA TO LOGIC LINE RATIO OF 10 TO 1 OR BETTER IS ACCEPTABLE. THIS MEANS
 THAT 10 OR MORE DATA LINES WERE PROCESSED FOR EACH LOGIC LINE.

 POSSIBLE CORRECTIVE ACTION:

 CONSIDER COMPRESSING OR ELIMINATING LOGIC LINES, AND USING MORE BASIC MAPPER
 FUNCTIONS, INSTEAD OF USING @CHG, @IF, @RDL, AND @RLN STATEMENTS.  SHORTEN
 LOGIC LOOPS AND, WHEREVER POSSIBLE, USE MULTIPLE STATEMENTS ON ONE LINE.
================================================================================
***          RESUME (F1) TO EXIT RUNA AND RE-DISPLAY THE LOG LIST           ***
================================================================================
                        ..... END REPORT .....
```

V3-18b

Summary

- The term efficiency means reducing I/Os and LLPs wherever and whenever possible without losing maintainability.
- I/Os perform the retrieval and transfer of data between peripheral devices.
- The reduction of I/Os is important because they are time consuming and costly for the system.
- LLPs are Logic Lines Processed.
- DLPs are Data Lines Processed.
- The Accounting Log contains log summary information for the RCR.
- The Accounting Log file can be accessed in three ways: LOG, LOGLA, and RUNA. These are the tools used to analyze the efficiency of runs.

Exercises

1. Define I/O.

2. Define LLP.

3. Define DLP.

4. What is the purpose of the Accounting Log?

5. Write a run that will match on Product Type and move Product Cost Field. (For this run use a report from drawer B of the JDOE database and one from drawer C).

   - Execute the run.
   - Wait a few seconds and F1 (PF1).
   - ADTO the Log to the RCR.
   - Print it (PR).
   - Repeat the same steps but with LOGLA.
   - Repeat the same steps but with RUNA.

4

Recovery Concerns and Techniques

Module Objectives

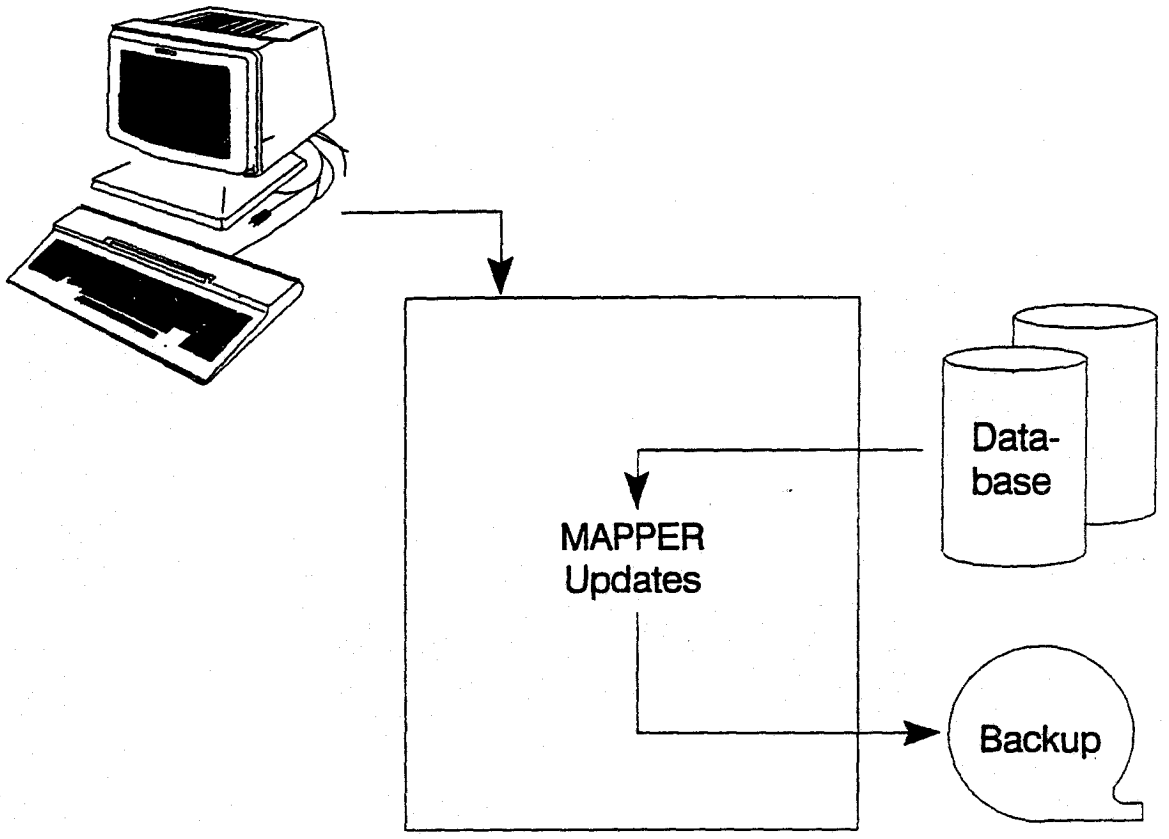
Upon completion of this module, you will be able to:

1. Identify various system recovery schemes.
2. State the differences between purge tapes and duplex files.
3. Define and code the DFU statement.

System Recovery

- Backup Tape
- Duplex Files

System Recovery



V4-1

Recovery Aids

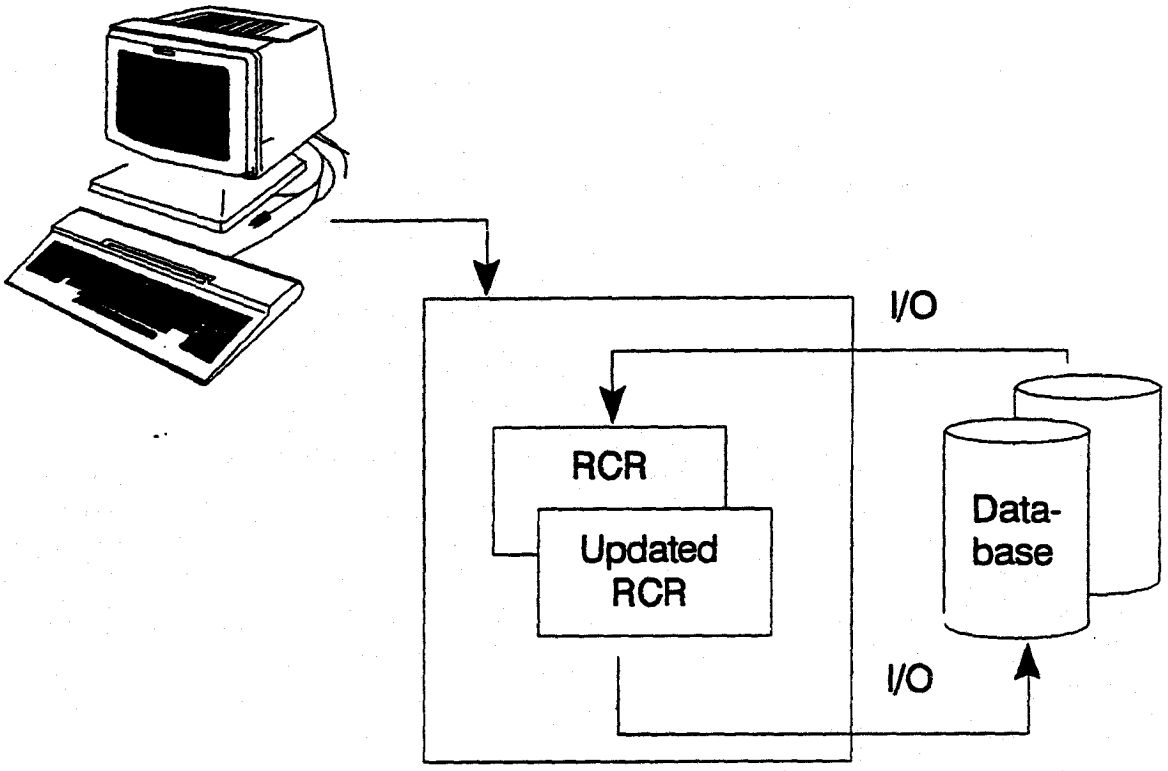
- Minimize updates
- Monitor report size

Update Functions

Include:

- First update of the day.
- Report manipulations - ADD, DUP, REP
- Line manipulations - LN+, LN-, LNX ...
- Update functions - MAU, SRU, CAU ...
- SOE update or Add Line function.

Update Functions



V4-2

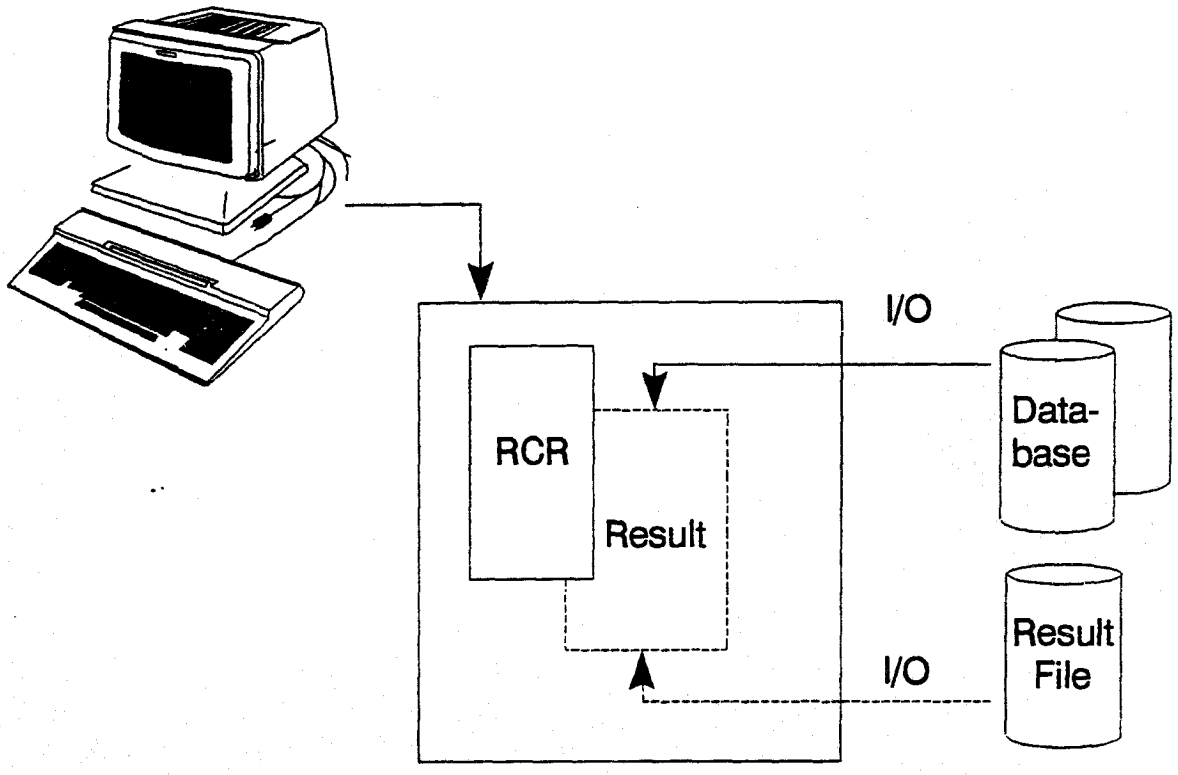
Report Size

- Always lock a report before updating.
- When designing an application, create the drawer to the width required.
- Several smaller reports are more efficient than one large report.

Results

- Process as much as possible in and with results.
- Updates to results are not recorded unless the result is saved as part of the report.
- Utilize RSL (Create Result Copy) statement as often as possible.
- Utilize RNM (Rename) statement as often as possible to take advantage of the eight result areas MAPPER provides (-1 through -8).
- Pivot reports - reports created, manipulated, and then deleted within the run should be avoided, as they can use unnecessary system overhead.

Results



V4-3

Results and RDC

```
*-----------------------------------------------------------
@Brk  Rdl,0,b,2,15  2-79  v131s  . Key line types
Bv131
@Rdl,0,c,2,15  2-79  v131s  .
Cv131
@Rdl,0,d,2,15  2-79  v131s  .
Dv131
@Brk  Rnm -1  Dsp,-1  . Break and Rename the result -1

@Brk,0,b  . Clear out -0
@Rdc,0,e,-1,2,,b,5  2-79  v131s  . Read Continuous -1 for B type lines
' \v131
@Brk  Add,0,b,-0,0,b,3  Dsp,-0 .. Take read data and append

@5:Brk,0,c  . Clear out -0
@Rdc,0,c,-1,2,,c,6  2-79  v131s  . Read Continuous -1 for C type Lines
' \v131
@Brk  Dsp,-0  . Break and Display

@6:Brk,0,d  . Clear out -0
@Lok,0,d,2  .
@Sru,0,d,2  ab(@)  ,@  Rnm  -2  Dsp,-0  . Search update for blank lines
@Rdc,0,e,-1,2,,d,7  2-79  v131s  . Read Continuous -1 for D type lines
' \v131
@Brk  Add,0,d,-0,0,d,-2  Dsp,-0  . Take -1 and -2 data and append
@Upd  Ulk  Dsp,0,d,2,99  . Update , Unlock and display
@7:Gto  end  .
```

V4-4

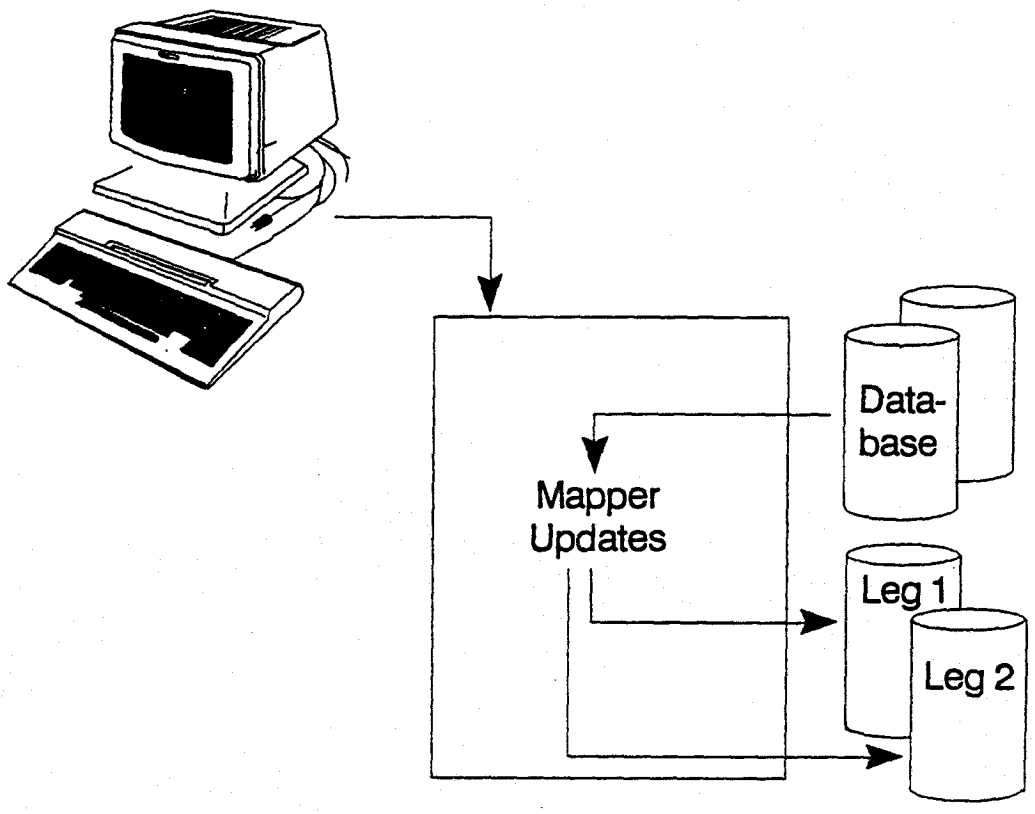
Results and CAL

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
@Sor,0,b,2 " 'CustCode' ,1 .
@Ldv,w v1a1=Tic$,v12i1=1 .
@Cal,-0 " 45-4 ,a if:a=v1amcov1;then:lt=1;if:a=v1arco;\
then:lt=2;if:a=v1dicov1;then:lt=3;if:a=v1fedsv1;then:lt=4;\
if:a=v1intrv1;then:lt=5;if:a=v1usscv1;then:lt=6 .
@Rnm -1 Dsp -1 .
@1:Rdc,0,b,-1,6,,v12 1-80 v10s80 .
v10
@Brk Dsp,-0,,,,y .
@If v12 = 6 gto 2 ; inc v12 gto 1 .
@2.
  stop run
@Gto end .
```

Duplex Files

- For each logical file, two physical files are created; referred to as "legs."
- Introduced to prevent downtime for a system.
- Reflect all data.
- I/Os double!!

Duplex Files

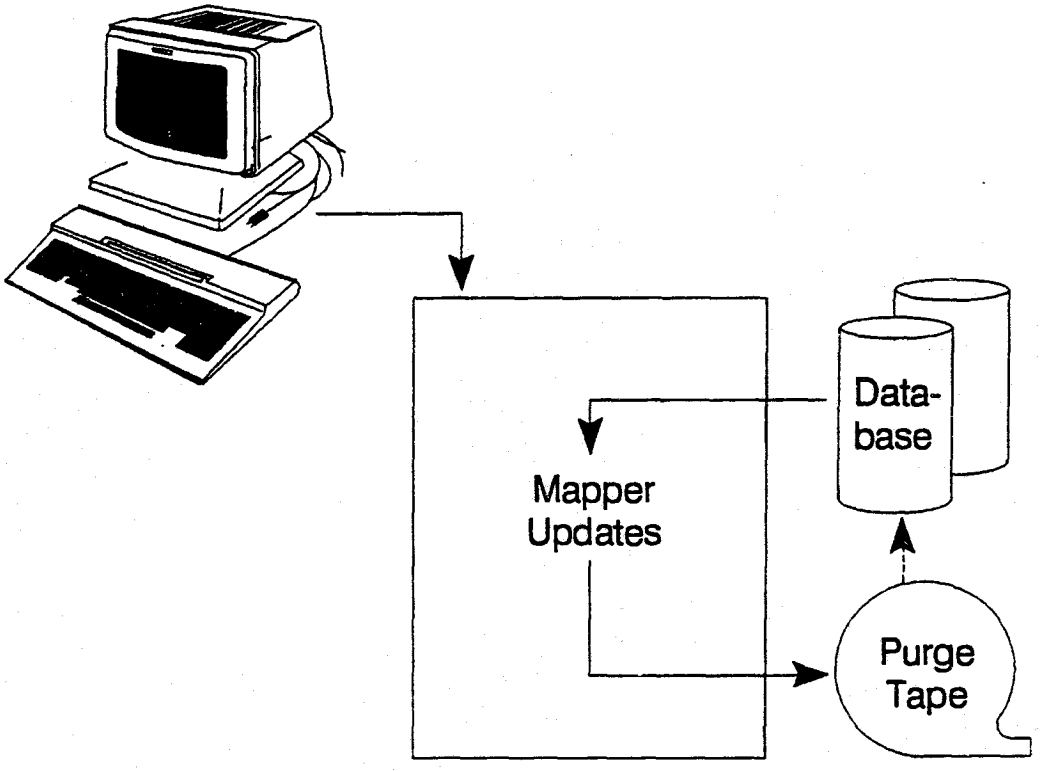


V4-6

System Restoration

- Purge MAPPER database reports to a tape device on a daily basis.
  - Streaming cartridge
  - 9-track tape device
- Restore reports to the database from the latest purge tape.

System Restoration



V4-7

Purges

- **Standard**
- **Incremental**

Standard Purge

- Writes to tape all cabinets and drawers that have been selected.

- Must be performed while MAPPER is active.

Incremental Purge

- Writes to tape only the reports in selected cabinets and drawers that have been updated since the last standard purge.
- After a purge of the entire database, the incremental purge writes only updated reports to the purge tape.
- Reduces time spent backing up database files.
- Must be performed while MAPPER is active.

Run Recovery Aids

- Checkpoint Reports
- Deferred Updates

Checkpoint Reports

- A technique that maintains the status of a run in execution at certain logical points.
- Can be used for run maintenance.
- Paired with the RCR's report number, but resides in another drawer.

@ DFU can do up to 64 reports on 1100.

Checkpoint Reports

```
. date 10 Jul 89   10:16:02          Rid 13E
*=============================================
@45: Task One
   Editing Sort
@Rep

@60: Task Two
   Updates to data
@Upd

@80:  Task Three
   Account Balances
@Rep

@125: Task Four
   Print the data
@Gto end
```

```
. date 10 J   89   10:16:04          Rid 13F
* Task      Start      Start      Recv      Rids
* Number .  Date   .   Time  .    Labl .   Beg  .End.  Mod  .Type.
*=============================================================
    1       881114     093614      45      18
    2       881114     094834      60      18
    3       881114     100502      80      140   143
    4       881114     102042     125      60


. STATUS: (In Process, Completed)   Completed, 11 Nov 88, 102549
. Keys for Recovery:
```

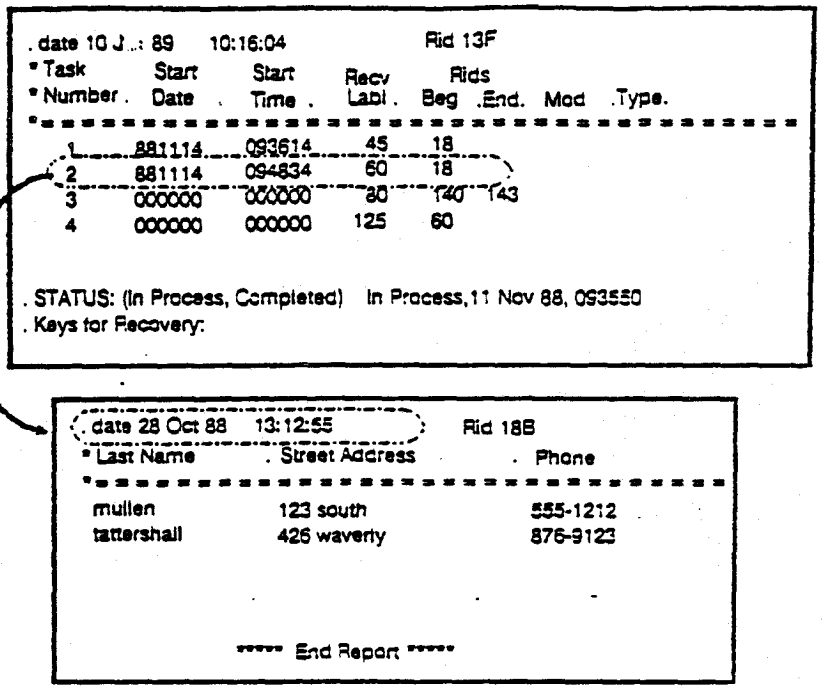
V4-8

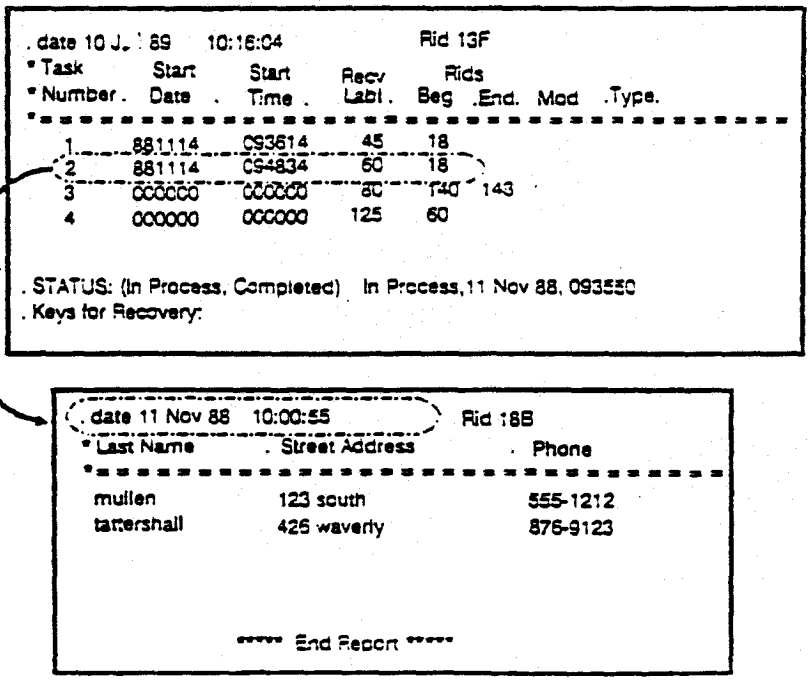
Checkpoint Report Examples

```
. date 10 J... 89    10:16:04                    Rid 13F
* Task          Start          Start          Recv          Rids
* Number .      Date    .      Time .         Labl .    Beg   .End.   Mod    .Type.
*=========================================================================
    1          000000         000000          45        18
    2          000000         000000          60        18
    3          000000         000000          80        140   143
    4          000000         000000         125        60

. STATUS: (In Process, Completed)    (status), (date), (time)
. Keys for Recovery:
```

V4-9

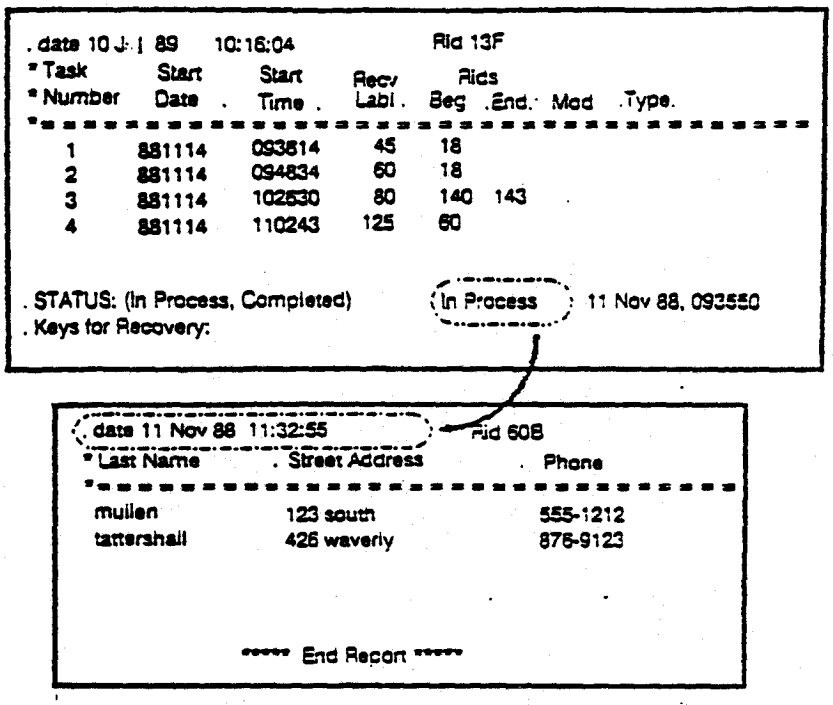
Checkpoint Report Examples



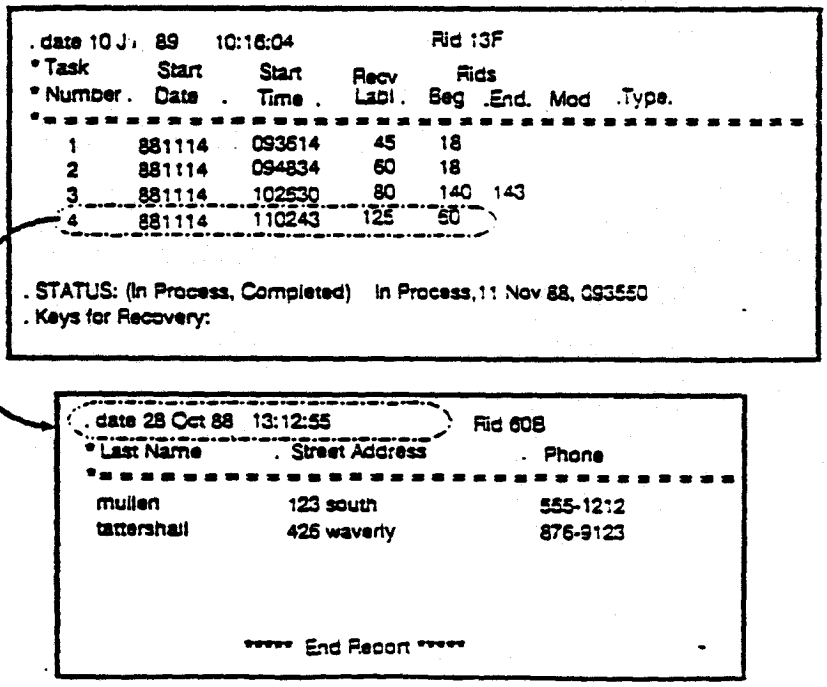


V4-10

Checkpoint Report Examples



versus



V4-11

Deferred Updates

- Adds update security against abnormal run termination.
- Defer Update (DFU) statement defers updates to reports until a Commit Update (CMU) statement is encountered.
- Update lock is placed on the deferred reports while DFU is executed.
- Up to ~~five~~ 10,64 reports may be deferred at the same time.

Deferred Updates

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
@Dfu,10  0,b,2,0,c,1   . defer updates
@Sru,0,b,2,6,,99  dh  15-9  ,greenbox1  del .
@Sru,0,c,1,6,,99  dh  2-9  ,greenbox1  del .
@Cmu . commit updates
@10 .
    Both reports are not available
@Gto end .
```

V4-12

DFU Statement

Description

The Defer Update (DFU) statement defers updates to reports until a CMU (Commit Update) is executed.

Format

```
@DFU[,lab] c,d,r[,c,d,r,...c,d,r] .
```

Fields

Field	Description
lab	Label to go to if an update lock cannot be granted for any of the specified reports.
c,d,r, ...c,d,r	Cabinet numbers, drawer letters, and report numbers of the reports on which to defer updates (up to five reports).

Example

```
@     dfu 0,c,4 .
@     sru,0,c,4,,099 dh 20-9],greenbox1 ext .
      .
      .
      other processing
      .
      .
@     cmu dsp,-0 .
@     rel .
@099:dcu .
```

Defer the update on report 4C, cabinet 0. An SRU statement is executed and the results extracted from the report. After additional processing, the CMU statement is executed, and the result is displayed. If the report did not contain any "greenbox1" entries, the run continues at label 99 and decommits the updates.

DFU Statement

The following run statements cannot be executed following a DFU until a CMU or DCU statement is executed:

DFU	Set defer update on a report or reports
DSG	Display graphics
DSM	Display a message
DSP	Display a report or result
LOK	Place an update lock on a report
ITV	Accept input from OTV output and initialize variables
OUM	Display blank function mask on the screen
OUT	Place data on the screen
REL	Release the run
RTN	Return to a remote MAPPER site
RUN	Execute another MAPPER run
SC	Create input screens or edit displayed data
WAT	Stall the run
XIT	Sign user off MAPPER
XUN	Exit MAPPER

Summary

- Any time an update occurs on a report, the updated report is written to disk.
- These additional I/Os slow response time, but run designers can exert some control over these additional I/Os by minimizing the number of updates and monitoring report size.
- The best solution is to work with results whenever and wherever possible.
- Duplex files, which contain all data, can be a means of recovery.
- Restoration of the database is performed by the MAPPER initialize process using purge tapes.
- Two techniques have been presented for run designers to utilize for run recovery; Checkpoint Reports and Deferred Updates. Weigh processing requirements in order to determine which technique should be used.

Exercises

1. Name the two methods that can be considered recovery techniques for U Series MAPPER, and indicate which is recommended.

2. Name two run recovery techniques.

3. What is the difference between purge tapes and duplex files?

4. Duplicate report 3B of the JDOE database.

   a. Write a run to perform a number of updates to the report. Defer the updates using the @DFU statement. Execute the run, and obtain the RUNA result.

   b. Repeat step 1a, deferring updates by locking the report, turning the report into a result, and making updates to the result. Then replace the result back into the original report. Execute the run, and obtain a RUNA result. How do these two methods compare?

5

Advanced Variable Concepts

Module Objectives

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

1. Utilize variable stack commands.
2. Utilize the BVT and VARIABLE runs.

Variable Review

- Must be named, defined, and initialized.
- V1-V199 per run.
- Definition consists of type and size.

Naming Variables
Initialization

```
*=====================================================
@Ldv     v1i3,v2i1,v3i3 .
@Ldv,w   v4a12=Date$,v5i8=Time$,v8a1=Soe$ .

         .
         .
         .
```

VERSUS

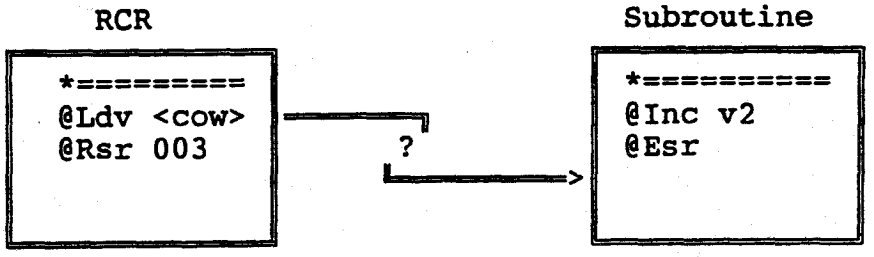
```
*=====================================================
@Ldv     <cab>i3,<drawer>a1,<report>i3 .
@Ldv,w   <date>a12=Date$,<time>i8=Time$,<soe>a1=Soe$ .

         .
         .
         .
```

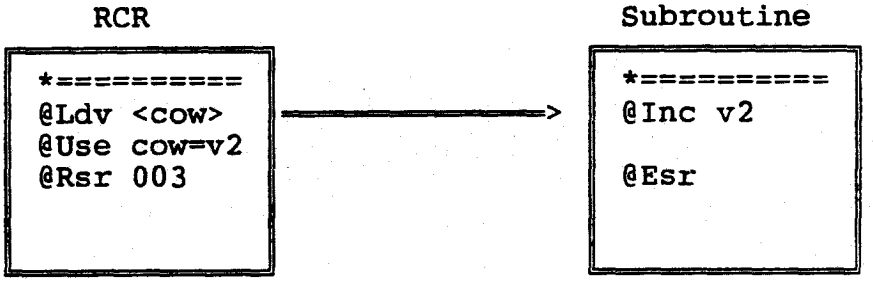
V5-1

Named Variables

- Introduced with 3R1.
- No more than 12 alphanumberic characters.
- First character must be alphabetic.
- Enclose in "greater-than less-than" symbols.

Assigning Variable Names to Variable Numbers
USE Statement





V5-2

USE Statement

Description

The USE Variable Name (USE) statement assigns a variable name to a specific variable number.

Format

```
@USE  name1=v1[,name2=v2,...,namen=vn] .
```

Fields

Field	Description
name1...namen	Name or names to assign.
v1...vn	Variable number or numbers to which the names are assigned. I want you to initialize the variable, include the type and size (such as v2h12).

Examples

```
@use  cabinet=v110i3 .
```

Assigns the name <cabinet> to v110.

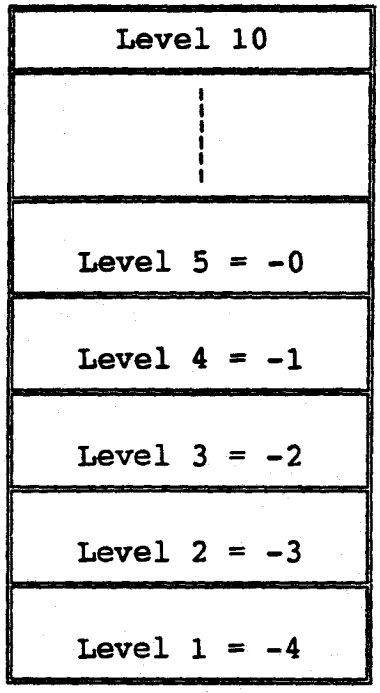
```
@use  name=v120s30 .
```

Assigns the name <name> to v120.

Variable Tables

- Two exist per RCR.
  - Numeric variables stored in sorted order.
  - Named variables loaded as they appear.
- Referenced by internal MAPPER naming scheme.

Variable Stacks



A-38

V5-3

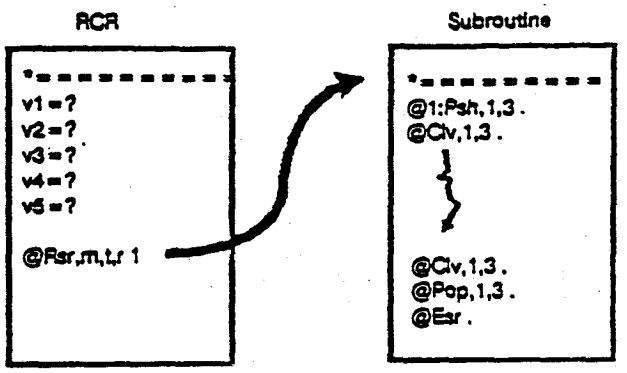
Variable Stacks

- Method used to increase the standard 199 variable limit.
- Buffer in memory.
- Ten levels per stack.
- Relative numbering scheme (-0 to -10).
- STACK$ holds current level number.

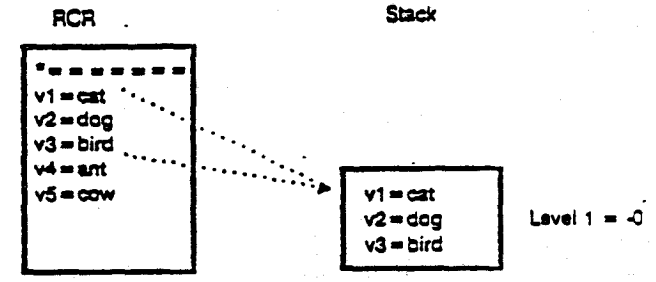
Stack Commands

- PSH (Push Variables)
- CLV (Clear Variables)
- POP (Pop Variables)
- PEK (Peek Variables)
- POK (Poke Variables)
- RMV (Remove Variables)
- XCH (Exchange Variables)

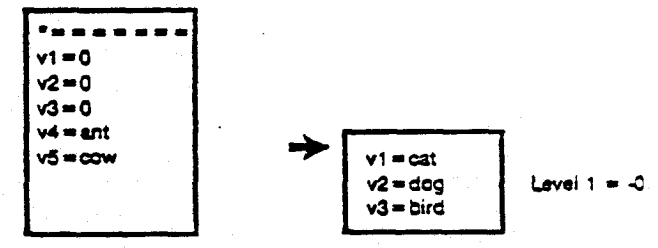
PSH, CLV, and POP



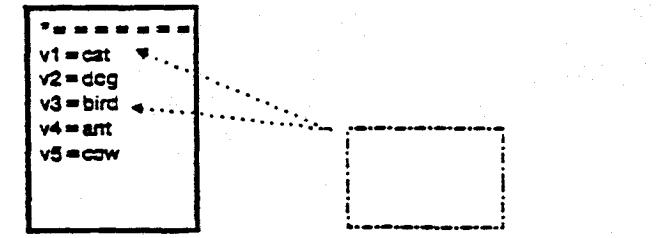
@PSH,1,3



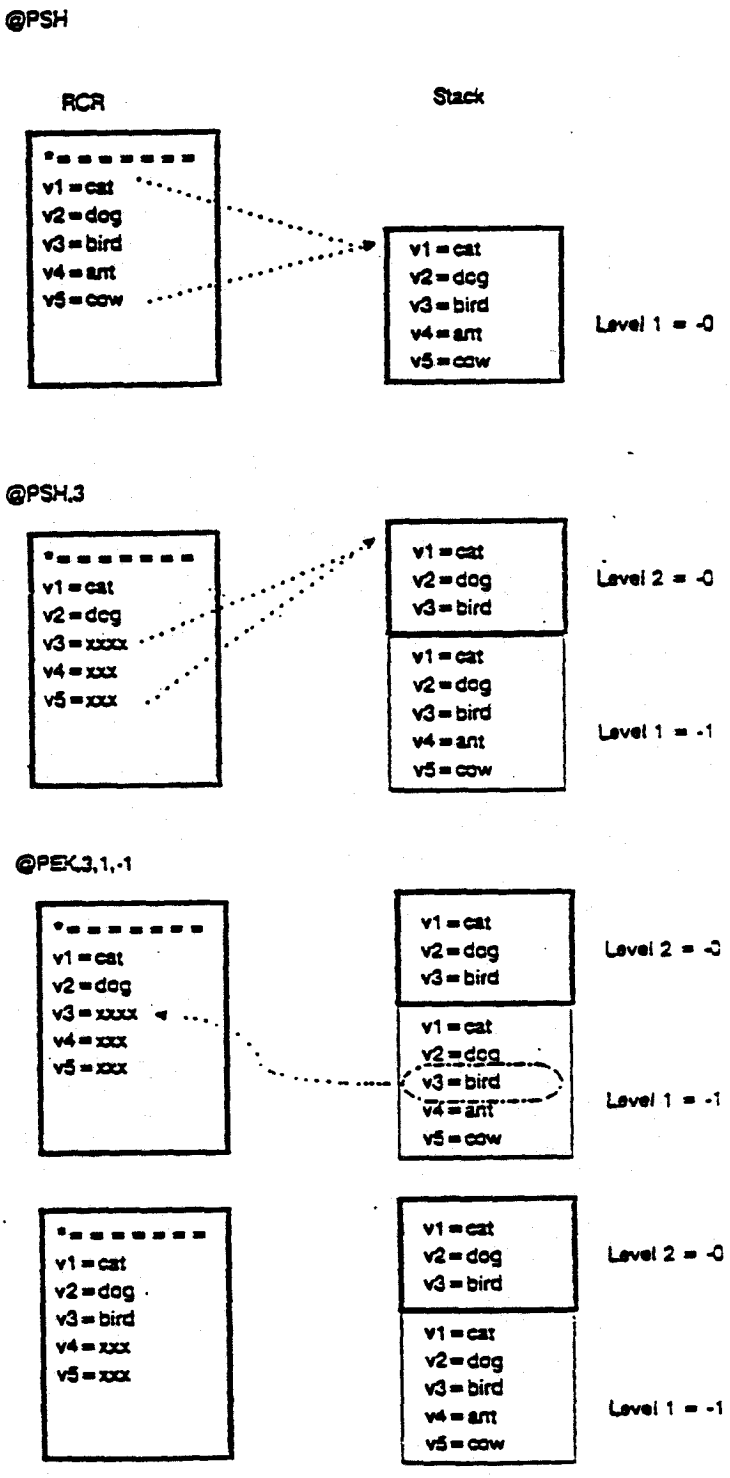
@CLV,1,3



@POP,1,3

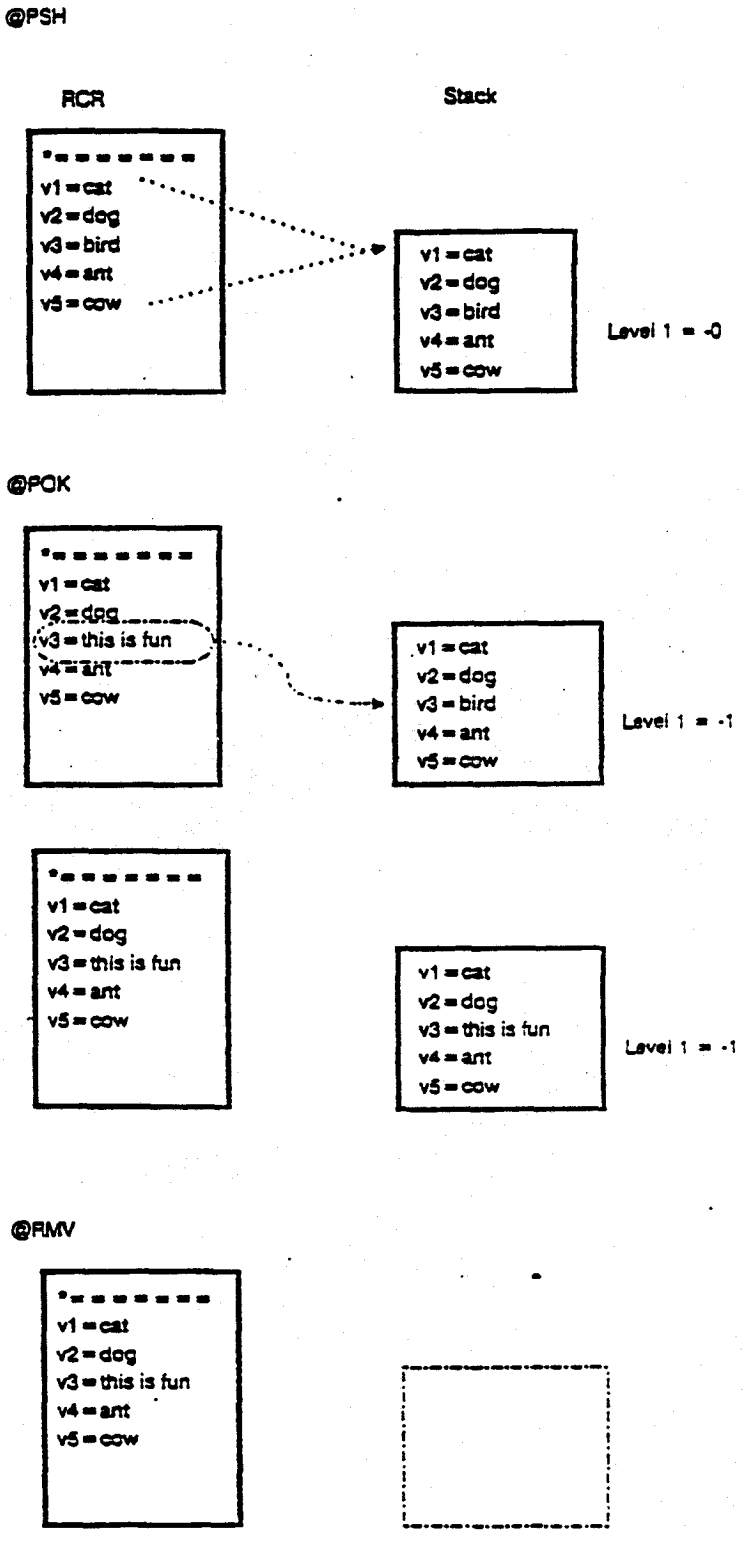


PEK


@PSH
RCR
Stack
v1 = cat
v2 = dog
v3 = bird
v4 = ant
v5 = cow
Level 1 = -0
@PSH,3
v3 = xxxx
v4 = xxx
v5 = xxx
Level 2 = -0
Level 1 = -1
@PEK,3,1,-1

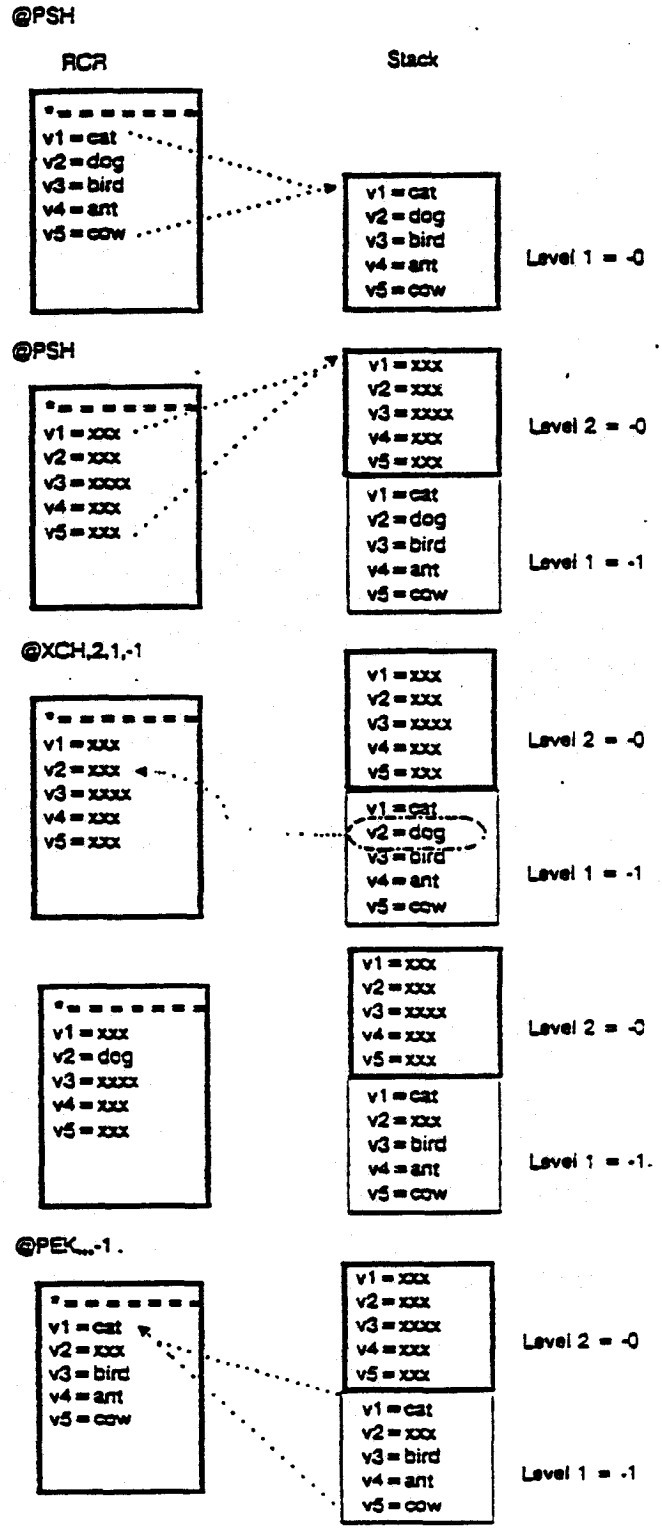
V5-5

POK and RMV



V5-6

XCH


@PSH
RCR
Stack
v1 = cat
v2 = dog
v3 = bird
v4 = ant
v5 = cow
Level 1 = -0
@PSH
v1 = xxx
v2 = xxx
v3 = xxxx
v4 = xxx
v5 = xxx
Level 2 = -0
Level 1 = -1
@XCH,2,1,-1
@PEK,,-1.

V5-7

PSH

Description

The Push Variables (PSH) statement saves both the contents and definition of variables so that you can restore them later in the run with a POP or a PEK statement.

Format

```
@PSH[,stvno,q,slv] .
```

Fields

Field	Description
stvno	Starting variable number to save. Default=1.
q	Number of variables to save. This includes existing and nonexisting variables. Default=all.
slv	Previously saved level (set) of variables to replace (a number from -1 to -10). Default=new stack item, where: -1 is the most recent level. -2 is the second most recent level .... etc.

Examples

```
@psh,1,10 .
```

Saves variables v1 through v10.

```
@psh .
```

Saves all variables.

CLV

Description

The Clear Variables (CLV) statement clears the contents and definition of a range of variables. Typically, CLV may be used to release string variable space.

Format

```
@CLV[,stvno,q] .
```

Fields

Field	Description
stvno	Starting variable number to clear. (q is assumed if stvno is not specified).
q	Number of variables to clear (all variables are cleared if this field is not specified).

Examples

```
@clv,1,10 .
```

Clears variables v1 through v10.

```
@clv .
```

Clears all variables.

POP

Description

The Pop Variables (POP) statement restores variables saved with a previous PSH statement and removes their level from the stack.

Format

```
@POP[,stvno,q,slv] .
```

Fields

Field	Description
stvno	Starting variable number to restore. Default=first variable originally saved.
q	Number of variables to restore. This includes existing and nonexisting variables. Default=number of variables originally saved.
slv	Previously saved level (set) of variables to restore (a number from -0 to -9). Default=most recent level, where: -1 is the second most recent. -2 is the third most recent. etc.

Examples

```
@pop .
```

Restores all variables saved with the previous PSH statement.

A stack exists where the fourth most recent level contains variables v3 through v20:

```
@pop,5,10,-3 .
```

Restores variables v5 through v14 (with v3, v4, and v15 through v20 remaining unchanged). The fourth most recent level is removed and the stack condenses so the fifth most recent level becomes the fourth.

PEK

Description

The Peek Variables (PEK) statement restores variables saved with a previous PSH statement.

Format

```
@PEK[,stvno,q,slv] .
```

Fields

Field	Description
stvno	Starting variable number to restore. Default=first variable originally saved.
q	Number of variables to restore. This includes existing and nonexisting variables. Default=number of variables originally saved.
slv	Previously saved level (set) of variables to restore (a number from -0 to -9). Default=most recent level, where: -1 is the second most recent. -2 is the third most recent. etc.

Examples

```
@pek .
```

Restores all variables saved on a previous PSH statement.

```
@pek,1,3 .
```

Restores variable v1 through v3.

```
@psh,1,18 .
@pek,5,10 .
```

PSH saves variables v1 through v18.
PEK restores v5 through v14 (with v1 through v4 and v15 through v18 remaining unchanged).

POK

Description

The Poke Variables (POK) statement replaces the contents and definition of variables saved with a previous PSH statement.

Format

```
@POK[,stvno,q,slv] .
```

Fields

Field	Description
stvno	Starting variable number to replace. Default=1.
q	Number of variables to replace. This includes existing and nonexisting variables. Default=all.
slv	Previously saved level (set) of variables to replace (a number from -0 to -9). Default=most recent level, where: -1 is the second most recent. -2 is the third most recent. etc.

Examples

```
@pok,1,3.
```

Restores the saved contents and definition of variables v1 through v3, which were saved on the previous PSH statement, with their current values.

A stack exists where the third most recent level contains variables v1 through v20:

```
@pok,5,10,-2 .
```

Replaces the saved contents and definition of variables v5 through v14 (with variables v1 through v4 and v15 through v20 remaining unchanged).

RMV

Description

The Remove Variables (RMV) statement removes variables saved with a previous PSH statement.

Format

```
@RMV[,slv] .
```

Fields

Field	Description
slv	Previously saved level (set) of variables to remove (a number from -0 to -9). Default=most recent level, where: -1 is the second most recent. -2 is the third most recent. etc.

Examples

```
@rmv .
```

Removes variables saved on the previous PSH statement.

A stack exists that contains five levels:

```
@rmv,-2 .
```

Removes the third most recent level. This condenses the stack; the fourth level becomes the third, and the fifth level becomes the fourth.

XCH

Description

The Exchange Variables (XCH) statement exchanges variables saved on a previous PSH statement with current variables.

Format

```
@XCH[,stvno,q,slv] .
```

Fields

Field	Description
stvno	Starting variable number to exchange. Default=1.
q	Number of variables to exchange. This includes existing and nonexisting variables. Default=all.
slv	Previously saved level (set) of variables to exchange (a number from -0 to -9). Default=recent level, where: -1 is the second most recent. -2 is the third most recent. etc.

Example

```
@xch,1,3 .
```

Exchanges the saved size, type, and contents of variables v1 through v3, saved on the previous PSH statement, with their current values.

Choosing Between Named and Numbered Variables

- To use numbered variables:
  - More efficient for complex runs.
  - Can be used in existing runs that use numbered variables for consistency.
  - Run statements are shorter.
  - Stack commands can be used.
- To use named variables:
  - Run statements are easier to understand.
  - The run is more self-documenting.

Variable Aids

- **BVT (Build Variable Table) Run**
- **VARIABLE Run**

BVT

- Builds a table that displays the names (numbered or named) and location of all variables in the RCR.
- Displayed at the end of the RCR as a result.
- Can be used to convert named variables to numbered variables and visa versa.

Sample BVT Run

Before BVT:

```
.DATE              13:31:48  RID   18E   27 JUN 89  JDOE
*RUN FUNCTION DATA:  a12801                          BY:  joanne stewart  E0010
*=============================================================================
:L1=10
@LOG .
@LDV <COUNTER>I3=0 .
@LDV,W <INPUT>I3=INPUT$ .
@BRK .
@INC <COUNTER> .
@001 .
<COUNTER> BOTTLES ON THE WALL.
@BRK .
@OUT,-0,2,1,<COUNTER> .
@IF <COUNTER> = <INPUT> GTO END ; .
@INC <COUNTER> .
@GTO 001 .
```

After BVT:

```
.DATE              13:31:48  RID   18E   27 JUN 89  JDOE
*RUN FUNCTION DATA:  a12801                          BY:  joanne stewart  E0010
*=============================================================================
:L1=10
@LOG .
@LDV <COUNTER>I3=0 .
@LDV,W <INPUT>I3=INPUT$ .
@BRK .
@INC <COUNTER> .
@001 .
<COUNTER> BOTTLES ON THE WALL.
@BRK .
@OUT,-0,2,1,<COUNTER> .
@IF <COUNTER> = <INPUT> GTO END ; .
@INC <COUNTER> .
@GTO 001 .


.VARIABLE TABLE
*    Name     .Vnum.Sq.          Line  Numbers            .   Comment
*============.====.==.=================================.=================
 COUNTER      V001 01 6*,9,11,13,14,15
 INPUT        V002 01 7*,14
                      ..... END REPORT .....
```

V5-8

BVT

Description

The BVT (Build Variable Table) run is used to build a table that displays the location of all the variables in a specified run control report.

Format

Display the RCR and enter one of the following requests on the control line:

BVT	Builds a variable table at the end of the RCR and displays it as a result. Save the result to keep the table.
CVT	(Convert Variable Table) converts numbered variables (such as v1) to named variables (such as <name>) using a previously built variable table saved at the end of the RCR.
CVT,N	(Convert from Named Variables) converts all named variables to numbered variables.

VARIABLE Run

- Tests contents of variables.
- Determines how variable types and input methods affect the contents of a variable.
- Displayed as a result.

Sample VARIABLE Run

```
This run shows what the contents of the non-string variables
look like after initialization and after some operations.
     Enter any initial value -> .              ,


VALUE IN VARIABLE WHEN INITIAL ENTRY IS <20>
 @CHG INVAR$ V4     . @CHG INPUT$ V1     . @CHG V2 V1 +1
 ================== . ================== . ================
 V4A6    = <20     > . V1A6    = <20     > . V2A6    = <     21>
 V4I6    = <     20> . V1I6    = <     20> . V2I6    = <     21>
 V4F6.3 = <20.000> . V1F6.3 = <20.000> . V2F6.3 = <21.000>
 V4H6    = <20     > . V1H6    = <20     > . V2H6    = <20    !>
 ** YOU MAY USE A RESUME (RSM OR F1) FOR ANOTHER VALUE **
                  ..... END REPORT .....
```

Summary

- Variables can be identified by traditional numbering methods or by descriptive names.
- Numbered variables are more efficient.
- The USE statement can equate named and numbered variables.
- Every RCR has two variable tables:
  - one for numbered variables.
  - one for named variables.
- Variable stacks
  - permit more than 199 variables to be used per run.
  - consist of up to 10 levels.
- STACK$ is a reserve word that holds the current level number.
- Stack commands manipulate variables on stacks:

  PSH - saves contents and definitions on a level.
  CLV - clears contents and definitions.
  POP - restores PSH variables and removes a level.
  PEK - restores PSH variables.
  POK - replaces contents and definitions saved with PSH.
  RMV - removes variables saved with PSH.
  XCH - exchanges variables saved with PSH.

- BVT (Build Variable Table) run - builds table that displays variable locations.
- VARIABLE run - test contents of variables.

Exercises

1. Write a run to perform the following:

   a. Initialize at least ten variables using multiple @CHG statements at the beginning of the run control report.

   b. Reuse variables by saving their contents in a data report.

   c. Use a series of @IF/GTO statements to locate a specific character within one of the variables you defined previously.

2. Write a second run to perform the following:

   a. Initialize the same variables in exercise 1-a as needed, with the @LDV statement instead of the @CHG statement.

   b. Reuse variables by saving them in a stack; then, when finished, restore the stack levels to the run control report.

   c. Use the @LCV statement to locate a specific character within one of the variables you defined previously.

3. Include the @LOG statment as the first statement of each run. Execute each run individually; execute RUNA against each of the @LOG results. Compare the results to see if the inefficiencies of the first run were remedied by the second run.

4. Utilize the BVT and VARIABLE runs against your RCR. Print out the results.

6

Efficient Run Structure

Module Objectives

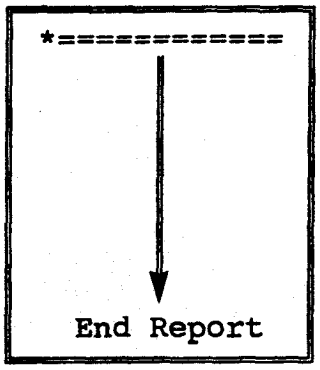
Upon completion of this module, you will be able to:

1. Build a Label Table for a run and understand its efficiencies.
2. Recognize various branching and subroutine methods.
3. Understand the usage of error routines vs. abort routines.

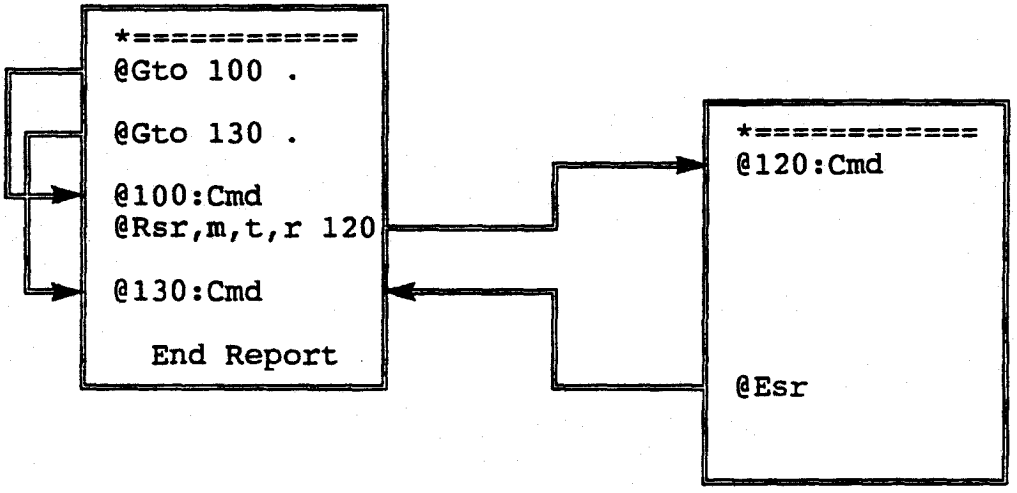
Modular Run Structure

- Labeling
- Looping
- Branching
- Subroutines

Modular Run Structure



Versus



V6-1

Labels

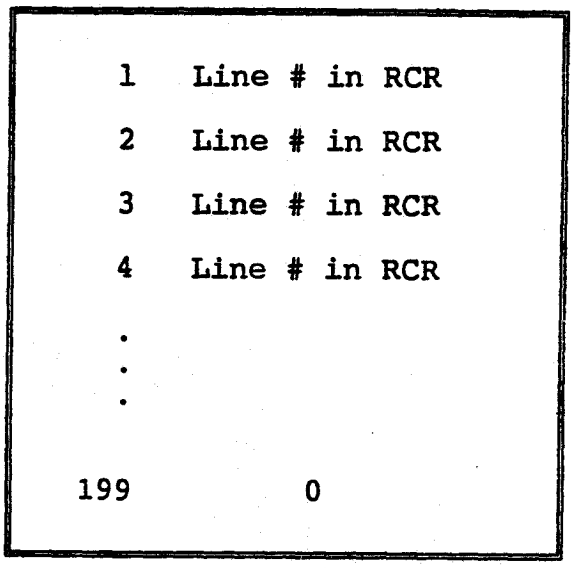
- Uniquely identify any statement line.
- MAPPER searches each line in a report for a label when it is referenced.
- Label tables can be developed to speed this process and make the run more efficient.

Label Table

- Similar to Variable Table.
- One may exist per RCR.
- Used by MAPPER to associate line numbers with labels.
- Built by BLT function or BLT statement.
- Fixed in size system-wide - 199 entries.

Label Table

Memory



V6-2

Label Definition Lines

- Runs that branch to labels are more efficient if labels are defined in the run.
- Predefines label locations; tell the system which lines have labels.
- Are be placed at the beginning of the RCR.

```
:L10=15,20=20,30=120
```

Label Definition Lines

```
*=============================================================
:L1=10
@log .
@ldv <counter>i3=0 .
@ldv,w <input>i3=INPUT$ .
@brk .
@inc <counter> .
@001 .
<counter> bottles on the wall .
@brk .
@out,-0,2,1,<counter> .
@if <counter> = <input> gto end ; .
@inc <counter> .
@gto 001 .
```

V6-3

Building Label Tables

- Use the BLT (Build Label Table) function or statement.
- Adds label table definition lines.
- Whenever a line is added or deleted, BLT should be used again to recalculate labels.
- If label definition lines exist, issuing BLT again will replace them with new ones.
- Use only on runs that have been debugged and tested, and on runs that do not change frequently.

BLT Function

- RCR must be on display.
- Enter BLT on control line.
- Label definition lines are entered automatically in the RCR before the first line that contains "@" in column one.
- Save the result!

BLT Example

```
.DATE   23 SEP 88  09:57:39  RID   14E   22 JUN 88  MICRO
.@991231 REV. 03.055W  880622   'EDIT' SAMPLE SCREEN EDIT                E0010
*=============================================================================
:L1=6,2=14,3=17,4=34,5=35
@BRK .
@001 .                                              RE-ENTRY POINT

          THIS RUN EDITS ORDER ENTRIES IN DRAWER B REPORTS

          ENTER REPORT NUMBER->    ,  (3 OR GREATER)
@BRK OUT,-0,2,23,1,1,,,,P .                         SOLICIT REPORT NUMBER
@CHG INPUT$ <RID>I4 IF <RID> > 2 GTO 002 ; .        BLOCK REPORTS 1 AND 2

          SORRY BUT EDITING OF REPORT <RID> IS NOT ALLOWED.
@GTO 001 .                                          ADD ERROR MESSAGE
@002:RDL,0,B,<RID>,,001 'ST CD' .
@LOK,0,B,<RID> .                                    LOCK THE REPORT
@LDV <LINE>I4=6 .                                   START AT LINE 6
@003:FDR,0,B,<RID>,<LINE>,,005 '' 'ST CD'  ,OR  ,<LINE> .
@RLN 'STATUS','PRODUCT','ORDER','CUST' <STATUS>I,<PROD>H,<ORDER>I,<CUST>H .

          LINE <LINE> IN REPORT <RID> HAS THE FOLLOWING ENTRIES
          CHANGE AS REQUIRED AND PRESS ENTER
          IF NO CHANGES ARE NEEDED, PRESS PF1

          STATUS DATE <STATUS>,
          PRODUCT TYPE <PROD>,
          ORDER NUMBER <ORDER>,
          CUSTOMER CODE <CUST>,
@CHG INVAR$ <STATUS>,<PROD>,<ORDER>,<CUST> .
@BRK OUT,-0,2,23,1,1,,,,P .                         OUTPUT EDITING SCREEN
@IF CURV$ = 0 GTO 004 .                             SKIP UPDATE IF PF1
@WRL,0,B,<RID>,<LINE> 'STATUS','BY','PRODUCT','ORDER','CUST' \
 ,<STATUS>,USER$(1-2),<PROD>,<ORDER>,<CUST> .       UPDATE THE LINE
@004:INC <LINE> GTO 003 .                           FIND MORE 'OR' ITEMS
@005 .                                              COMPLETION MESSAGE
          NO (MORE) ORDER ENTRIES TO EDIT IN REPORT <RID>
@    GTO END .
                        ..... END REPORT .....
```

V6-4

BLT Statement

Description

The Build Label Table (BLT) statement builds label tables in a RCR and creates a result.

Format

```
@BLT,c,d,r[,lab] .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report of the RCR for which to build label tables.
lab	Label to go to if an error condition exists (such as a duplicate or invalid label, or nonexistent report).

Example

```
@blt,0,c,4,010 .
```

Build a label table in RID 4C, cabinet 0, and create a result (go to label 10 in case of an error).

Clearing Label Tables

- Use the CLT (Clear Label Table) function or statement.
- Deletes label table definition lines from RCR and creates a result.
- If any changes are made to a line containing a label in an RCR that contains a label table, the existing label table must be cleared before the updated run is executed.

CLT Function

- RCR must be on display.
- Enter CLT on the control line; the result is displayed.
- Save the result!

CLT Statement

Description

The Clear Label Table (CLT) statement clears label tables in a RCR and creates a result.

Format

```
@CLT,c,d,r[,lab] .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report of the RCR in which to clear label tables.
lab	Label to go to if an error condition exists.

Example

```
@clt,0,c,4,010 .
```

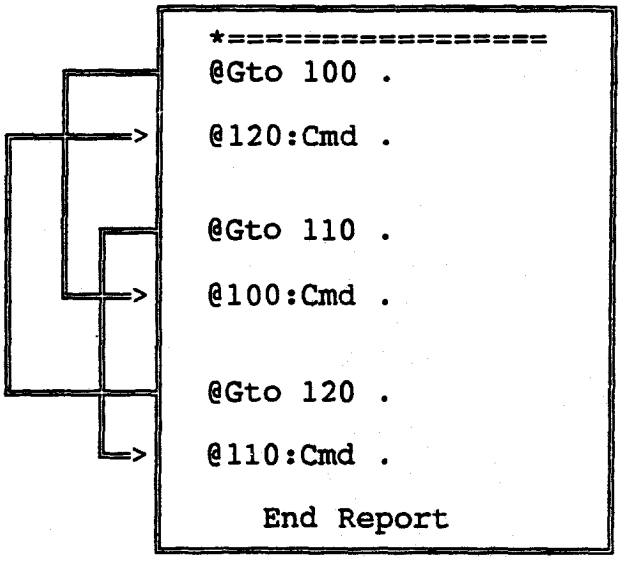
Clear a label table in RID 4C, cabinet 0, and create a result (go to label 10 in case of an error).

Branching
GTO

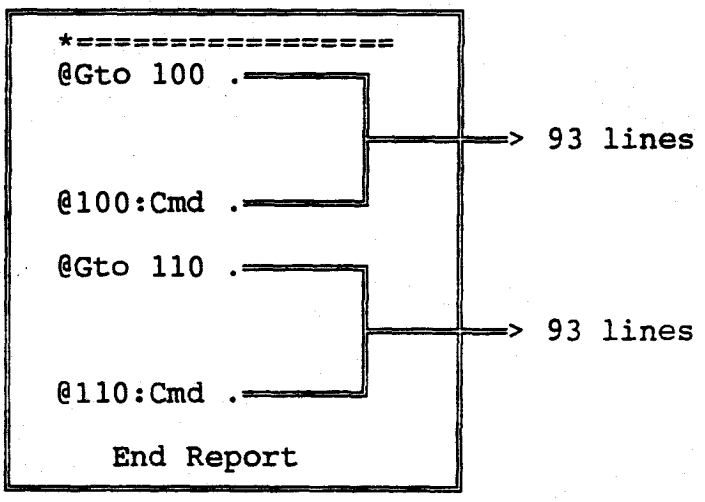
- Simpliest way to loop or branch within a run.

- Use cautiously to confine looping code within 93 line segments to improve efficiency.

GTO Statements



Versus



V6-5

GTO Tips

- GTO LIN+ and GTO LIN- may be be used instead of labels, but are not more efficient.

- Use computational GTOs (IF/GTO) whenever possible.

- The next line of the RCR should always be used in a conditional.

Computational GTO

```
*================================
@If v1 = 1 Gto 100 .
@If v1 = 2 Gto 200 .
@If v1 = 5 Gto 250 .

@If v1 = 1,2 Gto 10 .
@If v1 = 5,8,9 Gto 20 .
```

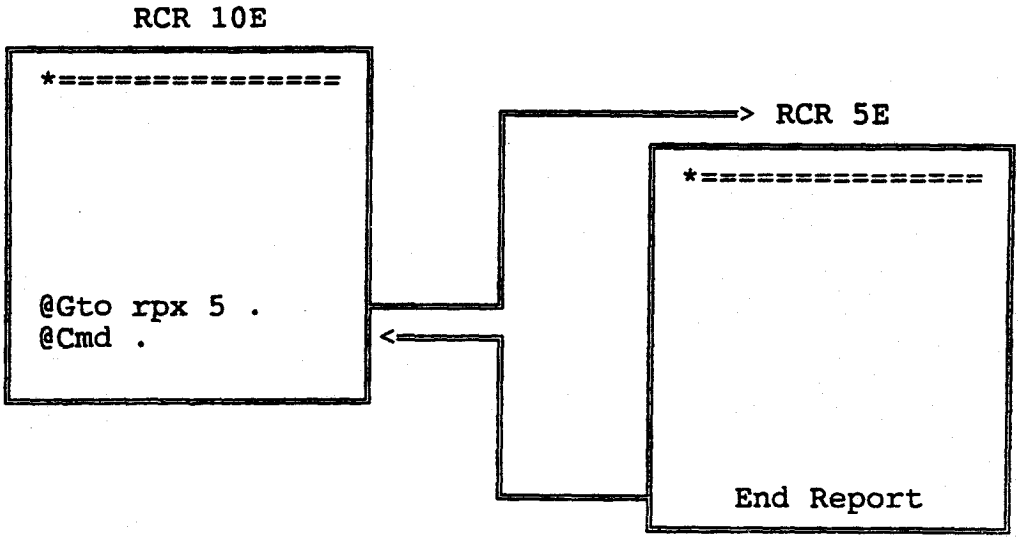
Versus

```
*================================
@If v1 = 1,(100),2,(200),5,(250) .
@If v1 = 1,2,(10),5,8,9,(20) .

@10:Cmd .
@If v1 > 5 Gto 10;Gto 50 .

@50:Cmd
```

GTO RPX

- Permits jumping out of the RCR to another report.
- The referenced report must be in the same cabinet and drawer as the calling report and doesn't need to be registered.
- Variables, results, and labels are automatically passed.
- All security checks of the RCR apply to the referenced report.

GTO RPX



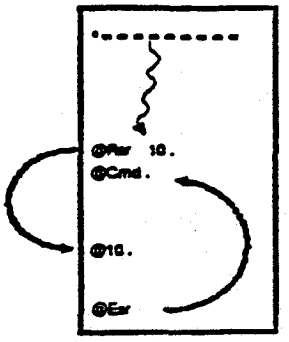
V6-7

Subroutine Review

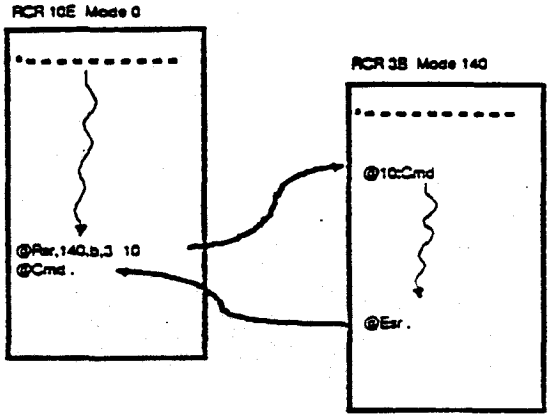
- Internal - reside within the RSR.
- External - reside in any cabinet and drawer.
- Do not have to be registered.
- Called with RSR, CSR, or CALL statements.

Subroutine Review

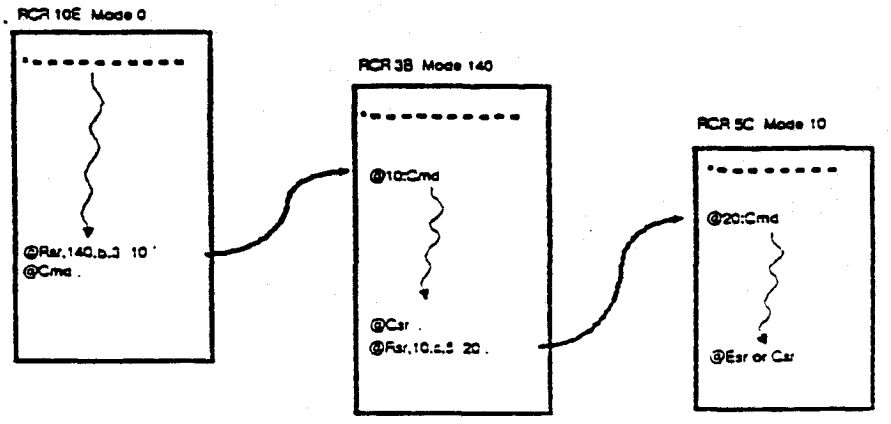
Internal



External with Esr



External with Csr

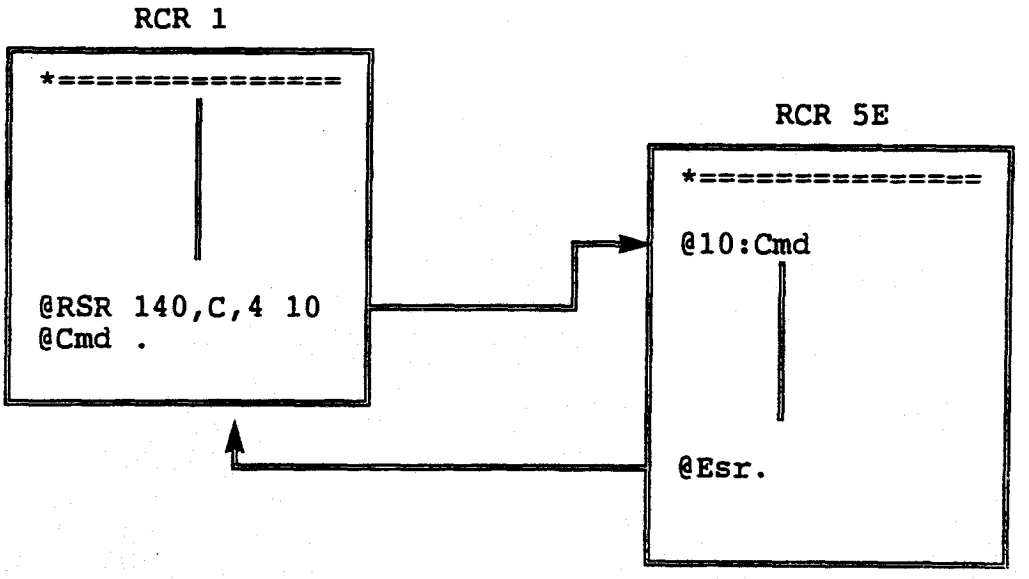


V6-8

RSR
(Run Subroutine)

- Passes control to the subroutine named.
- May be used for internal or external routines.
- Control returned to main routine with ESR (Exit Subroutine) statement.
- Control passed to another external subroutine with CSR (Clear Subroutine) statement.
- Utilizes reserve words CRPT$ and CDRW$.

RSR Statement



V6-9

RSR Statement

Description

The Run Subroutine (RSR) statement transfers control to a specified label to execute a subroutine.

Format

```
@RSR{,c,d,r lab ¦ lab} .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report containing the external subroutine.
lab	Label where the subroutine starts. This field is required.

CALL Statement

- Similar to RSR statement, but has several advantages.

- Transfers control to a subroutine and enables the manipulation of variables.

  - Internally, only variables processed are those used in the called subroutine.
  - Conflicts in variable names are eliminated.
  - Renamed results can be used, but are not transferred back to the main routine.

- RETURN statement returns control to the calling routine.

- Use instead of PEK, PSH, POK, and POP when possible to save system overhead.

CALL Statement

Description

The Call Subroutine (CALL) statement transfers control to a subroutine and enables you to manipulate the values of variables.

Format

```
@CALL[,c,d,r] lab([v,v,...]) .
```

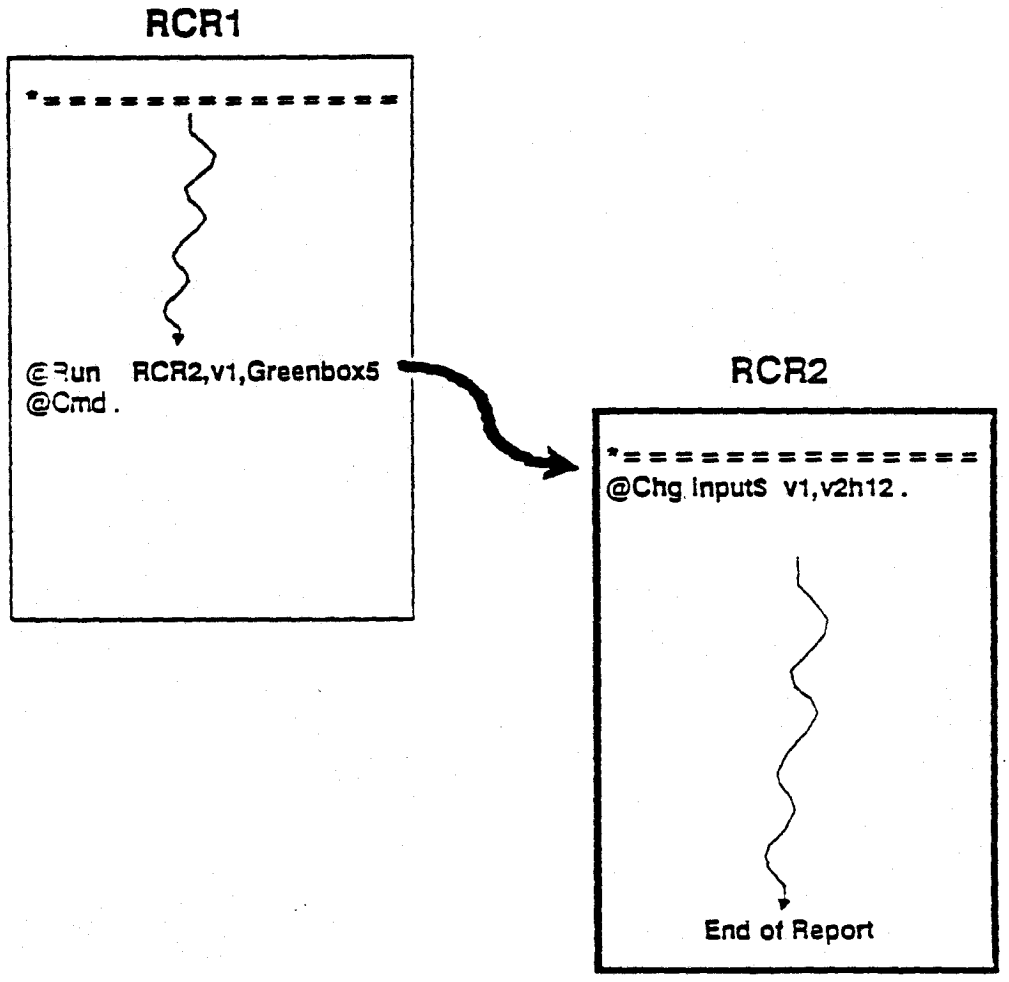
Fields

Field	Description
c	Cabinet of an external subroutine.
d	Drawer of an external subroutine (mandatory if cabinet is specified).
r	Report containing the subroutine (mandatory for external subroutine).
lab	Label where subroutine begins. At the subroutine label, you must specify the parentheses and a corresponding variable for each variable passed on the CALL statement. Use the following format for the label: `@lab:([v,v,...])`
(v,v...)	Variables to pass to the subroutine (maximum of 40). The parentheses are required.

RUN Statement

- Executes one run from another run.
- Allows passing of variables, literal data, and reserve words.
- Called with RUN (Run Start) statement.

RUN Statement



V6-10

RUN Statement

Description

The Run Start (RUN) statement terminates the current run and starts another run.

Format

```
@RUN run[,vld] .
```

Fields

Field	Description
run	Name of the run to start.
vld	Variables, literal data, reserved words, or any combination of these, to transfer to the run. Use INPUT$ to initialize these variables.

Example

```
@run  payroll,v5,v6,SMITH .
```

Starts the run called PAYROLL and sends the variables v5 and v6 and the character string SMITH to that run (which initializes these as variables with INPUT$).

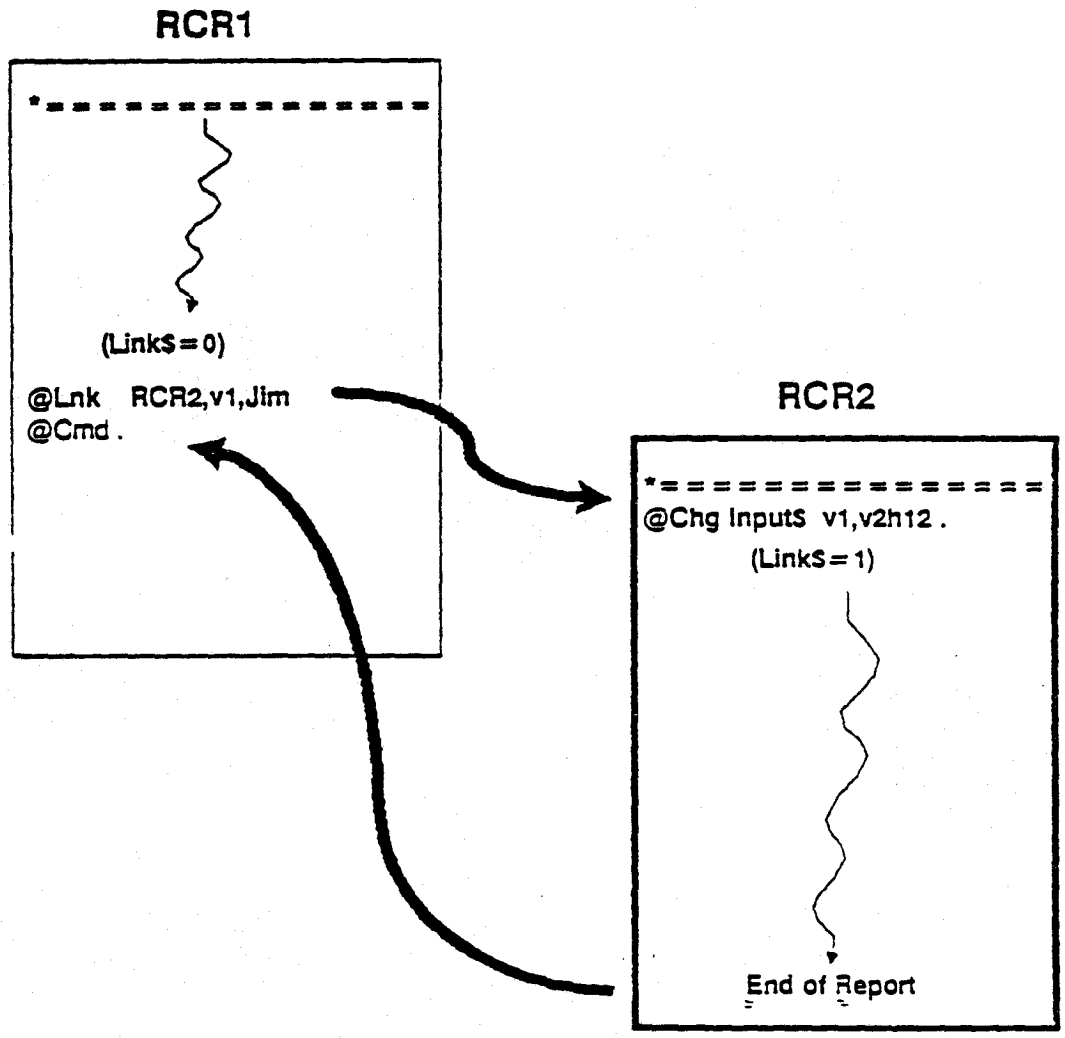
RUN Statement Considerations

- RUN RCR must be registered; PreRun overhead.
- Execution begins at line 3; no other entry points.
- Passes current value (-0) to the started run.
- A new RCR D-bank, Variable Table, and Label Table must be built.
- Calling run terminates as soon as the RUN statement executes.
- Use GTO RPX or RSR when possible.

LNK Statement

- Similar to the RUN statement, but control is returned to the calling RCR.
- Allows passing of variables, literal data, reserve words.
- Called with the LNK (Link to Another Run) statement.
- Utilizes reserve word LINK$.

LNK Statement



V6-11

LNK Statement

Description

The Link to Another Run (LNK) statement executes another run.

Format

```
@LNK run[,vld] .
```

Fields

Field	Description
run	Name of the run to start.
vld	Variables, literal data, reserved words, or any combination of these, to transfer to the run. Use INPUT$ to initialize these variables.

Example

```
@lnk  payroll,v5,v6,SMITH .
```

Executes the run called PAYROLL and sends the variables v5 and v6 and the character string SMITH to that run (which initializes these as variables with INPUT$). Once PAYROLL encounters a GTO END statement, control is returned to the next statement following the @LNK in the originating run.

XQT Statement

- Used to construct and execute run statements.
- Called with XQT (Execute) statement.
- Multiple statements can be executed.
- String length upto 640 characters.

XQT Statement

```
*=================================
@Rdl,Etype$,2,v1,99  1-80 v2s80 .
@Xqt  v2 .

@Brk .
     Search by:
          Status Date
          Ship Date
     Sort by Product Type
@Brk Out,-0,2,4,1,1 .
@Chg v1i2 Curv$+1  If  v1 ge 3 & le 5 gto  lin+1 ; gto 1 .
@Xqt,lin  v1 .
@Dsp,-0 .
@Rel .
@Srh,0,b,2,,,99 dh 'Status Date'  ,861225 .
@Srh,0,b,2,,,99 dh 'Ship date'  ,870614 .
@Sor,0,b,2  '' 'Product Type'     ,1 .
@1:Cmd
```

V6-12

XQT Statement

Description

The Execute (XQT) statement executes any valid run statement.

Format

```
@XQT{,lab ¦ vld} .
```

Fields

Field	Description
lab	Label or line number to execute.
vld	Variables or literal data and reserved words from which to formulate and execute a run statement.

Example

```
@rdl,edrw$,2,v1,099 1-80 v2s80 .
```

Read and execute a line from report number 2 in EDRW$.

Abort Routines

- Subroutines that pass control to the run instead of MAPPER if an abort condition occurs.

- Utilize the RAR (Register Abort Routine) statement.

- Utilize the reserve words:

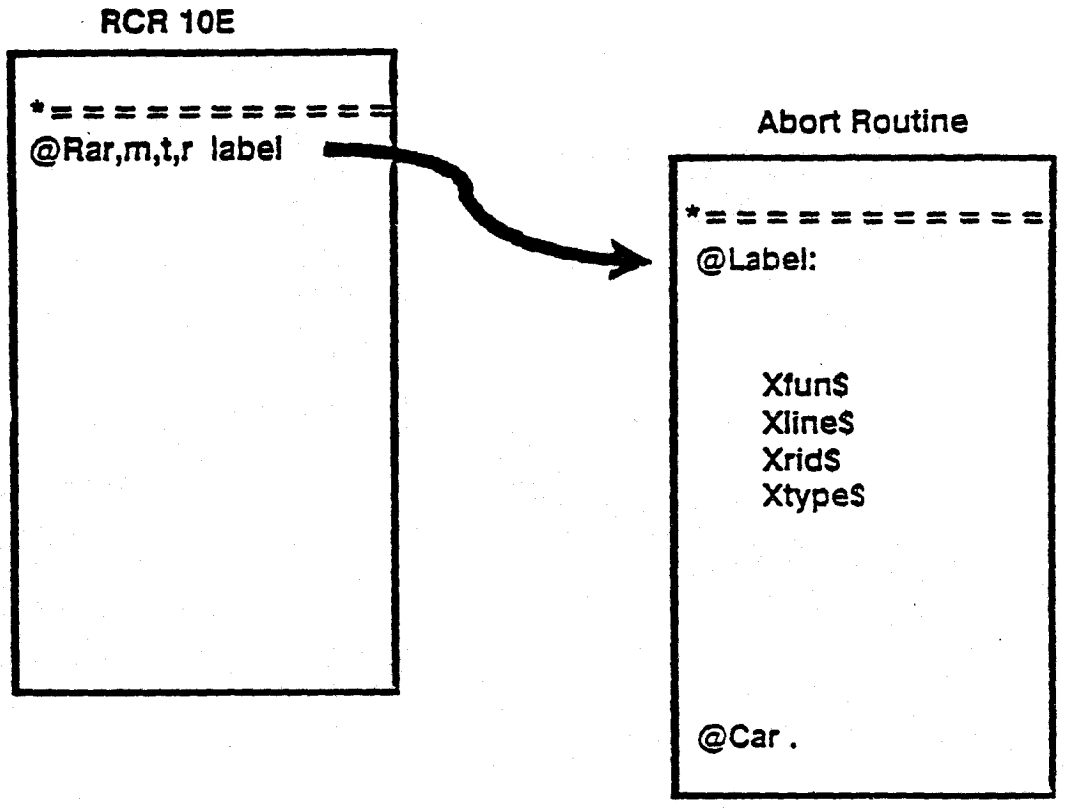
  XFUN$ - last function executed before abort.

  XLINE$ - RCR line number when aborted.

  XRID$ - RCR report number when aborted.

  XDRW$ - RCR drawer number when aborted.

Abort Routines



V6-13

RAR Statement

Description

The Register Abort Routine (RAR) statement registers or names a routine to be executed if the user aborts a run.

Format

```
@RAR{,c,d,r lab ¦ lab} .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of the report containing an external routine.
lab	Label in the report where an abort routine begins. If 0, the beginning of the RCR is assumed.

Example

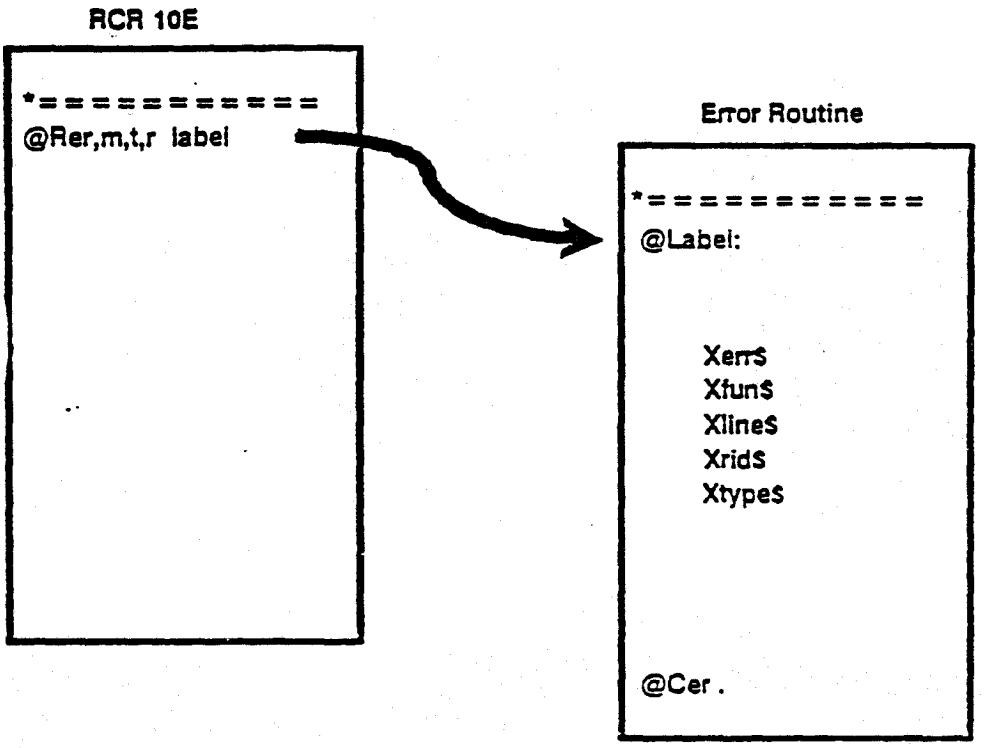
```
@rar,0,c,4 010 .
```

Register the external abort routine in RID 4C, cabinet 0, and specify that the routine start at label 10 in the report.

Error Routines

- Handle recoverable and nonrecoverable errors.
- Utilize the RER (Register Error Routine) statement to register a subroutine to be executed should an error routine occur.
- Should document:
  - Function and line that erred.
  - Variable contents.
  - Time of error.
  - Station and user executing run.
  - Output area contents.
  - Contents of result(s).

RER Statement



V6-14

RER Statement

Description

The Register Error Routine (RER) statement registers a routine to be executed if the run encounters an error.

Format

```
@RER{,c,d,r lab ¦ lab} .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of the report containing an external routine.
lab	Label in the report where an error routine begins. If 0, the beginning of the RCR is assumed.

Example

```
@rer,0,c,5 099 .
```

Register the external error routine in RID 5C, cabinet 0, and specify that the routine start at label 99 in the report.

Error Routine Example

```
*=================================
@Rer 10 .
@Srh,0,b,|   'St Cd'  ,or .
@Dsp,-0 .
      .
      .

@10:Ldv,w  <err>i3=xerr$,<func>h3=xfun$,<rept>i3=xrid$,\
 <line>i3=xline$ .
@Lsm,<err>  <msg>s80 .

     This run (rid - <rept>) has erred on line <line>, while
     executing the  <func>  function.

     The following error message was received -
     <msg>
```

```
*=================================

This run  (rid - 6)  has erred on line  5, while executing
the SRH function.

The following error message was received -
                  < < This field must be a numeric field > >
```

V6-15

Summary

- Label tables should be built at the beginning of every RCR and subroutine.
- Attempt to keep logic confined as much as possible to the 93 line efficiency limit.
- Computational GTOs should be used whenever possible to cut down on the number of commands processed for each IF function.
- The most often used path of a conditional should be the next line, not the specified label.
- @CALL, @GTO RPX and @RSR statements are recommended where possible over the @RUN or @LNK statements because both @RUN and @LNK require additional PreRun processes.
- The difference between the @RUN and @LNK statements respectively is:
  - RUN stops execution of the calling run altogether.
  - LNK suspends execution until execution of the linked run is complete.
- Abort Routines (@RAR) provide the Run Designer with a technique to pass control to the subroutine should the end user deliberately abort the run.
- Error Routines (@RER) provide the Run Designer with a technique to pass control to the subroutine should a recoverable or nonrecoverable error occur.

Exercises

1. This exercise uses code for the EDIT run, found in drawer ___, Report ___. To prepare for this exercise, copy this code into your run control report.

   a. Eliminate the first line of the run (the line beginning with :L) and the variable table lines at the end of the report. Add a line of code to test for the contents of STAT1 on the @LOK statement if someone else has update control of the report. Also, add an @LOG statement to the beginning of the run, then execute the run and obtain a RUNA result. Build Label Tables and Variable Tables.

   b. Move the lines following the first @CHG INPUT$ statement to a second RCR in the drawer. Use either the @GTO RPX or @RSR/ESR statements to branch to the new subroutine. Make any necessary adjustments to the run to recognize labels or variables, and create new Label and Variable Tables. Execute the run, and obtain a RUNA result.

2. Repeat 1b, this time using an @RUN or @LNK statement to branch to the subroutine. Again, adapt the run to pass the contents of necessary variables to the subroutine (in this case, the variable <RID> must be passed within the @RUN or @LNK statement.

   Compare the three RUNA results. Which method of branching is most efficient for this particular run?

3. Describe the differences between Error Routines and Abort Routines.

7

Screen Design Techniques

Module Objectives

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

1. Identify when to use screen reserve words INPUT$, INVAR$, INVR1$, and INSTR$.
2. Define the purpose of the LFC and SFC statements.
3. Compare the SFC and FMT statements.
4. Handle function key input by defining and coding the CHD and KEY statements.
5. Describe the purpose of the Forced Transmit of the OUT statement.

Screen Reserve Words

- **INPUT$**
- **INVAR$**
- **INVR1$**
- **INSTR$**
- @ITV (@OTV)

↓

incompatible across mainframes

(1100 / UNIX)

- don't use

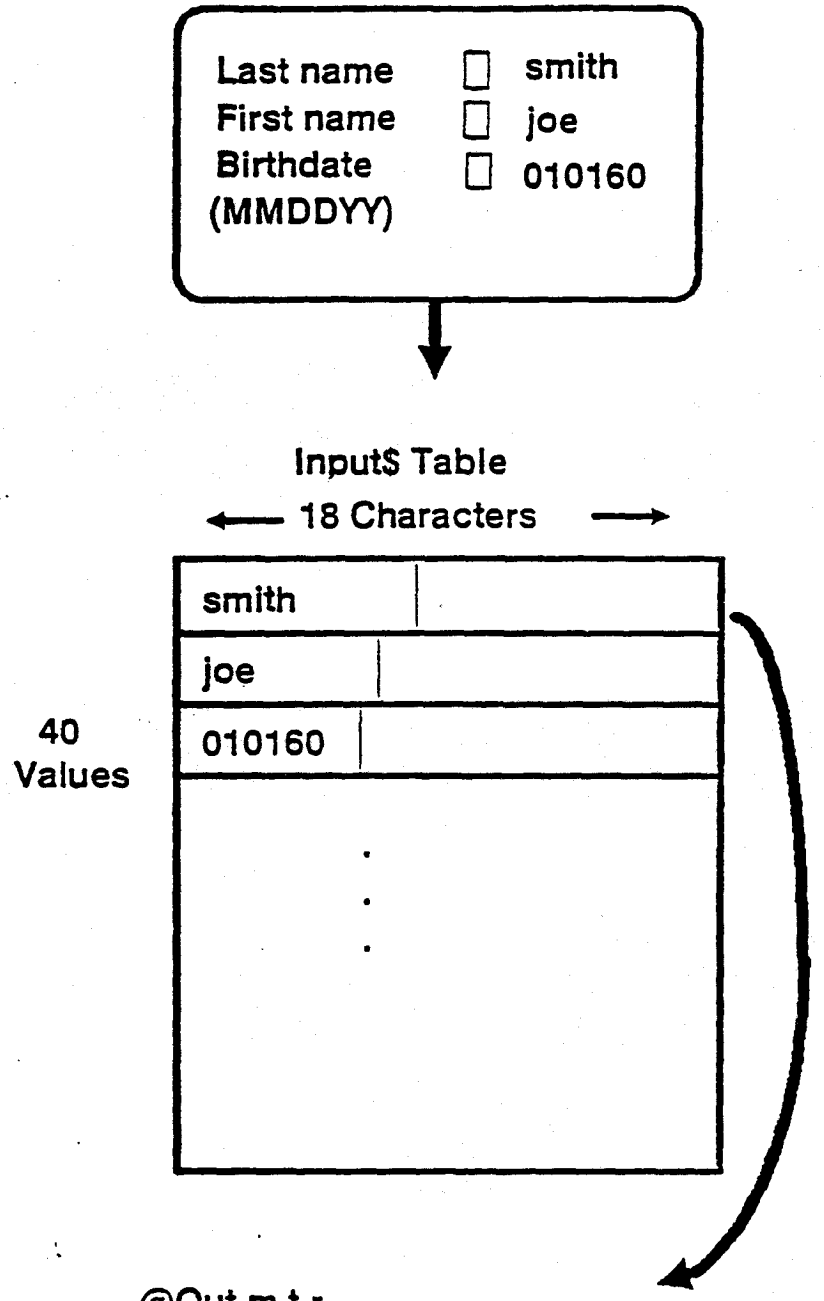
INPUT$

- Utilizes user input from the screen or initial input parameters.
- Data is captured from the character after the leading tab (or control character, if protected screens are used).
- The number of characters loaded from a field depends on the defined length of the variable; maximum 18 characters.
- No string values allowed.
- Information loaded into the INPUT$ table.

INPUT$ Table

- Automatically provided for all data entry screens.
- Fixed in size to accept up to 40 values.
- Each value may contain a maximum of 18 characters.

INPUT$ Table



V7-1

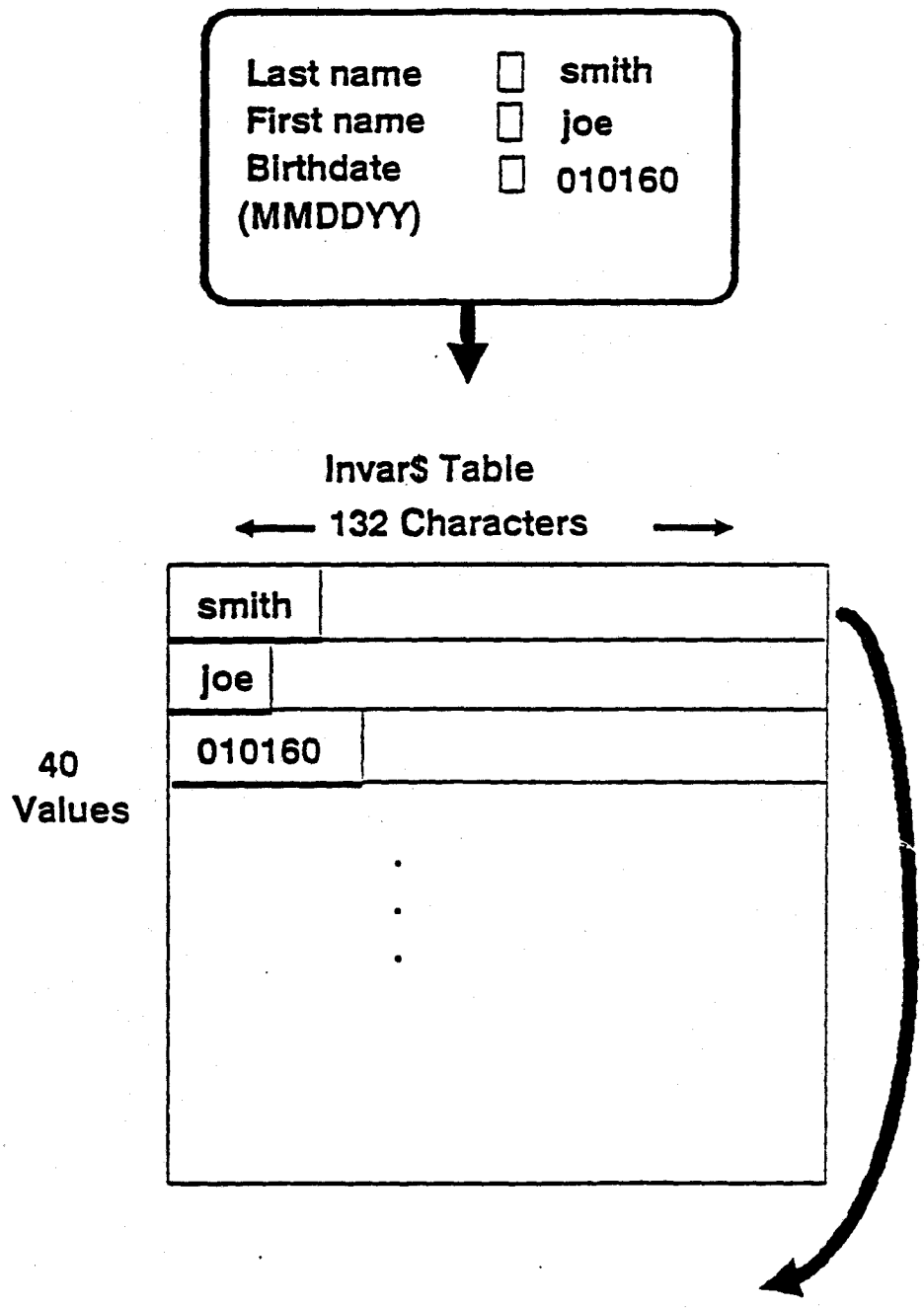
INVAR$

- Utilizes user input from fields on the screen.
- Data is captured from the character after the leading tab (or control character, if protected screens are used).
- The number of characters loaded from a field depends on the defined length of the variable; maximum 132 characters.
- String variables allowed.
- Information loaded into the INVAR$ table.

INVAR$ Table

- Fixed in size to accept up to 40 values.
- Each value may contain a maximum of 132 characters.

INVAR$ Table



@Chg Invar$ v5h20,v6h10,v7i6

@Out,m,t,r

V7-2

INVR1$

- Utilizes user input from fields on the screen (terminated by the size of the variable rather than the size of the field).
- Allows the loading of several fields into one variable.
- The number of characters loaded (including tab characters) depends on the defined length of the variable; maximum 132 characters.
- String variables allowed.
- Do not use with protected or formatted output screens.
- Information loaded into INVR1$ table.

INVR1$ Table

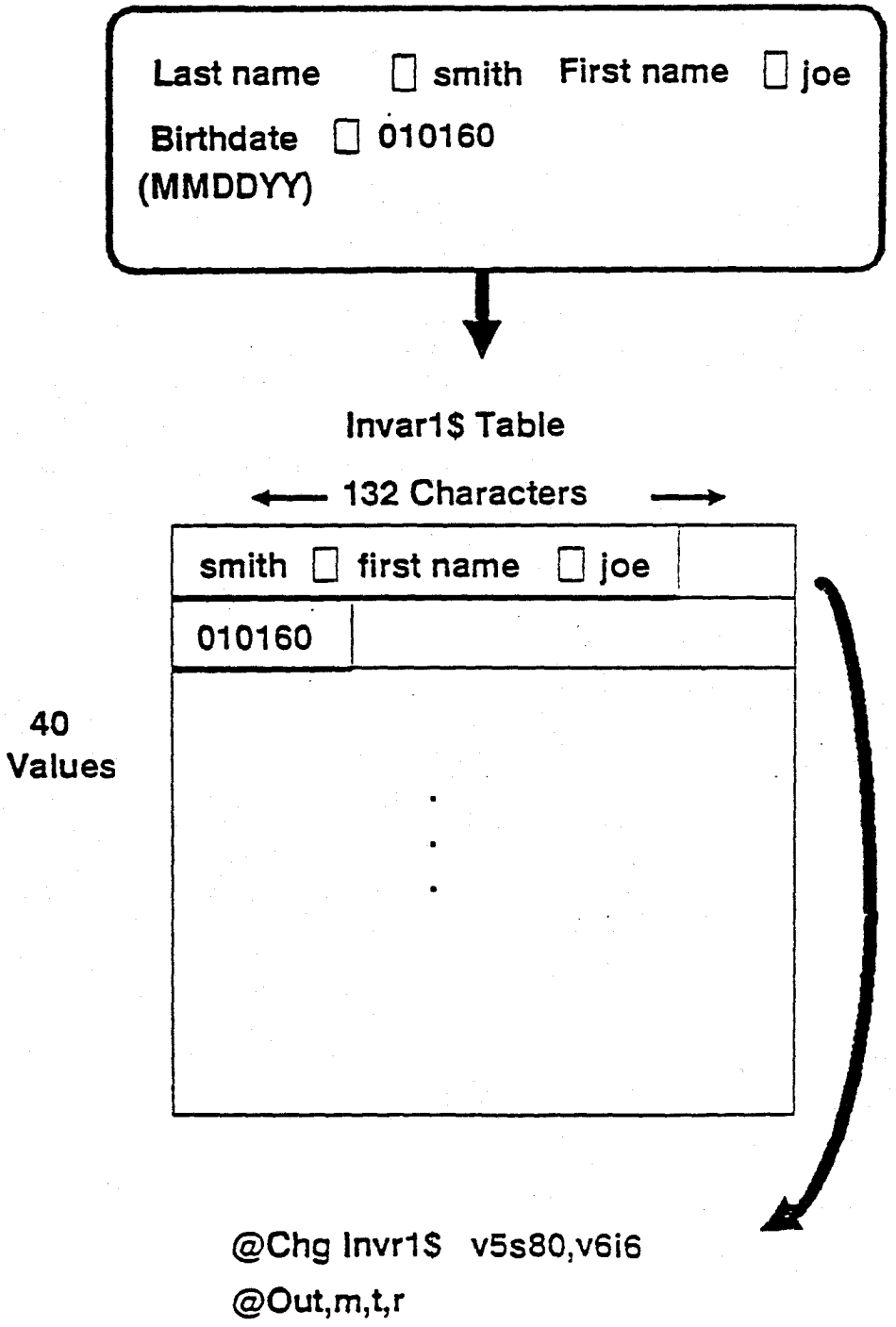
- Accepts more than 40 entries, but 40 values are still the maximum.

- MAPPER stops loading data in this table when one of the following conditions exist:

  - no more data on the screen.

  - the value in the INVR1$ table is full.

INVR1$ Table



V7-3

INSTR$

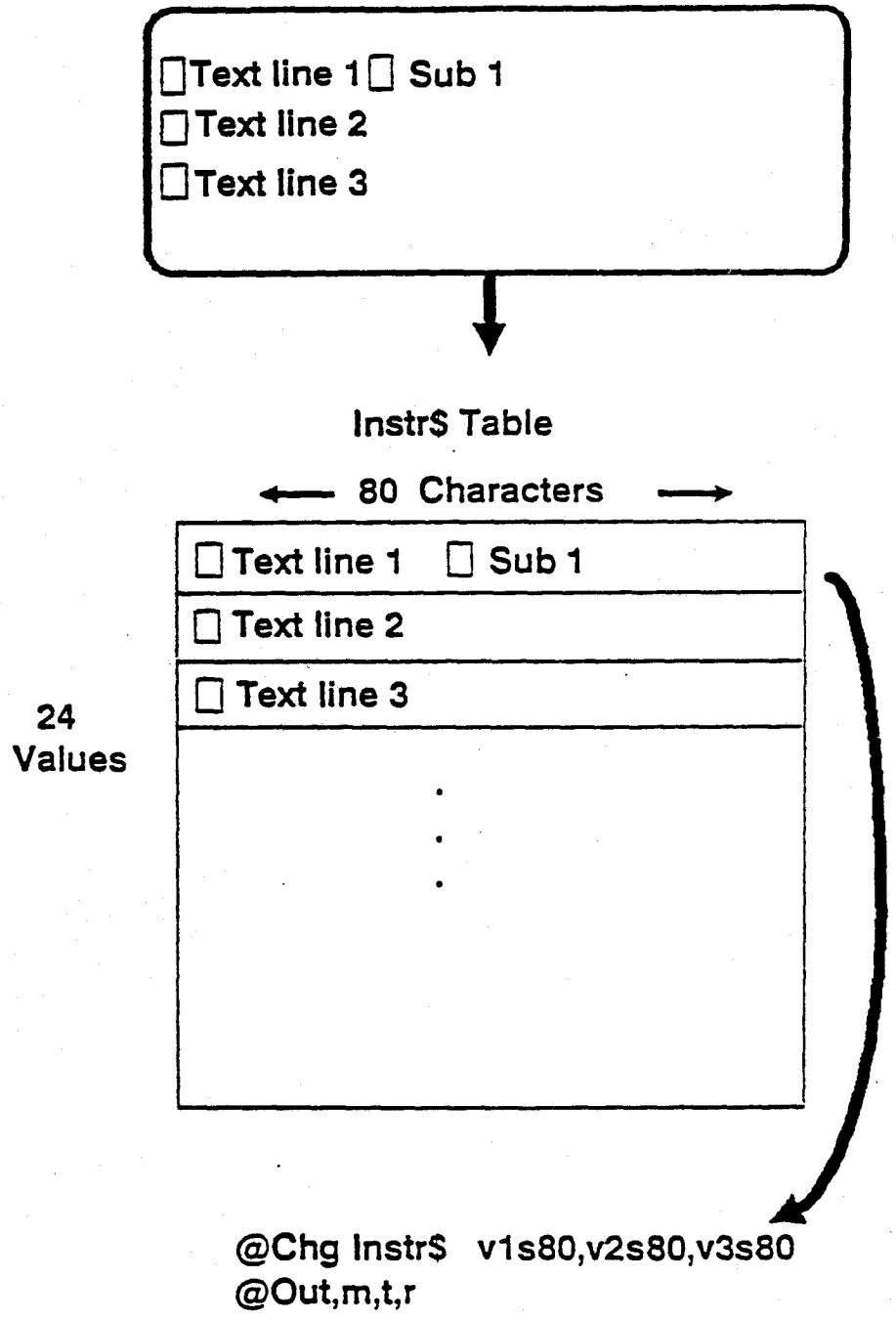
- Utilizes user input from entire lines on the screen.
- No leading tabs required.
- Input variables are terminated by the end of the line; 80 - 132 characters.
- Maximum usable number of variables is the vertical screen size.
- String variables allowed.
- Do not use with protected or formatted output screens.
- Information loaded into INSTR$ table.

INSTR$ Table

- Width fixed at 80 characters, unless a 132-column screen is used.

- Length set to vertical length of screen, typically 24 lines.

INSTR$ Table



V7-4

Advanced Screen Techniques

- Save/set the format of a report or result.
  - LFC (Load Format Characters)
  - SFC (Set Format Characters)
- Display function masks based on headers defined for specific cabinets and drawers.
  - OUM (Output Mask)
- Gain function key control.
  - CHD (Command Handler)
  - KEY (Function Key Input)

LFC
(Load Format Characters)

- Saves the format of a report or result currently on display (-0).
- Format captured in a variable.
- Runs must first be registered as format sensitive.
- Permits more than the maximum of six predefined formats.
- Typically used with SFC (Set Format Characters).

LFC Statement

Description

The Load Format Characters (LFC) statement captures the format of the report or result currently on display (-0).

Format

```
@LFC v .
```

Fields

Field	Description
v	Variable to capture the format of the report or result currently on display.

Example

```
@lfc v1s80 .
```

Capture the format of the current result in variable v1.

SFC
(Set Format Characters)

- Sets the format of a report or result.
- Restores a display format previously saved with an LFC Statement.
- Customizes a display format.
- If a report is specified, the report becomes the current -0.
- DSP, OUM, or OUT statements must be used after an SFC statement.

SFC Statement

Description

The Set Format Characters (SFC) statement sets the format of a report or result.

Format

```
@SFC[,c,d,r] vld .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of report or result on which to set the format. Default = -0.
vld	Variables, literal data (including spaces and Xs or other nonspace characters), or both, representing the format you want to set.

Example

```
@sfc 'xxxxxxxxxx  xxxxx' .
@dsp, -0 .
```

Set the format for columns 1 through 10 and 12 through 16 in the current result and display it.

LFC and SFC

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
@Dsp,0,b,2 .                          User manipulates screen
@Lfc  v1s80 .                         Detect columns on display
                                      and load into v1
@Srh,0,b,2  " 2-2  ,sh .              Search 2b and create -0
@Sfc  v1 .                            Set format for Dsp stmt
@Dsp,-0 .                             Display only requested cols.


@Sfc  'xxxxxx     xxx      xxxxxxxxxxx'
@Dsp,-0 .                             Display only columns 1-6,
                                      11-13,20-31
```

V7-5

OUM
(Output Mask)

- Display a function mask based on headers defined for a specific cabinet and drawer.

- Utilizes reserve word INMSV$:

  - must precede OUM statement.
  - places user input from mask into variables, which can be placed in other run statements to manipulate data within a report.
  - variables should be S type.

OUM Statement

Description

The Output Mask (OUM) statement displays a blank function mask on the basis of the headers defined for the specified cabinet and drawer.

Format

```
@OUM,ic,id[,ir,,,,,title .
```

Fields

Field	Description
ic,id,ir	Cabinet, drawer, and report number of the report that contains the mask. Default = report 0.
if	Format in which to display the mask.
title	Title (up to 12 characters) to be displayed above the mask.

OUM Example

```
@Chg Inmsv$ v1s79,v2s120,v3s120,v4s132 .
@Oum,0,d,2 .
@Ldv v8i2=1 .
@1:Ldv v10i3=v2(v8-3),v11i3=v3(v8-3),v12s80=v4(v10-v11) .
@Ldv,p v10=v10,v11=v11,v12=v12 .
@Srh,0,d,2 v1 v10-v11 ,v12 .
@Dsp,-0.
```

V7-6a

OUM Example

Output Mask

```
dh
.                    Corporate Production Status
*St  Order   Product   Ord  Cust   Unit      Extended    Req'd    Sale
*cd  Numbr   Type      Qty  Code   Retail      Retail    Delivr   Rep   Customer
*-------------------------------------------------------------------------
 **  *****  *********  ***  ****  ********  ********  ********  ****  ********
                          feds
```

Variables Loaded

```
.date 06 sep 88 15:01:07          Report Generation          Mapcoord
v1 = dh
v2 = 026
v3 = 004
```

Srh Results

```
.date              15:22:49        Rid 2D        06 Sep 88
.                    Corporate Production Status
*St  Order   Product   Ord  Cust   Unit      Extended    Req'd    Sale
*cd  Numbr   Type      Qty  Code   Retail      Retail    Delivr   Rep   Customer
*-------------------------------------------------------------------------
  or 97441  Blackbox        feds                        890202   djr
  or 54237  Blackbox        feds                        890202   mg
  or 54438  Blackbox        feds                        890223   djr
  or 99842  Whitebox        feds                        890613   spc
```

V7-6b

FMT
(Format)

- Similar to the SFC statement, but field names or column positions are used to designate report fields.

- DSP, OUM, or OUT statements must be used after an FMT statement.

FMT Statement

Description

The Format (FMT) statement creates a display format by selecting which fields of a report or result to display on a following DSP, OUM, or OUT statement.

Format

```
@FMT[,c,d,r] field[,...,field] .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of the report or result from which to display fields.
field	Fields to display (can be field names or column-character positions) the mask

Example

```
@fmt 'stcd','shipdate','custcode' .
@dsp,-0 .
```

Display the ST CD, SHIP DATE, and CUST CODE fields of the current -0.

FMT Example

```
@Chg Inmsv$ v1s79,v2s120,v3s120,v4s132 .
@Oum,0,d,2 .
@Ldv v8i2=1 .
@1:Ldv v10i3=v2(v8-3),v11i3=v3(v8-3),v12s80=v4(v10-v11) .
@Ldv,p v10=v10,v11=v11,v12=v12 .
@Srh,0,d,2 v1 v10-v11 ,v12 .
@Fmt,0,d,2 'St Cd','Product Type','Cust Code' .
@Dsp,-0 .
```

V7-7a

FMT Example

Output Mask

```
dh
.                    Corporate Production Status
*St  Order   Product   Ord  Cust   Unit     Extended   Req'd    Sale
*cd  Numbr   Type      Qty  Code   Retail     Retail   Delivr   Rep   Customer
*-------------------------------------------------------------------------
 **  ******  *********  ***  ****  ********  ********  ********  ****  ********
                         feds
```

Variables Loaded

```
.date 06 sep 88 15:01:07          Report Generation          Mapcoord
v1 = dh
v2 = 026
v3 = 004
```

Srh Results with FMT

```
.date              15:22:49        Rid 2D        06 Sep 88
.                    Corporate Production Status
*St    Product     Cust
*cd      Type      Code
*-------------------------
  or   Blackbox    feds
  or   Blackbox    feds
  or   Blackbox    feds
  or   Whitebox    feds
```

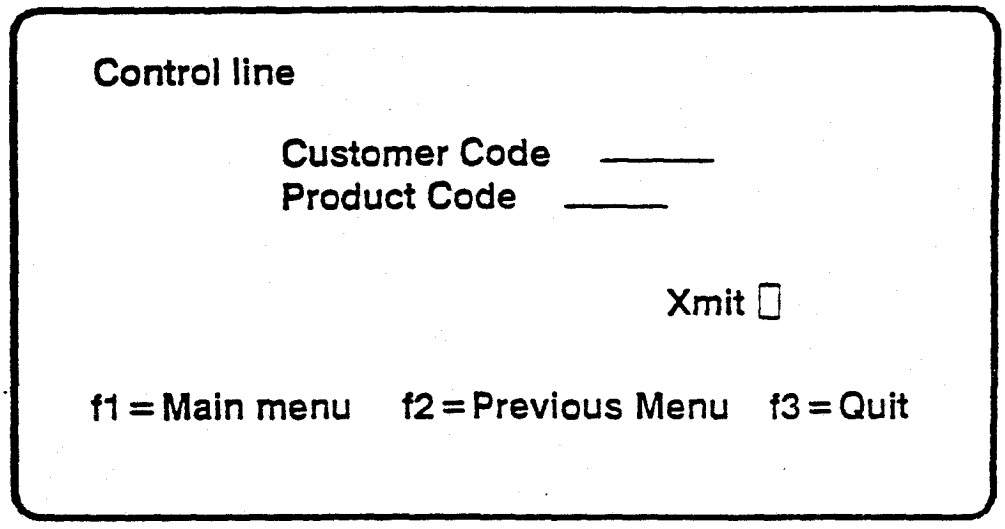
V7-7b

CHD
(Command Handler)

- Used to bypass MAPPER's normal chain of events when function keys are pressed or when information is entered on the control line.

- Utilizes reserve words ICVAR$ and FKEY$.

CHD
(Command Handler)



V7-8

CHD Statement

Description

The Command Handler (CHD) statement registers a routine to be executed whenever the user of the run enters information in the control line after a DSP, OUT, or SC statement.

Format

```
@CHD[,c,d,r,rel?] lab .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of the report that contains an external command handler routine.
rel?	Transfer release control (^) to the run, Y or N. Default = N.
lab	Label where the command handler routine starts. Use 0 to cancel the currently registered command handler routine.

CHD Example

```
*=============================
@Chd,,,y  99 .
@10:Brk .
                    Enter customer code          ,
                    Ener product code            ,

                                          Transmit from here -  ,
      f1 = main menu   f2 = previous menu     f3 = quit

@Chg icvar$  v1s80 .
@Brk chg invar$   <cust>a12,<prod>a8 .
@Out,0,f,-0,2,9,1,1 .
@If  Fkey$ = -1,(10),0,(20),1,(30),2,(40),3,(50) .
@20 .
     Process Data
@30 .
     Branch to main menu
@40 .
     Branch to previous menu
@50 .
     Quit
@99 .
     Handle entry on control line or caret
```

V7-9

KEY
(Function Key Input)

- Enables the designer to determine the function key the run user pressed after a noninterim display.
- Utilizes reserve word FKEY$.
- Format: @KEY .

KEY
(Function Key Input)

```
*=================================
@Chd,,,y  99 .
@10:Brk .
                  Enter customer code          ,
                  Ener product code            ,

                                    Transmit from here -  ,
      f1 = main menu   f2 = previous menu    f3 = quit


@Brk key  chg  Invar$  <cust>a12,<prod>a8 .
@Out,0,f,-0,2,9,1,1 .
@If  Fkey$  =  0,(20),1,(30),2,(40),3,(50) .
@20 .
     Process Data
@30 .
     Branch to main menu
@40 .
     Branch to previous menu
@50 .
     Quit
@99 .
     Handle entry on control line or caret
```

V7-10

OUT
(Output)

- The thirteenth subfield of the OUT statement controls how output containing spaces is displayed on the terminal:

  - Blank subfield
  - A in subfield
  - B in subfield

- @OUT,c,d,r,l,q[,outl,tabp,erase?,interim?, pdq,protect,fxmt?,**ab?**,blink] .

OUT
Blank in Subfield

- Non-space data is sent to the screen as is while spaces are handled:
  - If five or less characters exist between characters, spaces are sent.
  - If more than five spaces exist between characters, spaces are not sent.

OUT
Blank in Subfield

```
*==============================
@Brk .
  Screen Fields -
x      1          2          3          4          5


@Brk  Out,0,f,-0,2,3,1,,,y,,p  .
1   123456        12345       1234      1234567      123456
 aa        bbbbbbb       ccccccc     dddd         eeeee       fffffff
@Brk Out,-0,3,2,2,,n .
```

First Out

```
  Screen Fields -
x      1          2          3          4          5
```

Second Out

```
aa     1  bbbbbbb     ccccccc     dddd     4  eeeee    5  fffffff
```

V7-11

OUT
A in Subfield

Sends output line to the terminal, exactly as is, including all spaces.

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
@Brk .
  Screen Fields -
x      1              2              3              4              5


@Brk Out,0,f,-0,2,3,1,,,y,,p .
1  123456         12345          1234        1234567        123456
 aa          bbbbbbb        ccccccc       dddd          eeeee          fffffff

@Brk Out,-0,3,2,2,,n,,,,,a .
```

First Out

```
  Screen Fields -
x      1              2              3              4              5
```

Second Out

```
aa          bbbbbbb        ccccccc       dddd          eeeee          fffffff
```

V7-12

OUT
B in Subfield

No spaces are sent to the terminal.

```
*===========================
@Brk .
  Screen Fields -
x      1            2          3          4          5


@Brk  Out,0,f,-0,2,3,1,,,y,,p .
1  123456           12345      1234       1234567    123456
 aa        bbbbbbb       ccccccc    dddd       eeeee      fffffff

@Brk Out,-0,3,2,2,,n,,,,,b ..
```

First Out

```
  Screen Fields -
x      1            2          3          4          5
```

Second Out

```
aa     1 bbbbbbb   2 ccccccc   3 dddd    4 eeeee   5 fffffff
```

V7-13

OUT
Forced Transmit

- Designed to accept a screenful of input, regardless of where the user transmitted on the screen.

- A "Y" is placed in the twelfth subfield of the OUT statement.

- Resets the IO$, LLP$, and DLP$ counters.

OUT
Forced Transmit

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
@Brk .
          Enter your name                    ,
                    Employee number          ,
                    Home number              ,
                    Work number              ,
@Brk Out,-0,2,10,1,1 .
@If  Curv$ = 10  Gto 1 .
@Out,-0,11,1,10,5,,,,,Y  .
@1:Chg  Input$  <name>a12,<emp>i6,<home>a12,/
  <work>a12 .
```

V7-14

Summary

- INPUT$:
  - Up to 40 variables; maximum size of a variable is 18 characters.
  - No strings allowed.
  - Accepts values from external source.
  - CHG INPUT$ must be placed before the OUT or SC.
- INVAR$:
  - Up to 40 variables.
  - Strings allowed.
  - CHG INVAR$ must be placed before OUT or SC.
- INVR1$:
  - Same as INVAR$ except embedded tab characters are picked up.
  - Can load several fields into one variable.
  - Should not be used with protected or formatted output screens.
- INSTR$:
  - Accepts entire lines from screen; data terminated by RETURN.
  - Maximum number of entries is vertical to screen size.
  - Does not require leading tab; starts picking up from either control line or last SOE.
  - Strings allowed.
  - CHG INSTR$ must be placed before OUT or SC.
  - Should not be used with protected or formatted output screens.
- LFC (Load Format Character) and SFC (Set Format Character) are used to capture and set the displayed columns on the screen.

- OUM (Output Mask) is implemented when the user desires to enter data via a mask vs. input screens. FMT (Format) serves the same function as the SFC but for named fields.

- CHD (Command Handler) is used to gain control over the control line and the function keys. KEY is used to solely control the function keys and must be reissued before any DSP, OUT, or SC statement.

- The thirteenth subfield of the OUT statement controls the way spaces are sent to the screen. There are two alternatives; A option to send any spaces, and the B option to send no spaces.

- The twelfth subfield of the OUT statement controls the Forced Transmit feature. The primary purpose is to pick up any values from a screen that might otherwise be omitted.

Exercises

1. Write a run to see how each of the four screen input reserve words - INPUT$, INVAR$, INVR1$, and INSTR$ - handles input. Include a short input screen within the output area of the run. This screen should consist of two or three input fields on a single line; each field should start with a leading tab and end with a comma. Display the contents of each variable loaded with each reserved word.

2. Describe the purpose of the LFC and SFC statements when used in conjunction with each other.

3. Compare the SFC and FMT statements.

4. Use the @CHD or @KEY statements to control branching within a run based on which function keys are pressed.

5. Describe the purpose of the Forced Transmit of the OUT statement.

8

Basic Screen Control Techniques

Module Objectives

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

1. Identify the components of 4-to-1 and 5-to-1 output screens and distinguish between these two methods of screen design.
2. Use the Screen Control (SC) statement, along with selected screen commands, to create screens.

Screen Design Methods

- OUT statement P,I,E options
- 4-to-1
- 5-to-1
- SC (Screen Command)

OUT with Options

- Edit characters take up at least one screen position.

- P,I,E cannot accomodate commas on the screen. They serve as field delimiters.

- The Control Line cannot be written over or manipulated.

- Not much flexibility!

OUT with Options

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
@Ldv v10i6='',v20a12='',v3a4='',v1i1=1 .
@2: Brk Chg Invar$ v10,v20,v30
          Employee Status
      Employee Life # [ ]      2v10,
      Last name       [ ]      1v20,
      Org #           [ ]      2v30,

                              Xmit from here [ ]

@Brk Out,-0,2,9,1,1,,,,i,,,y .
*v10*
*v20*
*v30*
@If v2 = '' Gto 2 ; Gto end .
```

V8-1

4-to-1

- Four lines of input produce one line of output.
- Character by character screen control is allowed.
- A "4" is specified in the eleventh subfield of the OUT statement.
- Replaces the need for P,I,E options of the OUT.
- Terminal type dependent.

4-to-1

Four lines of input:

1. M Type Line - controls intensity, tab positioning, blinking.

2. N Type Line - controls editing, justification.

3. Emphasize Line - controls emphasis characters such as underline, strike through.

4. Actual Data.

4-to-1

```
*=====================================*
@Ldv v10h6='',v20a12='',v30h4='',v1i1=1 .
@2:Brk chg invar$ v10,v20,v30 .
L                                    L
C                                    C

          N          L          N          L
          C          C          C          C
                  Employee Status

          N                                 L
          C                                 C

          N                        L     N  L
          C                        B     C  C
                Employee Life Number   v10

          N                     L       N  L
          C                     A       C  C
                Last name       v20

          N                          L    N  L
          C                          B    C  C
                Organization Number     v30

          N                                 L
          C                                 C


          N                          L  N   L
          C                          P  C   C
                          Xmit from here -

          N                                 L
          C                                 C

@Brk Out,-0,2,2,10,1,1,,,,4 .

*v10*
*v20*
*v30*
@If v10 = '' gto 2  ; if v20 = ''  gto 2  ;  if v30 = '' gto 2 .
@Gto end .
```

V8-2

5-to-1

- Similar to 4-to-1, but with an extra line to control color monitors.

- A "5" is specified in the eleventh subfield of the OUT statement.

- Utilizes SCREENGEN - a run used to generate 5-to-1 screens from manual user input.

5-to-1

Background Colors

Letter Colors	Blk	Red	Grn	Ylw	Blu	Mgt	Cyn	Wht
Black	@	H	P	X		h	p	x
Red	A	I	Q	Y	a	i	q	y
Green	B	J	R	X	b	j	r	z
Yellow	C	K	S	[	c	k	s	{
Blue	D	L	T	\	d	l	t	|
Magenta	E	M	U	]	e	m	u	}
Cyan	F	N	V	^	f	n	v	~
White	G	O	W	-	g	o	w	?

V8-3

Screen Control

- Provides most of the capabilities of the OUT statement, with additional enhancements.

- Makes use of Screen Commands which are structured like other run statements, with command names, fields, and subfields.

- Creating and maintaining screens with the SC statement is easier and more efficient than the traditional 4-to-1 and 5-to-1 output methods.

Screen Control

```
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
prep                                    clear and protect screen
def,1,no,red/whi                        define an attribute
fld,3,20,7,38,,(rv,pr,whi/blu)          outer border field
fld,4,22,5,34,,(pr,yel/bla)             inner border field
fld,3,28,,22,f,(pr,whi/red)             screen title field
pc,5,27;Report ;fld,,,,16,u,(ai)        'Report' field
pc,6,27;Format ;fld,,,,1,u,1            'Format' field
pc,7,29;Line ;fld,,,,7,u,1              'Line' field
pc,5,34                                 final cursor


@Sc,-0 'l'                              Read screen commands
                                        from the current result
```

V8-4

SC Statement

- Format One:

  @SC,,,,,,tabp,sn,lab o scmnd .

- Format Two:

  @SC,m,t,r,l,q,tabp,sn,lab o fldtxt .

- Options:

  B Blink

  C Center

  L Line Control

  Q Quick Output

  T Field Text

  U Update Control

  X Forced Transmit

Screen Commands

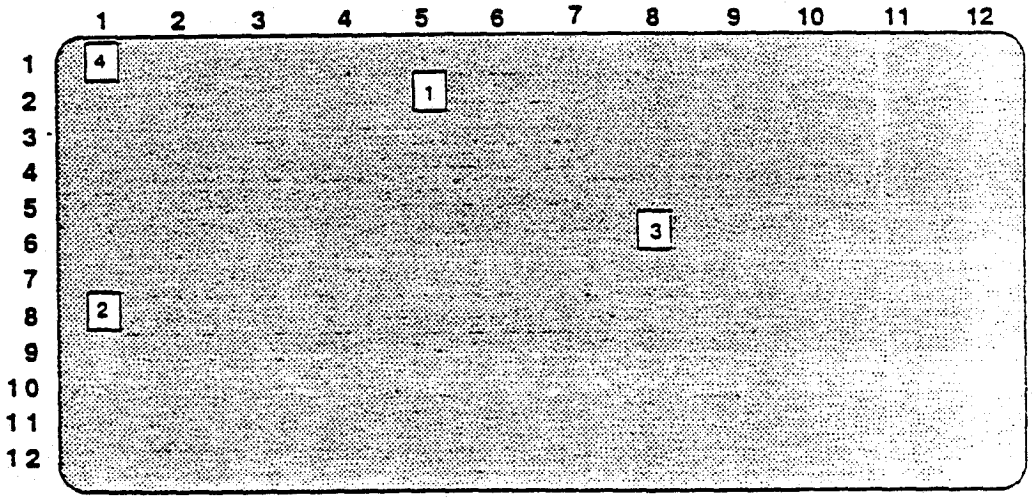
- Cursor Control
- Screen Editing
- Field Attribute
- Emphasis
- Special Commands

Cursor Control Commands

- HC (Home Cursor)
- PC (Position Cursor)
- CR (Cursor Return)
- TAB (Tab)

Cursor Control Commands

```
*==============
PC,2,5      (1)
CR,6        (2)
PC,-2,8     (3)
HC          (4)
```

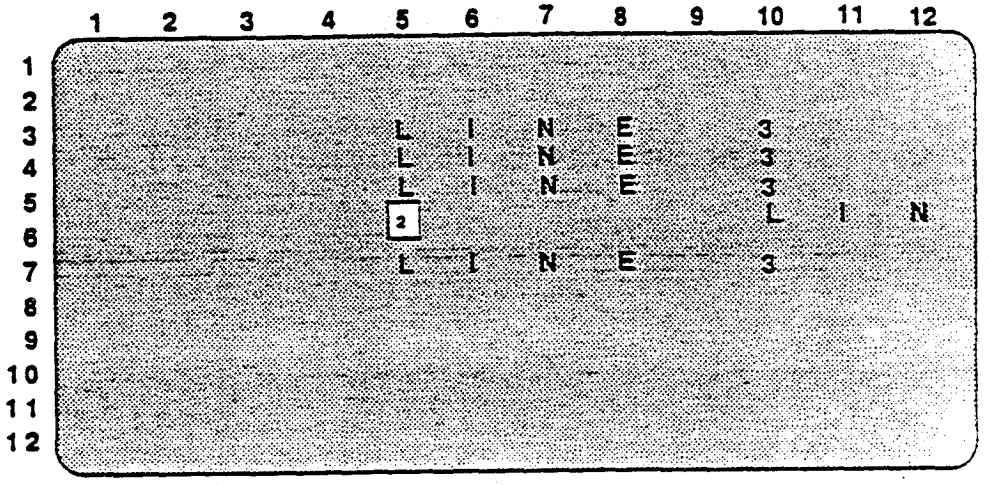


Screen Editing Commands

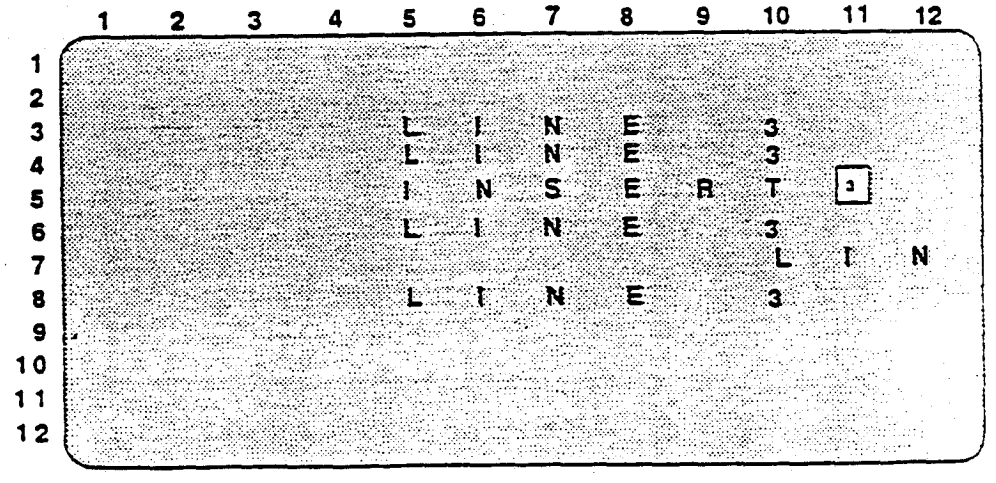
- CS (Clear Screen)
- ED (Erase Display)
- EUD (Erase Unprotected Display)
- EEL (Erase to End of Line)
- IL (Insert Line)
- DL (Delete Line)
- DUP (Duplicate Line)

Screen Editing Commands

```
*=============
CS;PC,3,5;'Line 3'      (1)
DUP,4;PC,6,5;IIL,5      (2)
```




```
PC,-1;IL,1;'Insert'     (3)
```



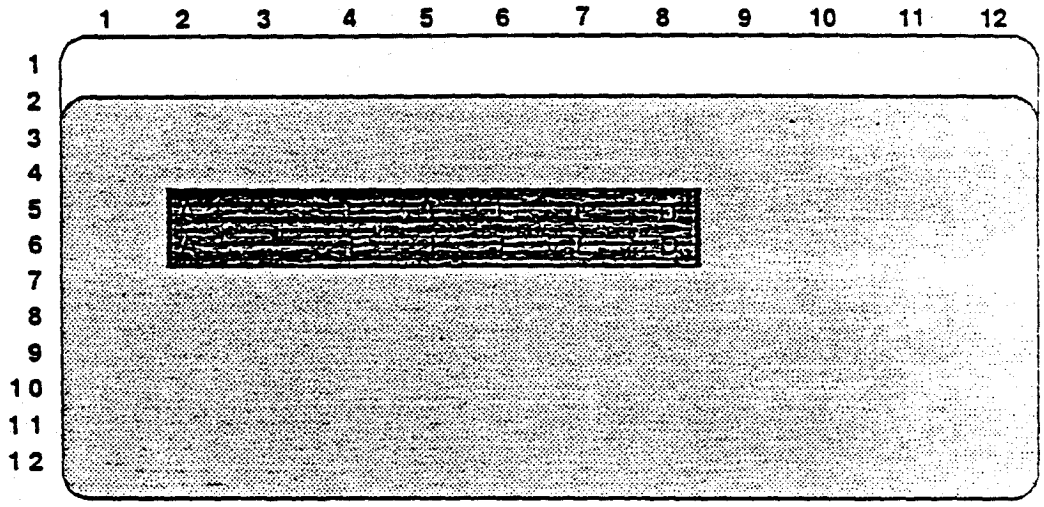
V8-6

Field Attribute Commands

- ATT (Define current attribute)
- DEF (Define subsequent attributes)
- FLD (Generate field)

Field Attribute Commands

```
*====================
DEF,1,PR,RV,WHI/GRE
PC,2,1;ATT,1
FLD,5,2,2,7,,(AI,WHI/BLU),1,'A Field'
```



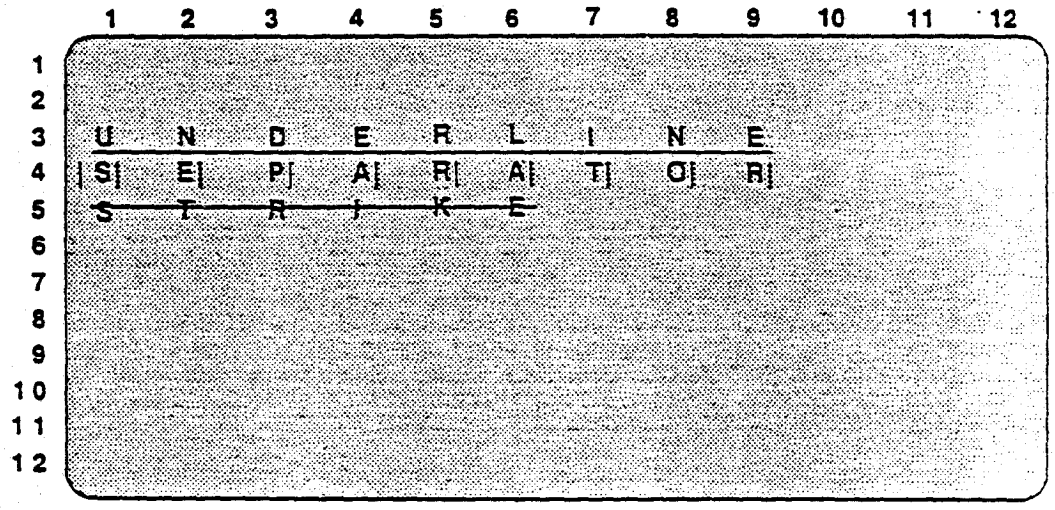
V8-7

Emphasis Commands

- EMP (Emphasis)

Emphasis Commands

```
*====================
PC,3,1;EMP,u;'Underline'
PC,4,1;EMP,|;'Separator'
PC,5,1;EMP,-;'Strike'
```

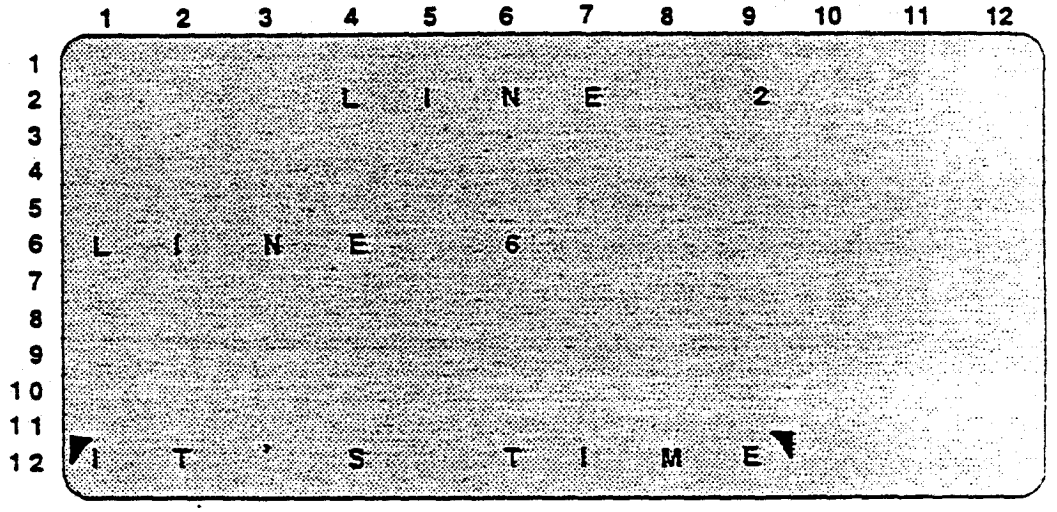


Special Commands

- **PREP**
- **DATA**
- **MSG**
- **SOE**
- **PRT**
- **END**

Special Commands

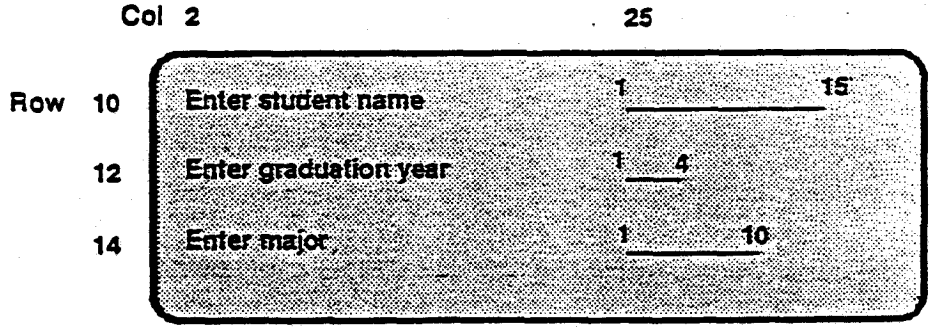
```
*====================
MSG,2,C,(AI,RV,WHI/BLU),'Line 2'
MSG,6,,(BL,RV,WHI/GRE),'Line 6'
PC,12,1;LB;'It';TIC;'s time';RB
```



V8-9

Screen Control Example
SC Within the RCR

```
*=================================
@Brk .
prep
data,10,2,6
Enter student name

Enter graduation year

Enter major

def,3,pr,whi/bla
fld,10,25,1,15,u,(ao,ts),3
fld,12,25,1,4,u,(no,ts),3,1988
fld,14,25,1,10,u,(ai,ts),3
pc,10,25
@Brk chg invar$ <name>a15,<year>i4,<major>a10 .
@Sc,-0 'T' ,1989 .
@If <year> lt 1988 gto 12 .
                .
                .
```



V8-10

Screen Control Example
SC Outside the RCR

```
                Rid 2H
*=================================
prep;pc,3,15;'In Pursuit of Trivia'
pc,6,1;'Geography';pc,6,24;soe;fld,6,26,1,1,,(ts,ai),(pr)
pc,8,1;'Entertainment';pc,8,24;soe;fld,8,26,1,1,,(ts,ai),(pr)
pc,10,1;'History';pc,10,24;soe;fld,10,26,1,1,,(ts,ai),(pr)
pc,12,1;'Arts and Literature';pc,12,24;soe;fld,12,26,1,1,,(ts,ai),(pr)
pc,14,1;'Science and Nature';pc,14,24;soe;fld,14,26,1,1,,(ts,ai),(pr)
pc,16,1;'Sports and Literature';pc,16,24;soe;fld,16,26,1,1,,(ts,ai),(pr)
pc,18,1;'Exit';pc,18,24;soe,fld,18,26,1,1,,(ts,ai),(pr)
pc,20,1;'Tab to the category of your choice and press Xmit';tab,2
```

```
                Rid 6F
*=================================
@Sc,0,h,2,,,,186,99 .
@If Soev$ = 5,(10),7,(20),9,(30),11,(40),13,(50),15,(60),17,(70) .
@10 .
.
@20 .
.
.
@30 .
.
.
.
@99 . Error on station number
```

V8-11

Screen Control Example
Screen Commands and SC

```
*=============================================
@Srh,0,b,26  d  45-4   ,amco .
@Dsp,-0,,,,y .
@Sc      'ul'  Msg,1,,,TicS' Search result; enter L , Xmit to gain line ctrl'TicS .
```

V8-12

Summary

- The components of 4-to-1 screens:
  - M line controls intensity, tab positioning, and blinking of characters.
  - N line controls editing, and justification of characters.
  - Emphasis line controls emphasis characters.
  - Data/Output line.
- 4-to-1 screens are specified by placing a 4 in the eleventh subfield of the OUT statement.
- 5-to-1 screens provide an additional line between the N line and Emphasis line to specify color.
- SC (Screen Control):
  - Saves in I/Os, LLPs, DLPs and space.
  - Screen Commands are used to build screens and placed in order of execution.
  - Any characteristics that are not supported by the terminal are ignored.
  - Modification and maintenance to existing screens is much easier.
- Screen Commands include:
  - Cursor Control - position the cursor on the screen.
  - Screen Editing - permit changes to the appearance of the screen after its been displayed.
  - Field Attribute - define the characteristics of fields.
  - Emphasis - highlight text on the screen.
  - Special - perform miscellaneous things such as displaying of messages, printing of data, etc.
- The SC statement has two formats permitting us to execute Screen Commands from within a run, outside a run, or on the SC statement itself.

Exercises

1. Identify the four components needed to produce one line of screen output in a 4-to-1 output screen.

2. State the difference between 4-to-1 and 5-to-1 output screens.

3. Use the DATA Special Screen Command in conjunction with the SC statement within a run to display any paragraph of text. Use Screen Editing, Cursor positioning, and Emphasis commands in conjunction with another SC statement to manipulate the text on the screen. (i.e. Move lines, insert spaces, etc.

4. Include Screen Commands within a report to create a screen that solicits three fields of information in the following format:

Name of field	Size	Attributes
Name	12	Alpha input only
Street Address	15	Any input
Zip Code	5	Numeric input only

Input fields should be unprotected, but protect the rest of the screen. Place these Screen Commands outside the main RCR and call them with the SC. Add appropriate run statements to the run to accept input and place the input into three variables. Test the input to make sure all three fields contain information. If not, display an error message and redisplay the input screen.

9

Indexing Schemes

Module Objectives

Upon completion of this module, you will be able to discuss the advantages and disadvantages of the Binary Find and Number Generation Indexing Schemes.

Indexing Schemes

- Binary Find
- Number Generator

Binary Find

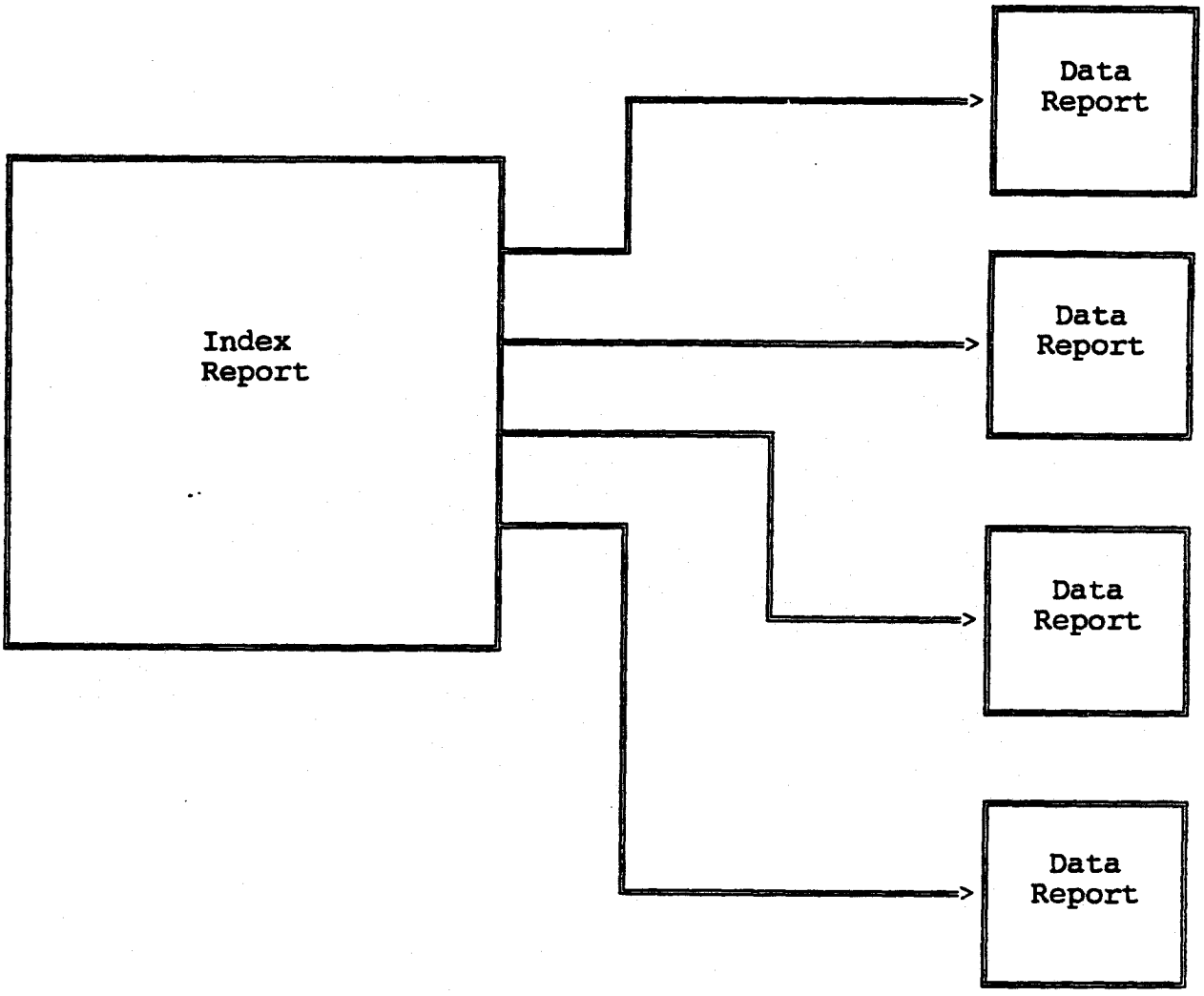
- Finds a single item in a large report or drawer that is presorted on a field or fields being searched.

- Finds data items in a sorted report by sampling the report from the midpoint.

- Creates and maintains an Index Report which points to a series of data reports.

- Useful for large amounts of data which can be grouped into sorted reports.

Binary Find



A-84

V9-1

Sorted Data Reports

- Reports must be sorted before a Binary Find.

- If any reports in the range are empty (headers only), one blank line must be added before building an index.

Sorted Data Reports

. Date 07 Sep 84 17:09:25 Rid 33B 07 Sep 84

Corporate Status Report B0202

* St *Cd	Status Date	By In	Product Type	Serial Number	Produc Cost	Order Number	Cust Code	Produc Plan
Ip	881224	Ls	Blackbox1	200000		84389	amco	881223
Ip	881219	Ls	Blackbox1	200001		84390	amco	881223
Ip	881225	Ls	Blackbox2	200002		84353	intr	881218
Or	870110	Ls	Blackbox4	.		94754	arco	
Or	841102	Ls	Blackbox8	.		96456	arco	
Ip	841216	Ls	Blackbox9	.		82381	feds	841216
Ip	841230	Ls	Blackbox9	299999		84361	ussc	841230

. Date 07 Sep 84 17:09:25 Rid 34B 07 Sep 84

Corporate Status Report B0202

* St *Cd	Status Date	By In	Product Type	Serial Number	Produc Cost	Order Number	Cust Code	Produc Plan
Ip	881224	Ls	Blackbox1	300000		84389	amco	881223
Ip	881219	Ls	Blackbox1	300001		84390	amco	881223
Ip	881225	Ls	Blackbox2	300002		84353	intr	881218
Or	870110	Ls	Blackbox4	.		94754	arco	
Or	841102	Ls	Blackbox8	.		96456	arco	
Ip	841216	Ls	Blackbox9	.		82381	feds	841216
Ip	841230	Ls	Blackbox9	399999		84361	ussc	841230

. Date 07 Sep 84 17:09:25 Rid 35B 07 Sep 84

Corporate Status Report B0202

* St *Cd	Status Date	By In	Product Type	Serial Number	Produc Cost	Order Number	Cust Code	Produc Plan
Ip	881224	Ls	Blackbox1	400000		84389	amco	881223
Ip	881219	Ls	Blackbox1	400001		84390	amco	881223
Ip	881225	Ls	Blackbox2	400002		84353	intr	881218
Or	870110	Ls	Blackbox4	.		94754	arco	
Or	841102	Ls	Blackbox8	.		96456	arco	
Ip	841216	Ls	Blackbox9	.		82381	feds	841216
Ip	841230	Ls	Blackbox9	499999		84361	ussc	841230

V9-2

BFN Statement

Description

The Binary Find (BFN) statement finds items very quickly in sorted reports be sampling the data at midpoint in the report or series of reports.

Format

```
@BFN,c,d[,r,l,lab] o cc ltyp,p [vrpt,vlno] .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report number of the report or result to scan.
l	Line number where the binary find starts its first its first sample.
lab	Label to go to if no finds are made or in case of an error.
o	Options field (see Run Design Reference Manual).
cc	Column-character positions or names of the fields to scan.
ltype	Line type to scan.
p	The item to find.
vrpt	Variable to capture the report number where the find was made.
vlno	Variable to capture the line number where the find was made.

Creating an Index Report

- Contains an individual line containing the report number and the first line of the report.
- Created with the B (Build Index) option on the BFN statement.
- The index must reside in the report directly preceding the first data report.
- Index Report returns as a result.

Creating an Index

```
@BFN,0,b,,,10 BR33-38 'Status Date','Serial No' ,=,K .
```

```
. Date 07 Sep 84  17:09:25   Rid  32B        07 Sep 84
.                                  Corporate Status Report                    B0202

* St   Status      By     Product      Serial      Produc     Order      Cust     Produc
*Cd    Date        In     Type         Number      Cost       Number     Code     Plan
*= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
       33                             200000
       34                             300000
       35                             400000

       .                                .
       .                                .
       .                                .

       38                             700000
    •
```

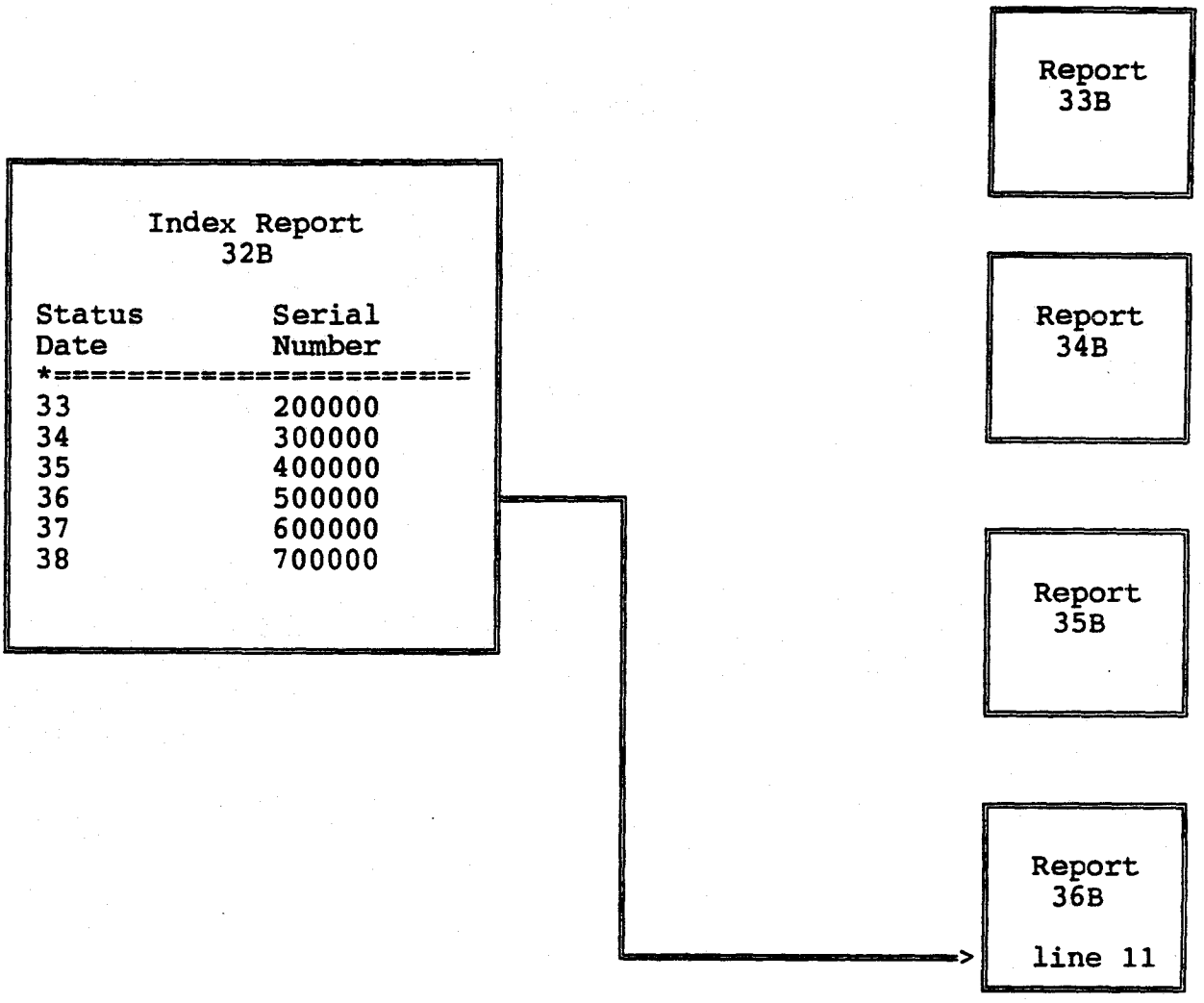
V9-3

Using an Index Report

- Once the Index Report is created, use BFN with the I option.
- The report in which the find is made becomes -0.
- Minimum 2 I/Os!
- Utilizes reserve word STAT1$.

Using an Index Report

```
BFN,0,b,,,10 I32 'Serial Number' ,500005 <rid>i3,<line>i4.
```



V9-4

Binary Find Techniques

- Used to add data in its proper place on the database.

- Used for multiple keys on number generation.

Binary Find Techniques

```
*=================================

@Bfn,0,b,,,10  I32  'Serial Number'  ,222299  <rid>i3, <line>i4 .

     .
     .
     .

@10:If Stat1$ ne 1  gto 99 ; .
@Lok,0,b, <rid> .
@Rsl,0,b, <rid> .
@Ln+,0,b, <rid>, <line>,1   Inc  <line> .
@Wrt,0,b, <rid>, <line>  'StCd','Status Date','Product Type',\
'Serial Number'  ,ip,880907,greenbox1,222299 .
@Rep,0,b, <rid> .
@Ulk .
```

Number Generator

- Data placed in the database randomly based on input.
- Data is evenly distributed, but may be costly to retrieve.
- Database must remain the same size!

Single Key

@LDV,n

- Always generates the same number given the same input.
- Input or data does not have to be numeric.

Single Key

LDV,n random report # generated = Input Data, Report Range

```
*=================================================
@3:Brk  Ldv,w  <soe>h1 = soe$.

                Add New Data  <soe>

                Find Old Data  <soe>

@Brk Out,-0,2,2,1,1,y,n .
@Ldv,w  <soev>i4 = soev$.
@Chg input$  <name>h6 .
@Ldv,n  <report>i3 = <name>,163-165 .
@Gto <soev> .
@1:Lok,0,a,<report>  Ln+,0,a,<report>,5,1 .
@Wrl,0,a,<report>,6,y  2-6  ,<name>  Gto 3 .
@2:Fnd,0,a,<report>,6,y  2-6  ,<name>  ,<line>i4 .
@Dsp,0,a,<report>,<line>,4 .
@Gto 3 .
@99 .

          Data not found

@Brk out,-0,2,1,1,1,y,n .
@Gto 3 .
```

V9-6

Multiple Keys

- @LDV,n can provide a solution for fast access to a large database.
- Data is placed into a two-dimensional scheme of reports.
- Maximizes mass storage utilization.

Multiple Keys Example

Sales Dept. - 400,000 customer records maintained on external history files.

Goal: to be able to bring a single file online and be able to efficiently access records by either customer number or item number. Data is to be spread evenly across 200 reports.

Placement Strategy:

```
Step 1:  Ldv,n cust-key = cust-no,1-20
Step 2:  Ldv,n item-key = item-o,1-10
Step 3:  cust-key * 10 + item-key = report#
Step 4:  sort reports by cust-no
```

Item Search Strategy:

```
Step 1:  Ldv,n cust-key = cust-no,1-20
Step 2:  Startrid = cust-key * 10
         Endrid = (cust-key * 10) + 10
Step 3:  Bfn osrStartrid-Endrid 'customer#' ,cust-no
```

Customer Search Strategy:

```
Step 1:  Ldv,n item-key = item-no,1-10
Step 2:  Range = item-key + 10, item-key + 20, ...
                                              item-key + 100
Step 3:  Srh,m,t dhrrida,ridb,ridc,,,, 'item' ,item-no
```

Summary

- Binary Find permits the building of an Index report with the B option and later accessing that Index Report with the I option.
- The Index Report:
  - assumes all data is sorted inter- and intra-database.
  - resides in the report directly preceding first data report.
  - returns as a result therefore must be saved.
- Disadvantages of Binary Find:
  - data must be kept sorted.
  - only one sort key is permitted.
- The Number Generator permits MAPPER to randomly generate numbers and place the data out on the database assuming the input values remain the same. This is accomplished by the LDV,n statement. The data is spread evenly but the disadvantages:
  - Database must remain the same.
  - Manual searches are extremely costly.

Exercises

Discuss the advantages and disadvantages of each of the following data indexing schemes:

- Binary Find -

- Number Generation -

10

Interfacing with UNIX

Module Objectives

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

1. Identify two types of UNIX files.
2. Identify basic UNIX file structure.
3. Define UNIX filename syntax.
4. Define and code @FIL.
5. Define and code @RET.
6. Define and code @UNX.
7. Define and code @XUN.

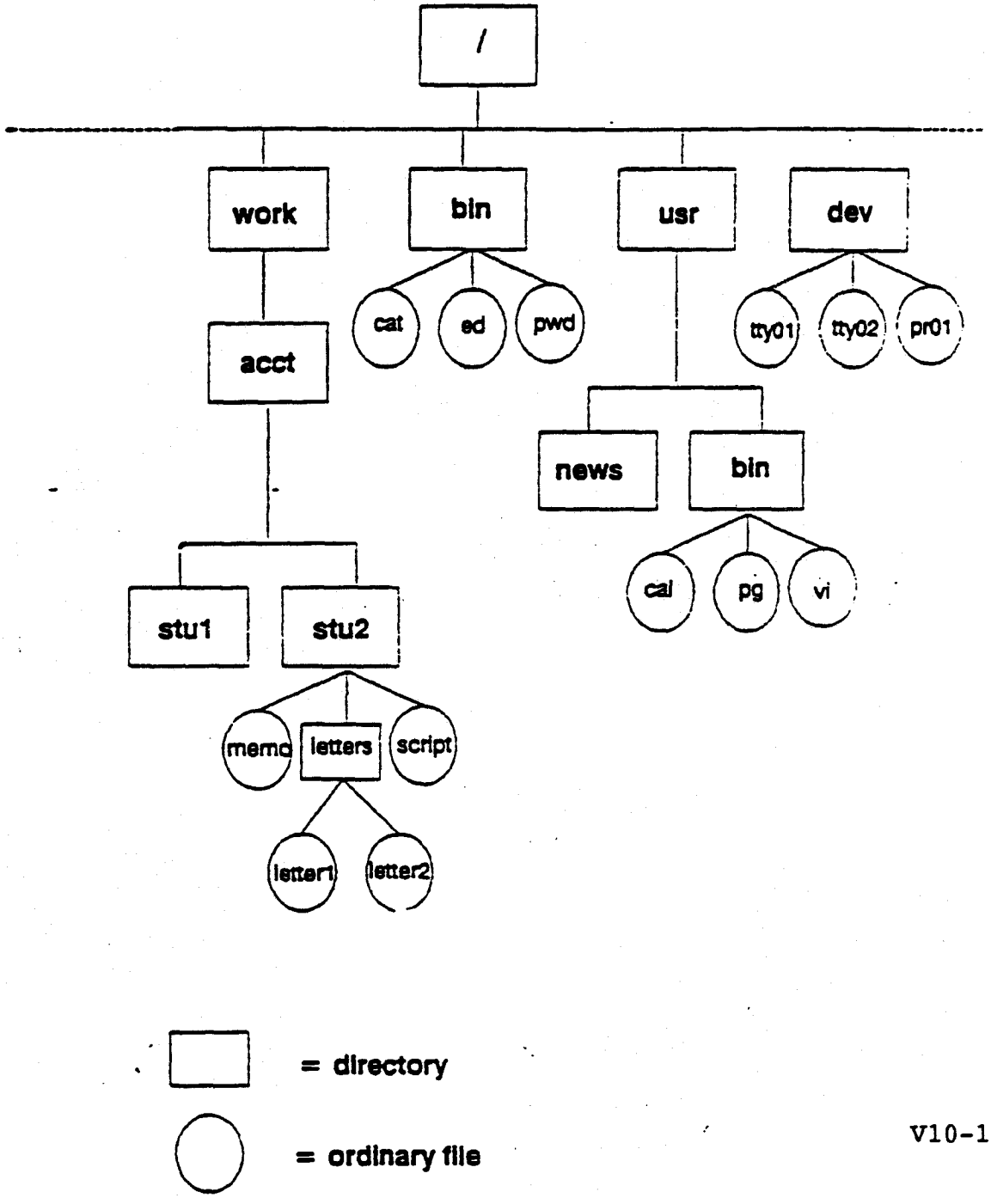
UNIX Overview

- Operating System.
- Application programs (tools or utilities).
- Over 200 commands.
- Written in C Language.

Types of UNIX FIles

- Directory files
  - Pointer to other files or directories.
- Ordinary files
  - Collection of data, text, or other information.

Sample UNIX File Structure



V10-1

UNIX Filename Syntax

/directory/directory/filename ...

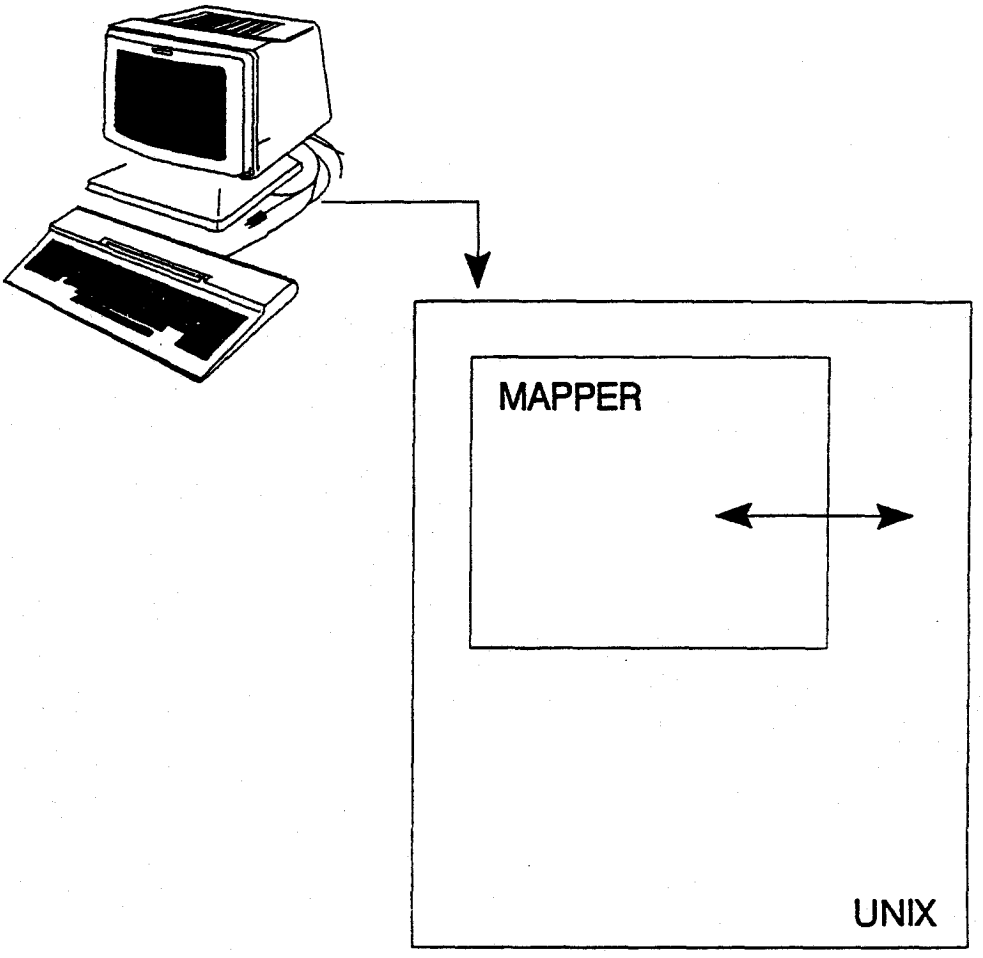
- Always starts with system supplied directory ... root.
- Directories that fall under "root" are subdirectories.
- Root, subdirectories, and filename together are referred to as "pathname."

MAPPER and UNIX

- MAPPER runs as an application under UNIX.

- MAPPER commands can:
  - create UNIX files (FIL).
  - retrieve UNIX files as MAPPER results (RET).
  - execute UNIX commands (UNX).
  - enter the UNIX system (XUN).

MAPPER and UNIX

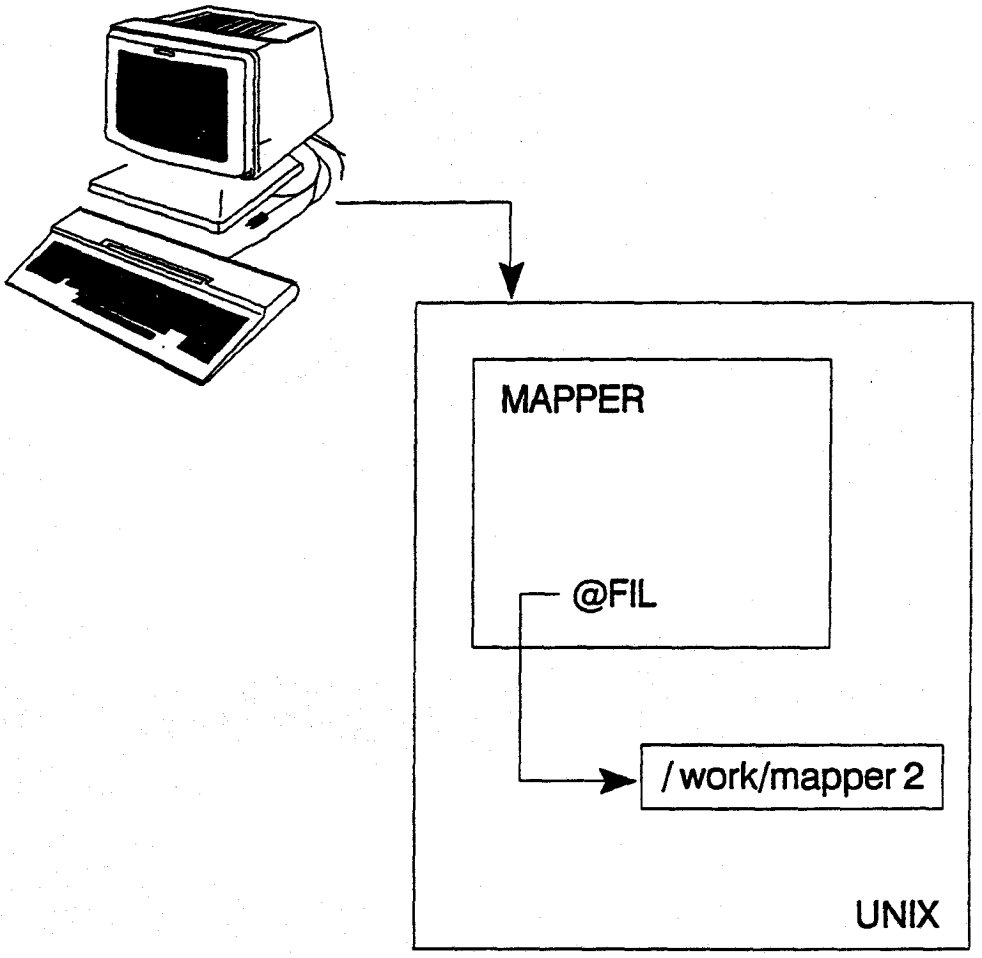


V10-2

FIL
(Create File)

- Creates a UNIX file using data from a MAPPER report or result.
- MAPPER or UNIX format.
- Utilizes Data Control Commands.

FIL
(Create File)



V10-3

FIL Statement

Description

The Create File (FIL) statement creates a UNIX file in the UNIX operating environment, using the data from a MAPPER report or result.

Format

```
@FIL,c,d,r[,mapperf?,hdrs?,lab] pthfn .
```

Fields

Field	Description
c,d,r	Cabinet, drawer, and report to be copied.
mapperf?	Create the report in MAPPER format, Y or N. Default = N, creates the report in UNIX format.
hdrs?	Place report headers into the file, Y or N. Default = N.
lab	Label to go to if the report or drawer does not exist.
pthfn	Full pathname and filename of the UNIX file into which to place this report (maximum length of the pathname and filename is 70 characters; maximum length of the file name is 14 characters.

Example

```
@fil,0,c,4,n,y /master/inventory .
```

Create a UNIX file called inventory in the master directory using the data in RID 4C, cabinet 0, including report headers.

Data Control Commands

- MAPPER commands that control the format of the data in the files being created.
- Entered on any line, beginning in column 1.
- Take effect from place of entry in the report.

Data Control Commands

```
*=========================================
@cmd
$DATA$
@cmd
@cmd
```

V10-4

Data Control Commands

$CLRT$ - translates tab characters to spaces.

$DATA$ - suspends any data translation by the $TRNA$ command; uses original data.

$DCML$ - Deletes all asterisk-type lines except header lines.

$DFFL$ - Deletes all period-type lines except header lines.

$ICML$ - Includes all asterisk-type lines.

$IFFL$ - Includes all period-type lines.

$TABA$ - Translates tab characters to the character represented by ASCII octal code nnn, or to the character y.

Format: $TABA$ {nnn|'y'}

where: nnn is the ASCII octal code of the character; y is the character (enclosed in apostrophes).

$TRNA$ - Translates character x to the character represented by ASCII code nnn (octal) or to the character y, or re-establishes the translation previously suspended by $DATA$.

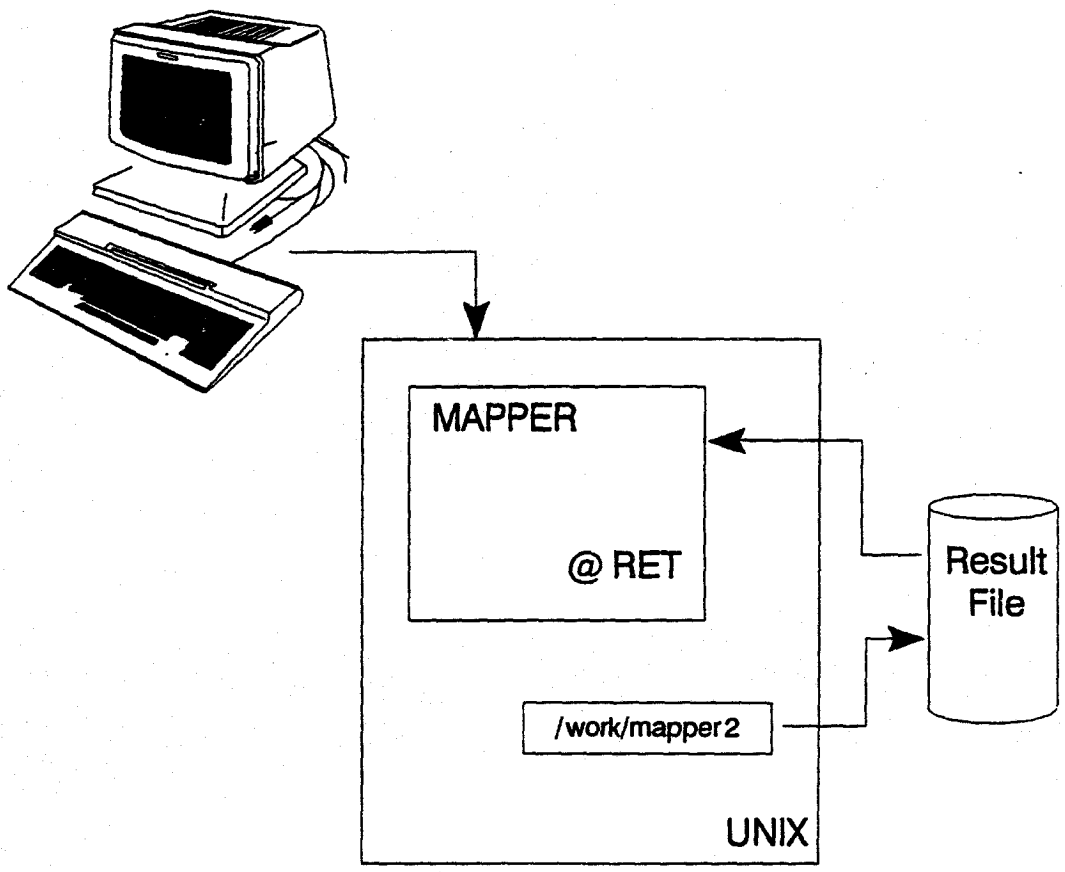
Format: $TRNA$ {x,nnn|x,'y'...}

where: x,nnn translates character x to the character nnn in the ASCII character set; x,'y' translates character x to the literal 'y'.

RET
(Retrieve File)

- Retrieves a UNIX file as a result.
- Result must be saved.

RET
(Retrieve File)



V10-5

RET Statement

Description

The Retrieve File (RET) statement retrieves a UNIX file as a result.

Format

```
@RET,c,d[,mapperf?,hdrs?,lab] pthfn .
```

Fields

Field	Description
c,d	Cabinet and drawer into which the report should be placed.
mapperf?	File is in MAPPER format, Y or N. Default = N.
hdrs?	Add headers from the receiving drawer to the result, Y or N. Default = N.
lab	Label to go to if the file is not found.
pthfn	Full pathname and filename of the UNIX file to retrieve.

Example

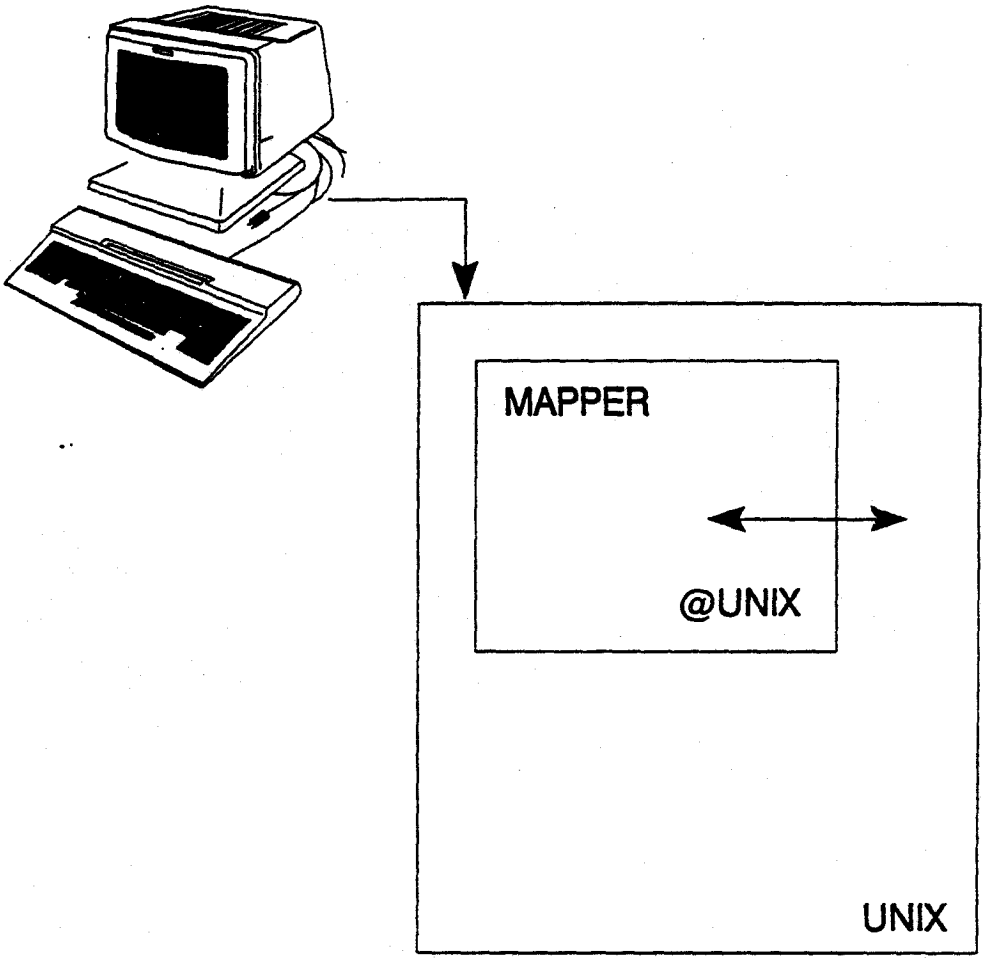
```
@ret,0,c,n,n /master/inventory .
```

The file /master/inventory, which is not in MAPPER format, is created as a result in cabinet 0, drawer c, with no headers.

UNX
(Unisys System V Interface)

- Interfaces with UNIX operating system.
- Allows the execution of UNIX commands.
- UNIX commands, arguments, and filenames are case sensitive.

UNX
(Unisys System V Interface)



V10-6

UNX Statement

Description

The Unisys System V Interface (UNX) statement interfaces with the UNIX operating system and allows the execution of UNIX commands.

Format

```
@UNX[,c,d,login,pw,lab] o cmd [args] .
```

Fields

Field	Description
c,d	Cabinet and drawer into which the result should be returned. (-f option)
login	UNIX login. Default = current run user login.
pw	Password for the login.
lab	Label to go to if the UNIX command fails.
o	Options field. (next page). Must be entered as lower case letters. If no options are used, mark position with ' '.
cmd	UNIX command to be performed. If not command is specified, mark position with ' '.
args	Arguments that may be passed to the UNIX command. If used, the entire command and argument must be enclosed in apostrophes.

UNX Statement

Options:

- c Clears screen after UNX statement has finished processing.

- f Output of the command is returned as a result in the designated drawer.

- p Execute the .profile of the user before executing the UNIX command.

- w After command execution and display of output, press PF1 to resume the run.

Example:

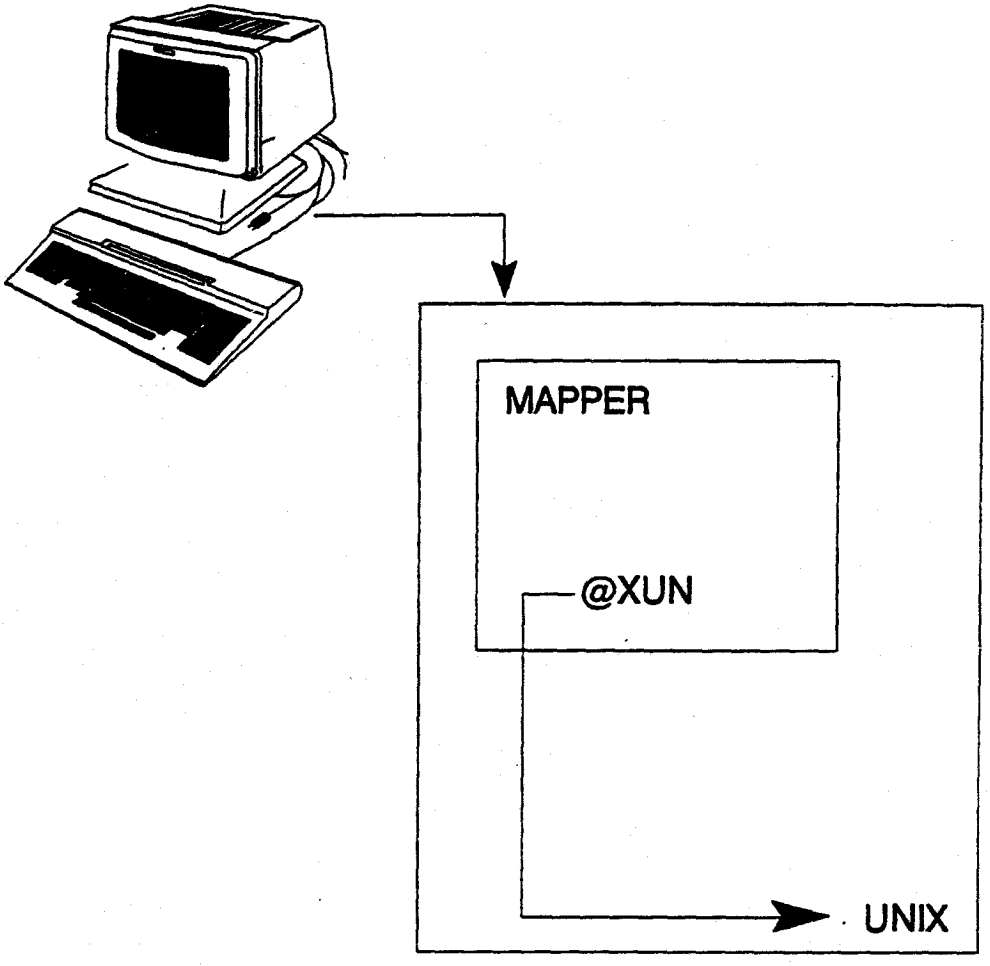
```
@unx,0,c -f '/bin/ls -l /usr/work/smith' .
```

List the contents of the "/usr/work/smith" directory and place the output in cabinet 0, drawer B result.

XUN
(Exit MAPPER System)

- Terminates the active MAPPER run.
- Signs off and terminates MAPPER for that station.
- Exits to UNIX.
- Format: @XUN .

XUN
(Exit MAPPER System)



V10-7

Summary

- UNIX is an operating system that U Series MAPPER interfaces with.
- UNIX files can be directory files or ordinary files.
- The FIL statement allows MAPPER to create a UNIX file in the UNIX environment, using the data from a MAPPER report or result.
- Data Control commands can control the format of data in files being created.
- The RET statement retrieves a UNIX file as a MAPPER result.
- The UNX statement allows the execution of UNIX commands.
- The XUN statement terminates the MAPPER run and exits to UNIX.

Exercises

Matching: Letters may be used more than once.

1. ____ Pointer to other files or directories.
2. ____ Allows MAPPER to create a UNIX File in the UNIX environment, using the data from a MAPPER report or result.
3. ____ Terminates the MAPPER run and exits to UNIX.
4. ____ Collection of data, text, or other information.
5. ____ System-supplied directory.
6. ____ Retrieves a UNIX file as a MAPPER result.
7. ____ Root, subdirectories, and filename.
8. ____ Allows the execution of UNIX commands.
9. ____ Directories that fall under the system supplied directory.

A. @XUN
B. @RET
C. #UNX
D. @FIL
E. Directory files
F. Ordinary files
G. Root Directory
H. Pathname
I. Subdirectories

A

KSA Concepts

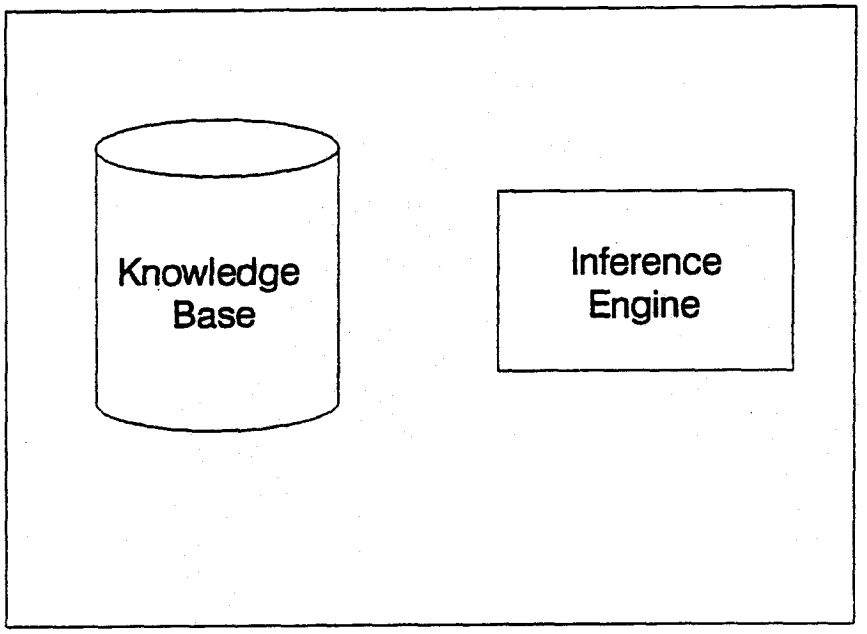
Topics:

- Knowledge-Based Systems
- KSA (Knowledge Systems Access)
- KSA and MAPPER
- KSA Components
- KSA Core Module
- Schema Definition Module
- Runtime Module
- KSA Access

Knowledge-Based Systems

- Knowledge Base
- Inference Engine

Knowledge-Based Systems



VA-1

KSA
Knowledge System Access

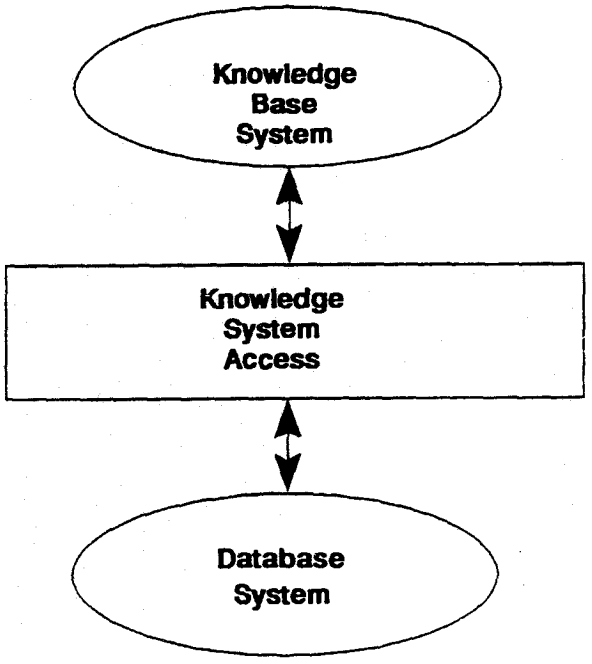
- Uses fourth generation language product (MAPPER, LINC, ALLY) commands to transfer data in files to the knowledge system and back.

- Adds knowledge processing to existing MAPPER applications.

- A module of KES II.

- Written in C Language.

KSA
Knowledge System Access



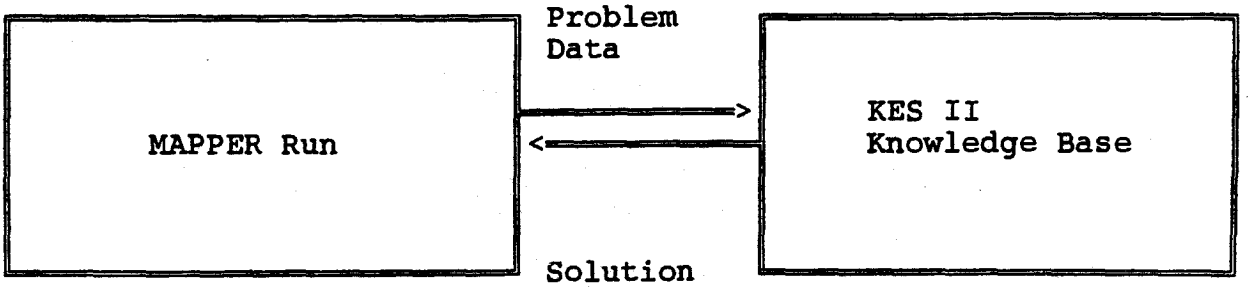
VA-2

KSA and MAPPER

- A MAPPER application, by communicating to a KES II knowledge-based system, can access the reasoning and problem solving techniques reserved for human experts.

- No modification to the MAPPER database is necessary.

KSA and MAPPER



KSA and MAPPER
Overview

- The run files the data which contains information about a problem for processing.

- The run transfers control to KSA.

- KSA communicates with the KES II Knowledge Base, activating its inferencing mechanism, providing it data, and obtaining the solution to the problem.

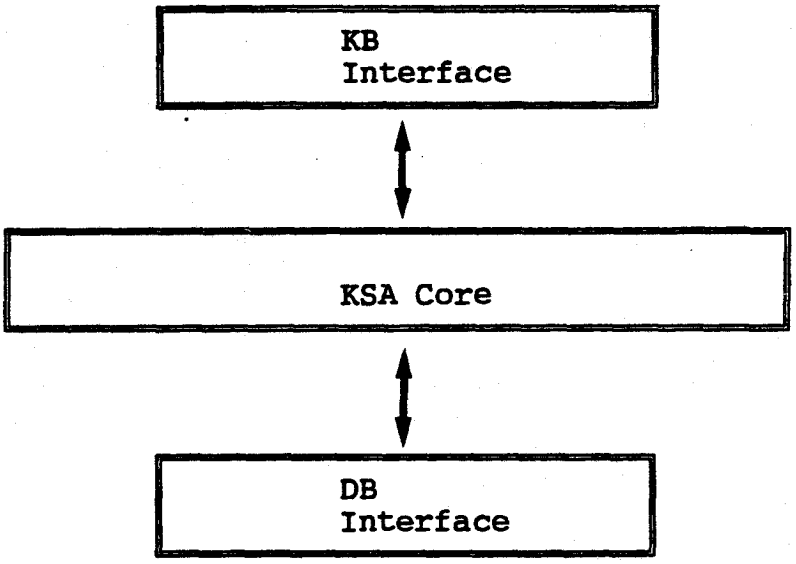
- The solution is retrieved by the MAPPER run, which processes the solution and continues executing.

KSA Components

- Knowledge Based Interface for KES II.
- Database Interface for U Series MAPPER.
- KSA Core Module.

KSA Components

KSA



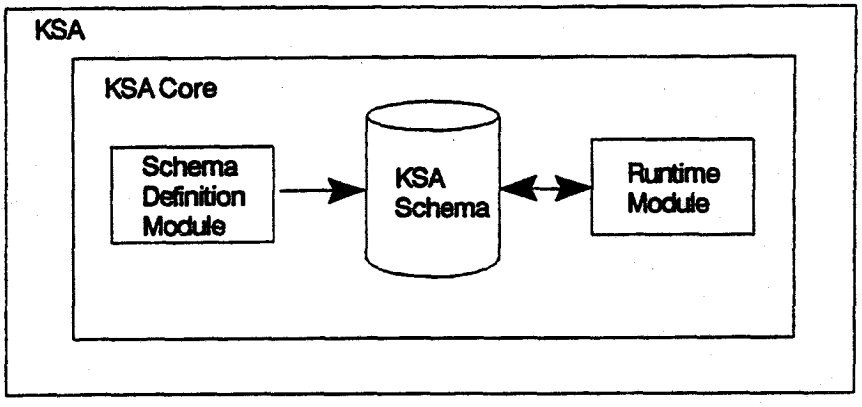
VA-4

KSA Core Module

- First of three KSA components.

- Contains:
  - Schema Definition Module
  - KSA Schema
  - Runtime Module

KSA Core Module



Schema Definition Module

- Utilizes MAPPER data descriptions to associate MAPPER column names to the appropriate attribute with the KES II PS Knowledge Base.

- Can design the:
  - KES II PS Knowledge Base against an existing MAPPER database.
  - MAPPER database against an existing KES II PS Knowledge Base.
  - MAPPER database in conjunction with the KES II PS Knowledge Base.

- Result is KSA Schema.

- Data descriptions may be:
  - RID0 descriptions
  - Experimental RID0 descriptions
  - Header portion of a data RID.

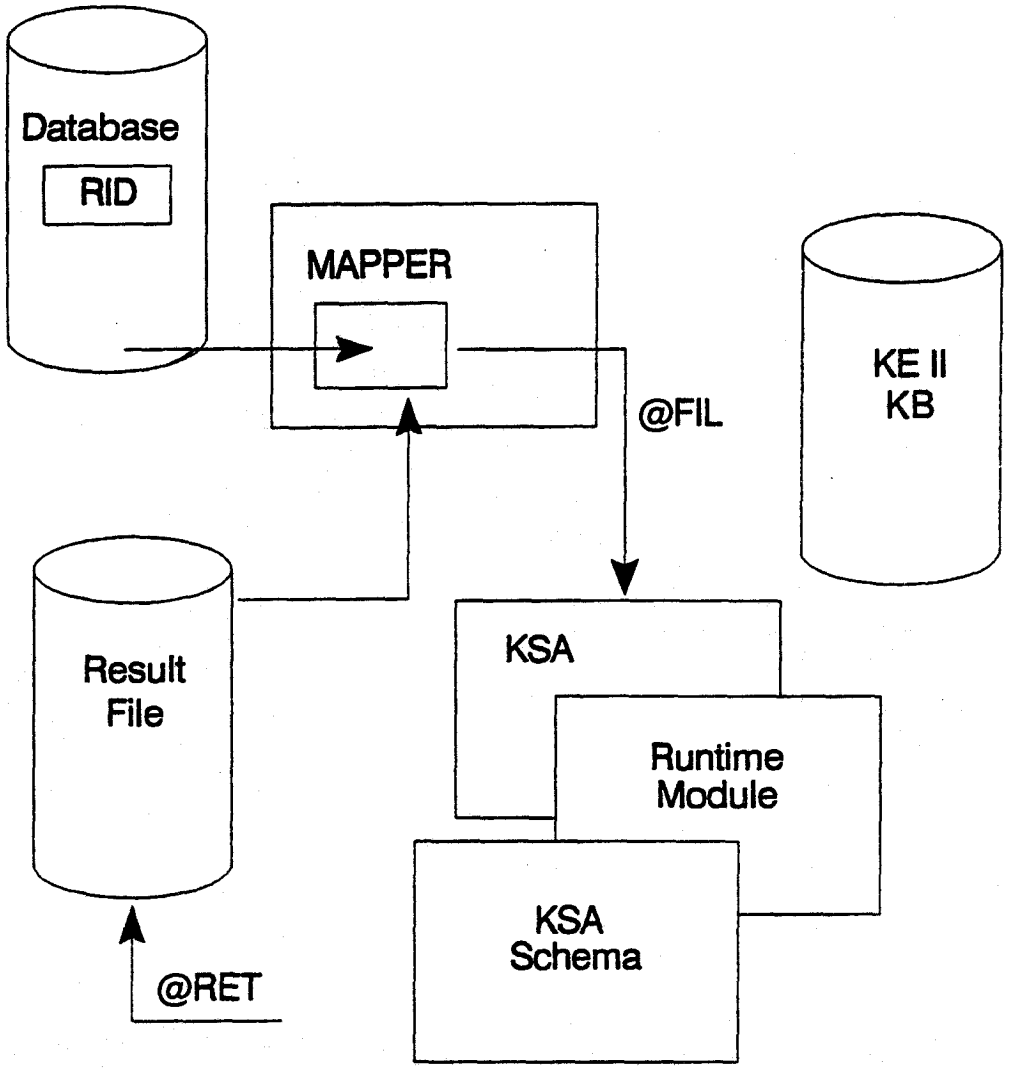
Runtime Module

- Utilizes the KSA Schema.
- Receives data from a MAPPER cabinet.
- Delivers data to the KES II Knowledge Base when required.
- Receives the result data from the KES II Knowledge Base.
- Returns the requested result data to MAPPER.

Accessing KSA

- Three basic steps:
  - Build the KES II Knowledge Base.
  - Define the Schema Definition Module.
  - Execute KSA.
- Utilize MAPPER commands:
  - FIL (Create File)
  - UNX (UNIX Interface)
  - RET (Retrieve File)

Accessing KSA



VA-6

Accessing KSA
Building the Knowledge Base

- Usually performed by a KES II expert.
- Knowledge Base file name will have a ".kb" extension.
- Must be "parsed" with the appropriate KES II parser; the file name will have a ".pkb" extension.

Accessing KSA
Defining the Schema Definition Module

- Display defined MAPPER RID0 reports.
  - A maximum of eight may be used; all must reside in the same MAPPER cabinet.
- Send to UNIX, with the @FIL statement.
  - The RID0s must have ".rid" filename extentions.
  - If more than one RID0 is sent, they must be concatenated in the UNIX environment with the UNIX CAT command.
- Use the @UNX statement to enter the UNIX environment.
  - Execute def to build the KSA Schema, which will have a ".sch" extension).

    def.exe 1 <rid0 description file name>

Executing KSA

- MAPPER data (input tables to KSA) must have .dat extensions.

- In UNIX, execute ksa, which utilizes the KSA Schema and the KES II Knowledge Base.

  ksa <#> <schema> <pkb> [<input table>]

ksa	call to the actual KSA program.
<#>	number of arguments to follow.
<schema>	name of the schema file to be used.
<pkb>	name of parsed knowledge base to be used.
<input table>	name(s) of input table(s) to be used.

- Result files are output of the execution.

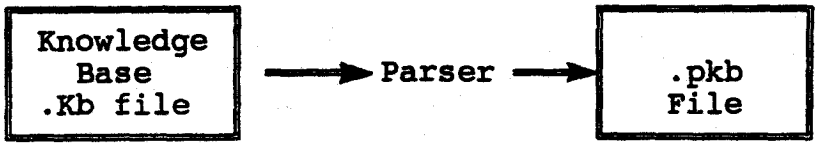
  map_kes_0.res
  map_kes_1.res
  map_kes_2.res
  etc.

- Use @RET to retrieve back into MAPPER.

Accessing KSA

Knowledge Base Building

Step One

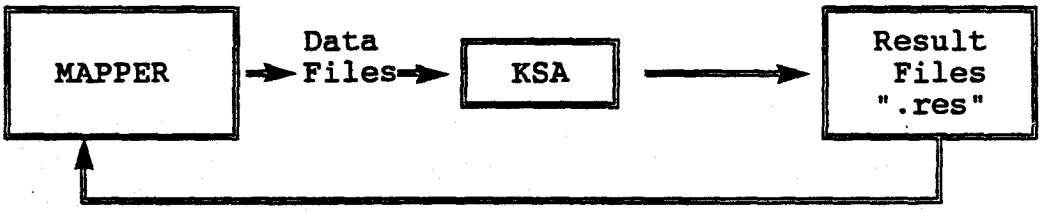


Schema Definition

Step Two

KSA Execution

Step Three



VA-7

Summary

Knowledge-based systems are made up of a Knowledge Base and an Inference Engine.

KSA (Knowledge System Access) adds knowledge processing to existing applications, using database management or 4GL products.

KSA is comprised of a KSA core module, a knowledge-based interface for KES II PS, and a database interface for U Series MAPPER.

The MAPPER command @FIL, @UNX, and @RET are used with KSA.

B

Networking

Topics:

- Networks
- NET (Network Sign On) @NET,,B
- NWR (Network Write) @NWR,ϕ,C4,ϕ,B3 (A → B)
- RTN (Return Remote)
- NRD (Network Read) @NRD,ϕ,B,2,ϕ,C,4 (B → A; issue, recv)
- RRN (Remote Run)
- NRN (Network Run)
- NOF (Network Off)

o NRN (Remote RUN Command)

eg @NRN "@RSL,ϕ,B,2 ^NM-1".

@NRN "@TOT,-1" (note: you lose -ϕ between NRN calls)

o NRM (Remote Command)

eg - FCC

@NRM "Fcc"

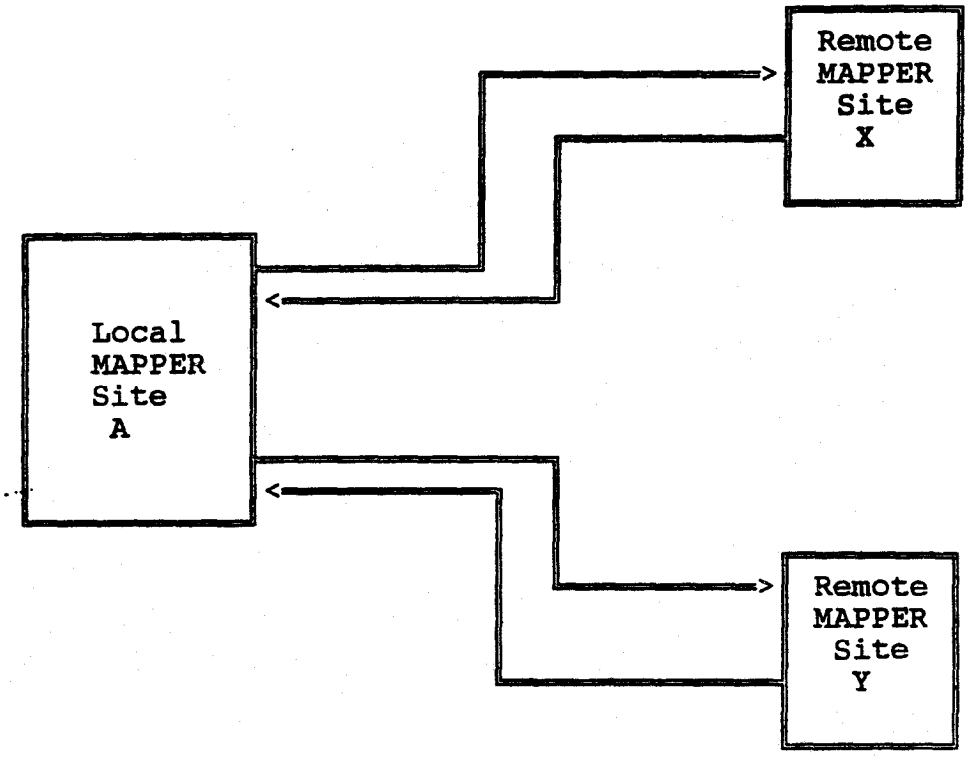
CS B ≠ @NET,,B

@NRM "^"

@NOF

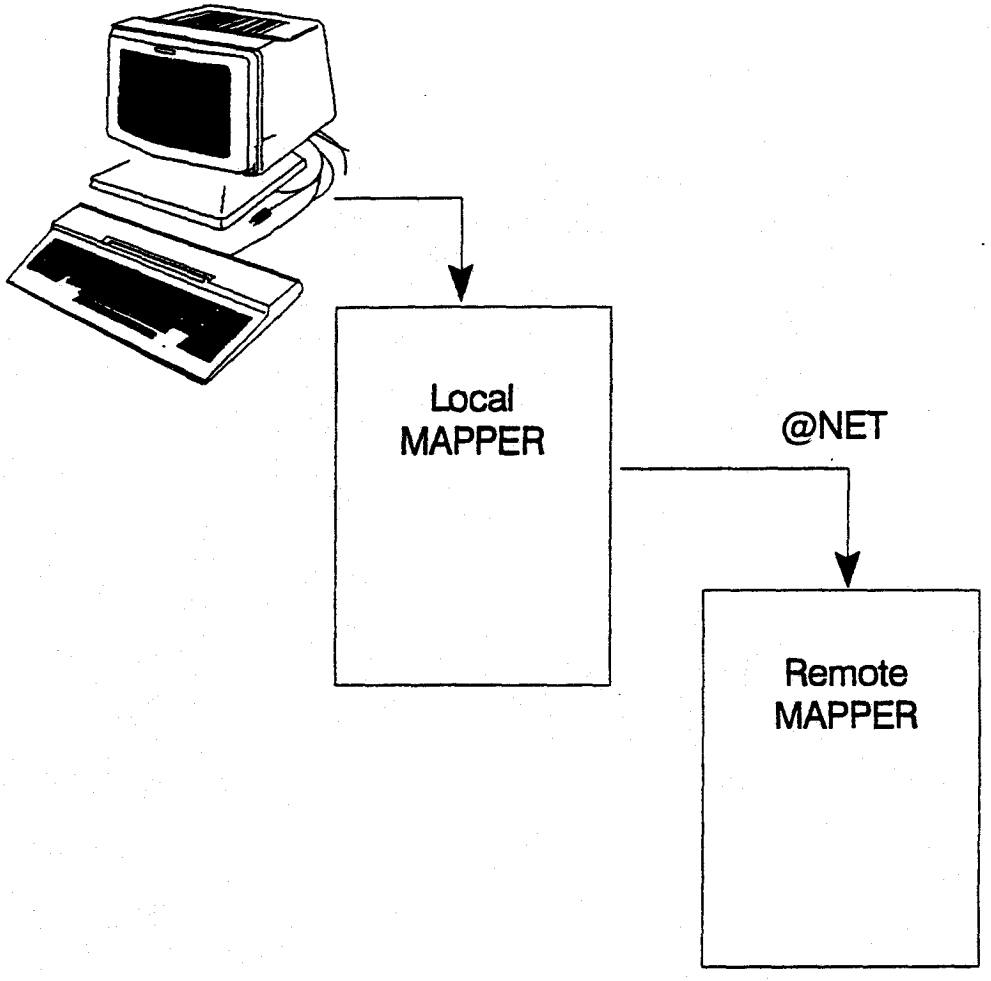
^^

Networking



VB-1

NET Statement



VB-2

NET Statement

Description

The NET (Network Sign On) statement signs on to the remote MAPPER system, from the calling (local) MAPPER system.

Format

```
@NET,net-id[,site-id,rmu,rmd,rmpw,trnrpt,mtr,lab].
```

Fields

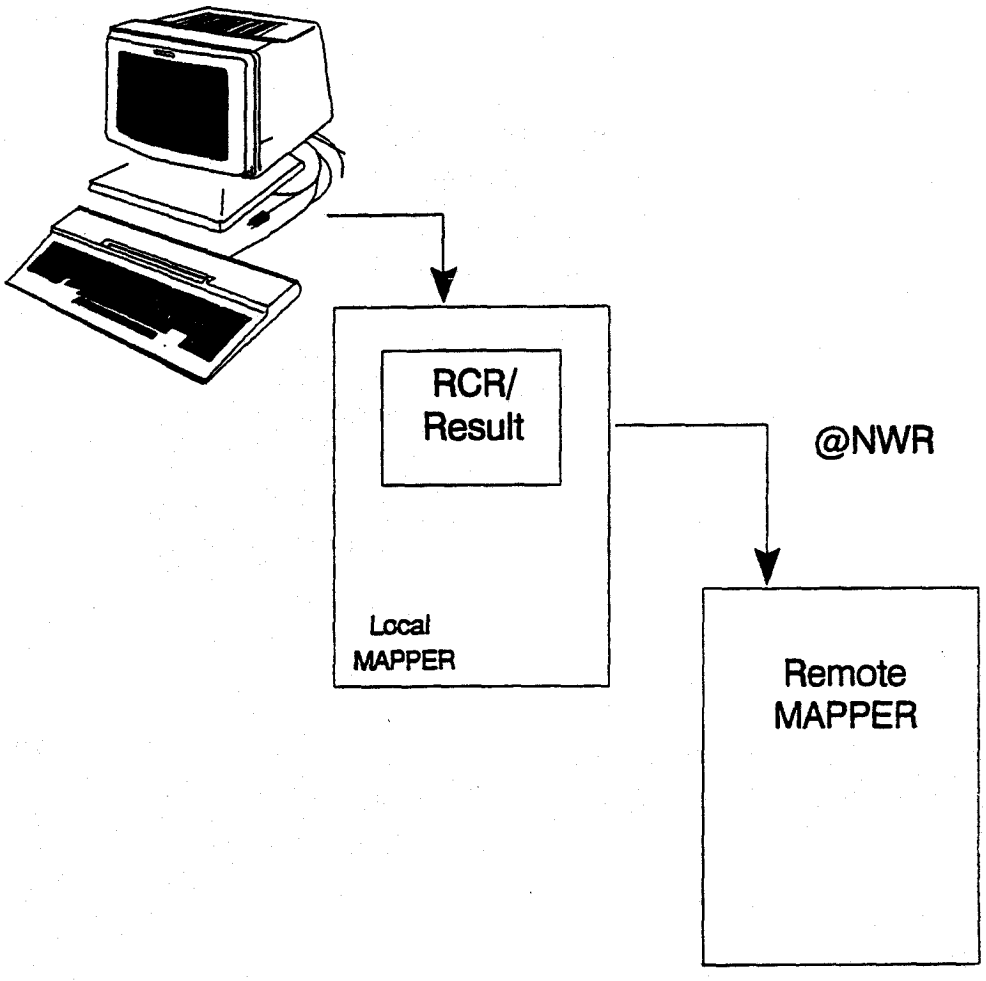
Field	Description
net-id	Network identifier of the host system.
site-id	Site identifier of the remote MAPPER system.
rmu	User-id registered on the remote MAPPER system.
rmd	User department number for the remote MAPPER sign on.
rmpw	User-id password for the remote MAPPER sign on.
trnrpt	Report that contains a translation table used to translate characters between the local and remote MAPPER sites.
mtr	Monitor transferred data, Y or N. Default = N.
lab	Label to go to if the connection fails.

Example

```
@net,sys21,x,smith,8,smith,,y .
```

Sign on to remote host sys21, site identifier X, with sign-on smith,8,smith, and monitor the signon to the remote site.

NWR Statement



VB-3

NWR Statement

Description

The NWR (Network Write) statement enables a user to send a report or result from a local MAPPER system to a remote MAPPER system.

Format

```
@NWR,ic,id,ir,rc,rd,rr[,lab vmsg] .
```

Fields

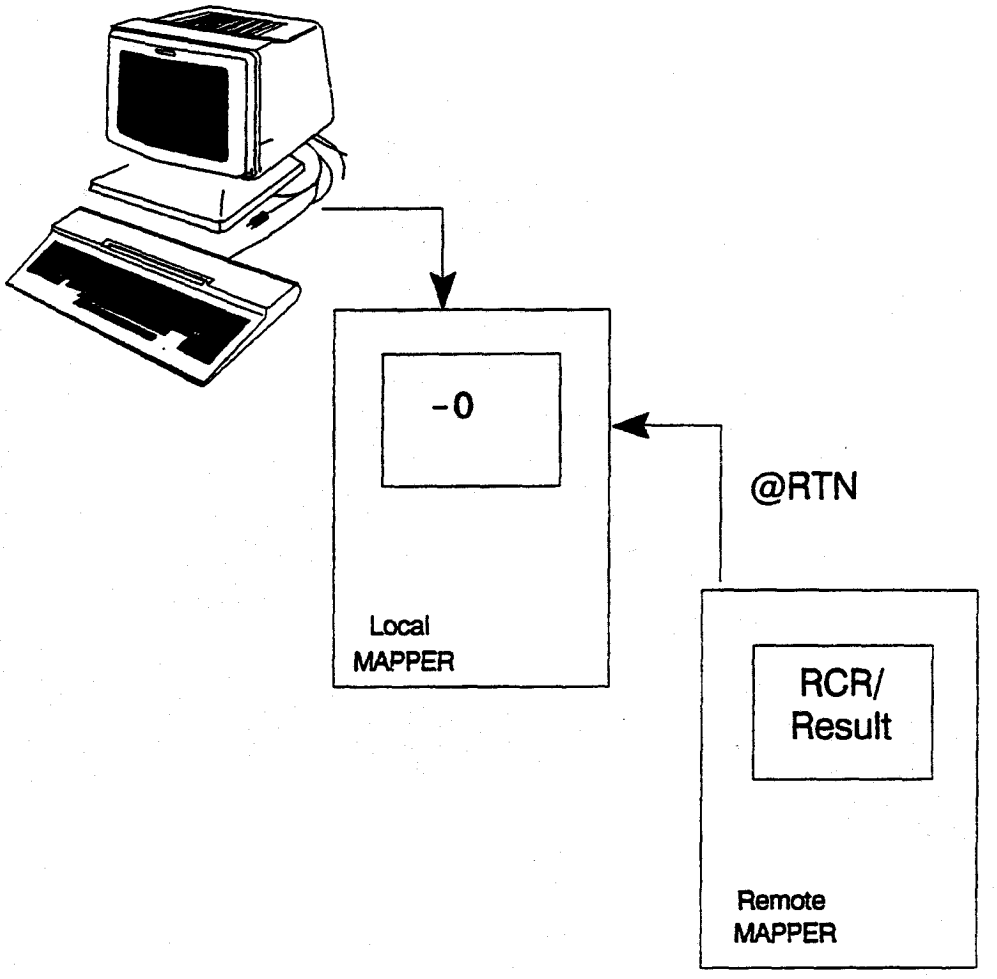
Field	Description
ic,id,ir	Cabinet, drawer, and report number of the issuing report on the local MAPPER system.
rc,rd,rr	Cabinet, drawer, and report number of the receiving report on the remote MAPPER system.
lab	Label to go to if an error occurs.
vmsg	Variabled to contain an 80-character error message if the label is taken.

Example

```
@nwr,0,c,4,0,c,3,010 v1s80 .
```

Write RID 4C, cabinet 0, to RID 3C, cabinet 0, on the remote MAPPER system. (If an error occurs, go to label 10)

RTN Statement



VB-4

RTN Statement

Description

The RTN (Return Remote) statement returns a report or result from a remote MAPPER site to the local (calling) MAPPER software site as a -0 result. The result is received by the calling user as a message.

Format

```
@RTN,rc,rd,rr .
```

Fields

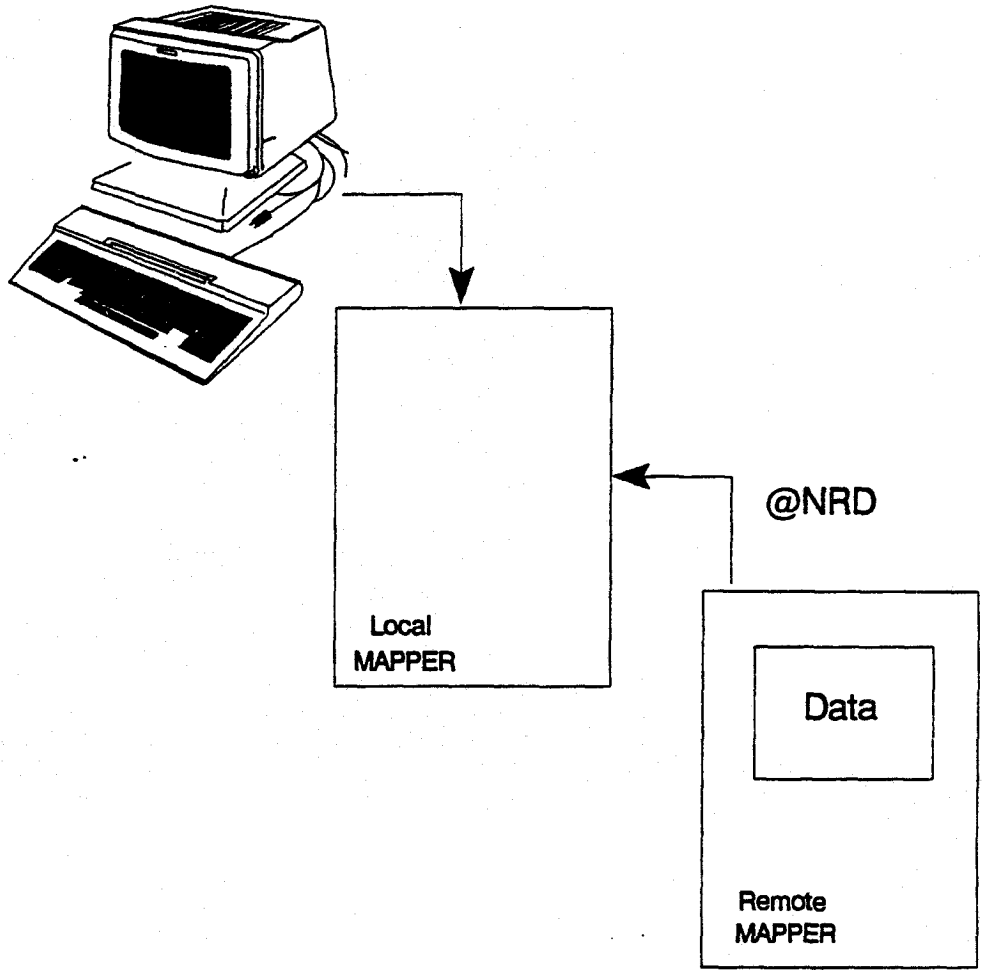
Field	Description
rc,rd,rr	Remote cabinet,, drawer, and report to send.

Example

```
@rtn,0,c,4 .
```

Send RID 4C, cabinet 0, to the user that executed the remote run statement.

NRD Statement



VB-5

NRD Statement

Description

The NRD (Network Read) statement enables the user to read and return data from a remote MAPPER system to the local MAPPER system.

Format

```
@NRD,ic,id,ir,rc,rd,rr[,lab vmsg] .
```

Fields

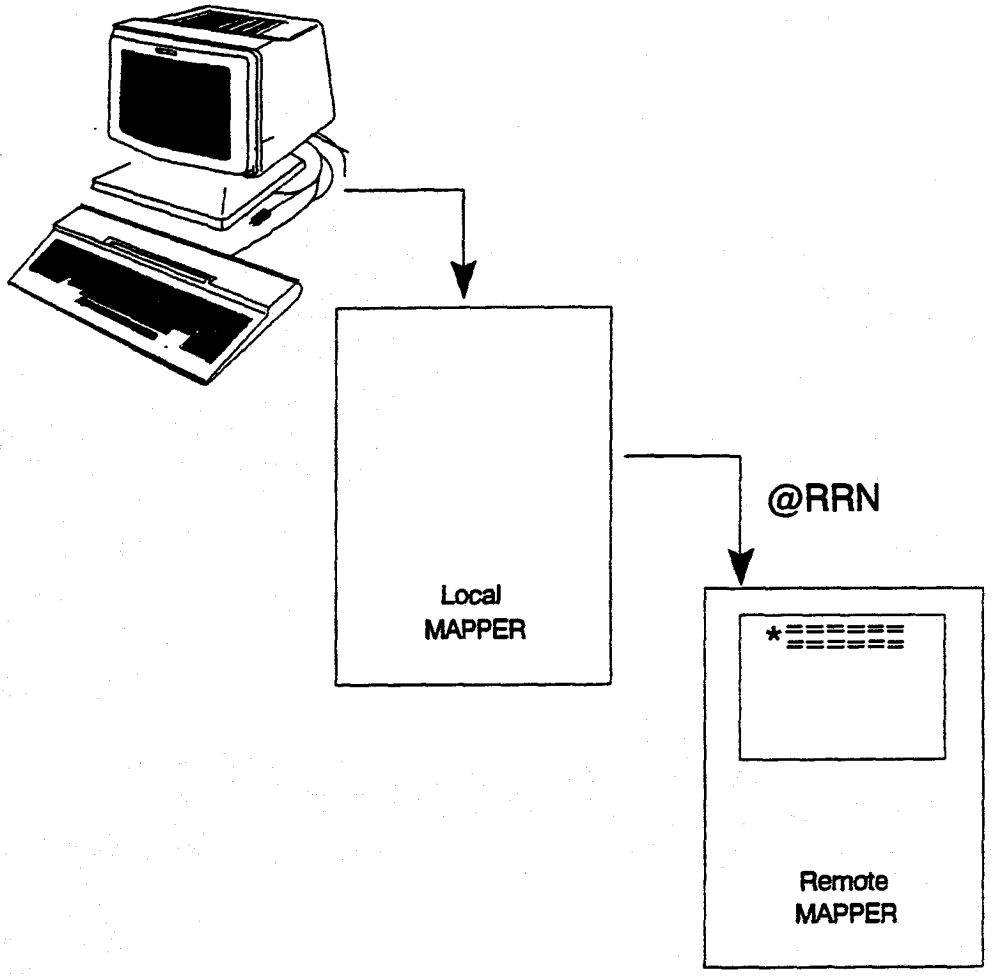
Field	Description
ic,id,ir	Cabinet, drawer, and report number of the issuing report on the remote MAPPER system.
rc,rd,rr	Cabinet, drawer, and report number of the receiving report on the local MAPPER system.
lab	Label to go to if an error occurs.
vmsg	Variabled to contain an 80-character error message if the label is taken.

Example

```
@    nrd,0,c,4,0,c,5,010 v1s80 .
@    dsp,-0 .
 .
 other processing
 .
@010:gto end .
```

Read RID 4C, cabinet 0, on the remote MAPPER system and place the data into RID 5C, cabinet 0, on the local MAPPER system (go to label 10 in case of error).

RRN Statement



RRN Statement

Description

The Remote Run (RRN) statement starts a run at another MAPPER site.

Format

```
@RRN[,lc,ldr,lr] run[,vld] rms,rmu,rmd
[,rmpw,rmc,rmdr] .
```

Fields

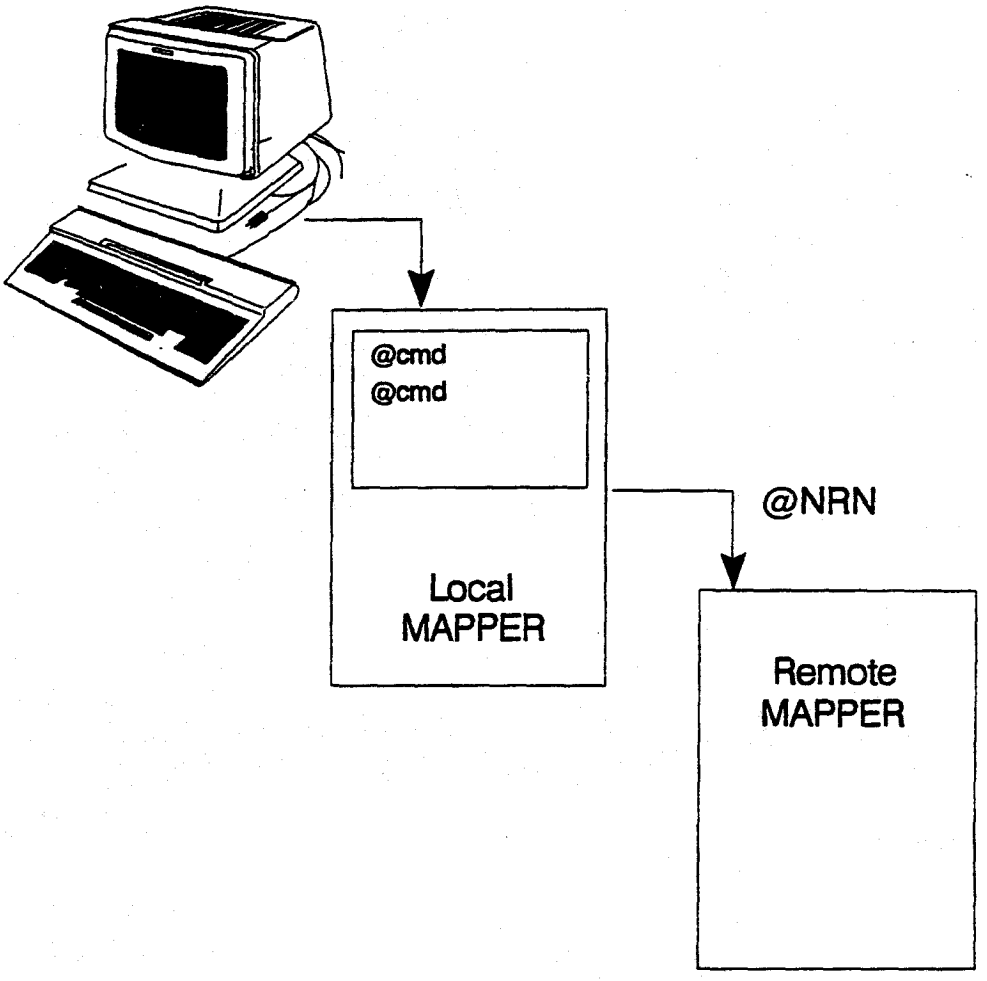
Field	Description
lc,ldr,lr	Local cabinet, drawer, and report.
run	Name of the run to execute at the remote site (registered and resident at remote site).
vld	Variables, literal data, reserved words, or any combination of these, to transfer to the remote site for use as variables by the remote run with INPUT$.
rms	Remote MAPPER site letter (A-Z).
rmu	User-id registered at the remote site.
rmd	User sign-on department number registered at the remote site.
rmpw	User sign-on password registered at the remote site.
rmc,rmdr	Remote cabinet and drawer.

Example

```
@rrn master,data5 c,smith,8 .
```

Start the run MASTER at remote site letter C, transfer data DATA5 (picked up by INPUT$ at the remote site), and use the SMITH,8 sign on at the remote site.

NRN Statement



VB-7

NRN Statement

Description

The Network Run (NRN) statement allows a local MAPPER system to pass run statements to a remote MAPPER system to be executed.

Format

```
@NRN[,lab] "run statements" [vmsg] .
```

Fields

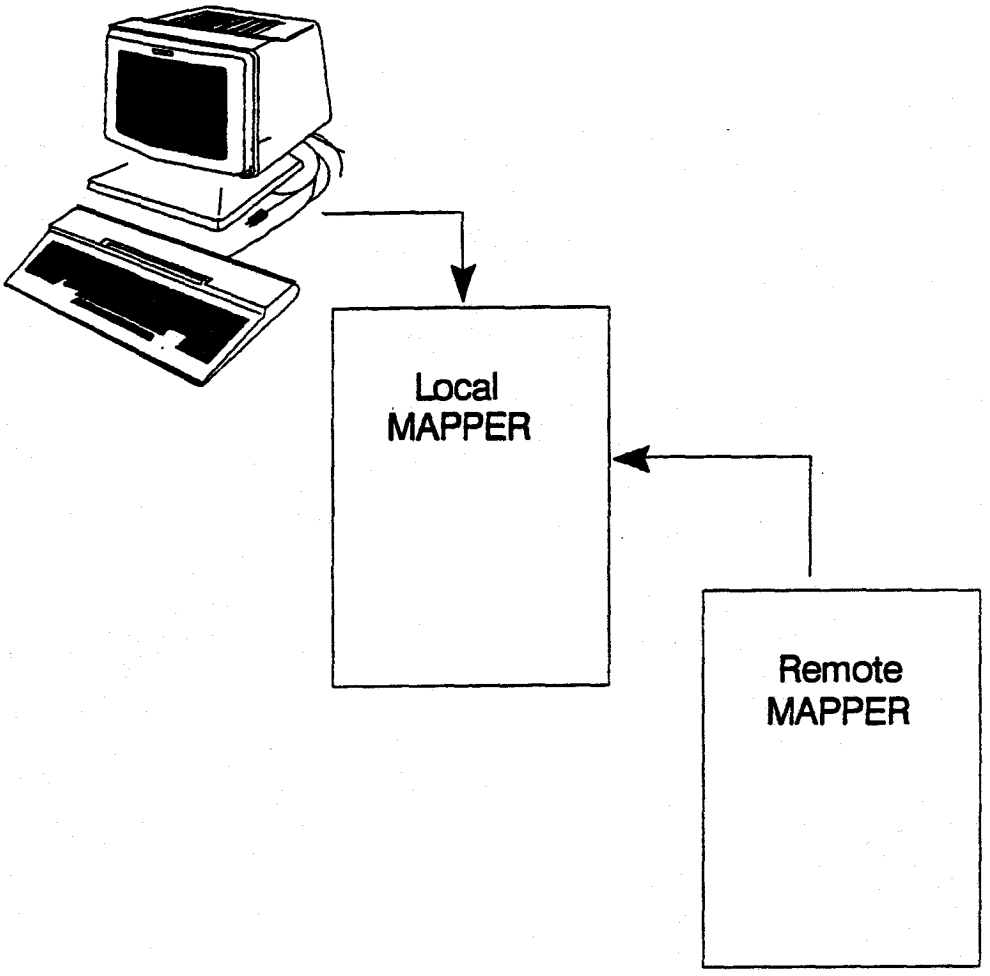
Field	Description
lab	Label to go to if an error occurs.
"run statements"	MAPPER run statements to be executed on the remote MAPPER system. Enclose the run statements in quotes. You may pass up to 256 characters in the run statements.
vmsg	Variable to contain an 80-character message if an error occurs.

Example

```
@nrn,099 "@tot,0,c,1 '' 18,6,25-7,65-8,+,-,= \
rnm -2" vls80 .
```

Perform a TOT statement in the remote MAPPER system and rename the result -2 for further processing.

NOF Statement



VB-8

NOF Statement

Description	The Network Off (NOF) statement signs the local MAPPER system caller off the remote MAPPER system.
Format	@NOF .

Summary

- NET (Network Sign On) statement signs on to the remote MAPPER system.
- NWR (Network Write) statement enables a user to send a report or result from a local MAPPER system to a remote MAPPER system.
- RTN (Return Remote) statement returns a report or result from a remote MAPPER site to a local (calling) MAPPER site.
- NRD (Network Read) statement enables the user to read and return data from a remote MAPPER system to the local MAPPER system.
- RRN (Remote Run) statement starts a run at another MAPPER site.
- NRN (Network Run) statement allows a local MAPPER system to pass run statements to a remote MAPPER system to be executed.
- NOF (Network Off) statement signs the local MAPPER system caller off the remote MAPPER system.

UNIX

Appendix A
Summaries: Functions and Runs

You may find additional information specific to your MAPPER system by signing on and entering `help,readme`.

This Appendix contains a complete list of MAPPER manual functions and runs in alphabetical order, and a summary of options for 10 of the most used manual functions.

Functions and Runs

MAPPER functions and runs are listed in alphabetical order by name. The formats are under the name and to the right of the format. Optional fields are in brackets ([*xxx*]) and "either or" conditions are in braces ({*xxx* | *xxx*}). A brief description of the function follows the format.

Abort
Abort key
Terminates a function or run in progress.

Acknowledge Message[1]
OK [*response*]
Acknowledges the message, removes it from the message queue, and redisplays it on your screen. If the sender requested an acknowledgment, the OK function returns one to the station of any user signed on with the sender's user-id.

Add Line
♦]*q*+[*predfl*]
Adds a line or lines.

Add On[1]
ADON ADON *rd*[c]
Appends a report to a displayed report or result.

Add Report
AR AR {*rd*[c]|*d*[c]}*title*
Adds a new report in the specified drawer.

Add To[1]
ADTO ADTO *rd*[c]
Appends the displayed data to another report.

Append Line
♦][*n*]A[*b*]
Copies lines from a report and appends the lines to an existing temporary buffer.

Arithmetic[1]
A A {-|*rd*[c]}
Performs computations using arithmetic expressions.

Auxiliary
AUX AUX *sn*[*sl rdc f*]
Queues data to an auxiliary printer.

Background Run
BR *run*[,*d1*,*d2*...]
Executes a run as a background process.

Binary Find[2]
BF BF {-|*rd*[c]|*d*[c]} [*f*]
Finds and displays data, within sorted reports.

Cabinet Switch
C {*cpassword* | *cnm password*}
CS CS [*sl*,]c[*r*]
Selects a cabinet; *cpassword* is the cabinet number and password (for example, C 0open), and *cnm password* is the cabinet name and password (for example, C home open).

Calculate[1]
CAL CAL *rd* [*c f*]
Performs complex computations and conditional evaluations on reports.

Calculate Update[1]
CALU CALU *rd*[*c f*]
Performs complex computations and conditional evaluations for update on reports.

Calendar[1]
CALENDAR[,*mmm yyyy*]
Displays calendar on screen.

Change[4]
CHG CHG *rd*[*c f*] or
CHG [*rdc f*];/*tgtstr*/*replstr*/*o*
Changes a character string to a new image.

Communications Output Printer
COP COP *sn*[,*cys*,*f*,/*sp*]
For BTOS: COP *printer-name*
Sends the currnet report to the auxiliary printer.

Compare Report
CMP CMP *rd*[*c f*,/]
Compares the contents of one report or result with another report.

Copy Report to DOS
SAV PCU
Copies a MAPPER report to a DOS file in either MAPPER format or straight ASCII format.

Create File[1]
FILE
Copies a report to a native data file.

Create Result Copy[1]
RSLT RSLT *rd*[c]
Creates a result copy of a report.

Create Temporary Format
VIEW
Change the selection of report fields displayed.

Date[1]
DATE — DATE *rd*[c *f*]
Performs computations on dates.

Decode Report[1]
DECODE
Transforms encoded report into readable report.

Delete[3]
DEL [*password*]
Deletes lines of an update result from a report.

Delete Line
♦]*q*-
Deletes a line or lines.

Delete Report
DR — DR *rd*[c]
Deletes a report.

Device[1]
DEV — DEV,*sn*
List auxiliary printer.

Display Alternative Format
F*n*
Display a report in a different format (0-25).

Display and Hold Headings
DH
Hold the headings at the top of the screen as you roll through the report

Display Line Count
DLC[,S]
Places line numbers of report or result on screen display.

Display Report
D — *rd* [*f*,*l*] or
Displays a report. — D *dataname* [*f*,*l*]

DOS
DOS — PCU
Enter DOS.

Drawer Password
DPW °Enter Drawer Password°
The message displayed when you try to access a report that has a drawer password.

Drawer Table of Contents
T
Displays the available drawers in your cabinet.

Duplicate Line
♦]*x*X[*q*]
Duplicates a line or lines.

Duplicate Report
XR — XR {*rd*[*c*]|*d*[*c*]} [*title*]
Duplicates a report.

Encode Report[1]
ENCODE
Transforms report into unreadable code.

Exit MAPPER System
EXIT
Exits MAPPER system and returns to previous environment.

Extract[3]
EXT [*password*]
Deletes lines of an update result from the original report and redisplays the result.

FIND
F — F {-|*rd*[*c*]|*d*[*c*]} [*f*]
Finds and displays data.

Form Generation
FORMGEN
Builds report for generating new drawer.

Help
HELP — HELP,*topic*
Provides information on MAPPER capabilities and usage.

Harvard Graphics
HG — PCU
Executes Harvard Graphics software and, optionally, sends a MAPPER report for processing.

Hold Lines on Screen
H*n*
Holds a specified number of lines at the top of the screen as you roll through the report.

Index[1]
I — I [*q*]*d*
Indexes a drawer.

Index User[1]
IU IU *[q]d*
Indexes a drawer by user, update dates, and range of reports.

Insert Line
]xl/[{,q|-ll}]
Inserts a line or lines.

Iterative Calculate
ICAL[*] ICAL *equationset*[,*r*]
Creates Calculate equation sets that can be saved, altered, and used again.

Kill
KILL KILL {*runname*|,*sn*|*runname sn*}
Terminates an active run.

Language
LANG LANG *n*
Specifies the language to display system messages.

LIMITS
LIMITS
Displays the highest report number allowed and the maximum number of lines per report allowed for cabinet and drawer.

Line Control
L
Restores the control line when it has been overwritten.

Line Zero
LZ LZ *rd*[c]
Displays identification information.

Locate[4]
LOC LOC *tgtstr* or
LOC *rd*[c *f*] LOC [*rdc f*];/*tgtstr*/*o*
Locates and displays a character string.

Look Switch
LOOKSWITCH
Alternates between the two user interfaces of MAPPER software — menu interface and control line interface.

Lotus 1-2-3
LOTUS PCU
Executes Lotus 1-2-3 software and, optionally, sends a MAPPER report for processing.

Match[1]
MA MA *rd*[c *f*]
Compares, matches, and moves data.

Match Update[1]
MAU — MAU *rd*[*c f*]
Matches data for update.

Message Waiting
MSG
Accepts an incoming message.

Microsoft Word
WORD
Executes Microsoft Word software and, optionally, sends a MAPPER report for processing.

Move Line
♦]*x*M/[{,*q*|-*ll*}]
Moves a line or lines.

Name
NAME — NAME *name*[,*c*,*d*,*r*,*dept*,*user*,*act*,*desc*]
Creates a directory name for a cabinet, drawer, or report.

Names[1]
NAMES
Displays data names registered in System Directory.

Paint
PNT
Paints the screen.

Password
PSW *password*
Assigns, changes, or clears report write passwords; unlocks reports for updating.

Print
PR — PR *rd*[*c f*]
Queues data to a system printer.

Put Line
♦]P[*b*]
Copies lines from the temporary yank or append buffers into a report or result.

Read Password
RPSW — RPSW {*password*| *keyword*}
Limits access to individual reports.

Reformat Report[1]
RF — RF *rd*[*c f*]
Moves columns of data.

Release Display

Releases displays or runs.

Remote Run

RR *run*[,*d1*,*d2*...]
Starts a MAPPER run at a remote site.

Replace Report

REP — REP *rd*[*c title*]
Replaces a report with a displayed report or result.

Resume

RSM
Resumes executing a suspended function or run.

Retrieve File[1]

RET
Retrieves a native data file.

Save Report Version[1]

SV
Renames a report or result for temporary storage. After doing this, press **Resume** or enter **rsm** (the Resume function) to redisplay it as a result.

Search[1]

S — S {-|*rd*[c]|*d*[c]} [*f*]
Searches for data.

Search Update[1]

SU — SU {-|*rd*[c]|*d*[c]} [*f*]
Searches reports for update.

Send Report

SEND — SEND *sn*[*sl*,*ack?*]
Sends a report or result to another station.

Send Report to User

SNU — SNU *user-id*[,*dept*,*sl*,*ack?*]
Sends an entire report or result to another user.

Shift Display

S*n*
Shifts the displayed portion of the report.

Sign off MAPPER Software

X
Signs station off from the MAPPER software.

Sign on MAPPER Software

]*user-id*,*dept-no*[,*passwd*]
Signs on to MAPPER software.

SOE Update
♦*changes*
Changes data between SOE (♦) and the cursor (▮).

Sort[1]
SORT SORT *rd*[*c f*]
Reorders lines of data.

Sort and Replace Report
SORTR SORTR *rd*[*c f*]
Reorders the lines in a report and replaces the sorted result into the original report.

Start
START
Starts a job in the native environment.

Station-to-Station Message[1]
SS SS,*C*
Sends messages (up to a full screen) to another station.

System[1]
SYSTEM
Displays active users, active runs, and other system information.

Totalize[1]
TOT TOT *rd* [*c f*]
Performs arithmetic and move operations.

Transfer DOS File to MAPPER
RES PCU
Transfer a DOS file to the MAPPER system.

Undo
UNDO
Reverses the last operation (only available for some functions).

For U Series only: Unisys UNIX Interface[5]
UNIX [*o*[*fnm*]] [*cmd* [*arg*]]
Accesses the UNIX system and lets you issue UNIX commands.

Update[3]
UPD [*password*]
Blends lines in the results of an update function into reports.

Yank Line
♦][*n*]Y[*b*]
Copies lines from a report or result into a temporary buffer.

Notes on Functions and Runs

1. Creates a result.
2. Creates a result with the N or O option.
3. Use with update functions:
   - Calculate Update
   - Change (with the OU option)
   - Locate (with the OU option)
   - Match Update
   - Search Update
4. Creates a result with the O or OU option.

Abbreviations for Functions and Runs

ack?	Acknowledgement request
act	Action(C,D,I,R,or,Q)
arg	Arguments passed to UNIX command
b	Buffer number
c	Cabinet number
changes	Modifications to report
cmd	UNIX command
cnm	Cabinet name
cpsw	Cabinet number and password
d	Drawer letter
d1,d2...	Data items
dept	Department number
desc	Description of named cabinet, drawer, or report
eqnm	Equation set name
f	Format
fnm	UNIX file
key	Character key
keyw	Special keyword
l	Line number
ll	Last line number
lvl	Encryption level
mmm	First three letters of month
msg	Message
n	Number
nm	Name
o	Options
predfl	Predefined lines defined in report 0
psw	Password
q	Quantity of lines
qd	Quantity of lines to index and drawer
r	Report number
rd	Report and drawer
replstr	Replacement string
run	Run name
sl	Site letter

sn	Station number
tgt	Target
tgtstr	Target string
user-id	User-id
x	Number of times to duplicate a line
yyyy	Year
-	Report or result on display
*	Use report field names

Options for 10 Common Functions

The following list includes the options for 10 common manual functions:

BF	A, B, C[2], E, F, I[*n*], K, N, O, P, Q, R*x*{-*y*}[4], S, U, @, /
CAL	A, C, E, I, J(*x*), K[*n*], L, N[*n*], O, R*n*, S(s), T, V, X, *
CHG	A, C[2], F, O, OU, S*x*{-*y*|,*n*}, T*x*
DATE	A, n, T, W
F	A, C[2], R*x*{-*y*| *y*}, @, /
LOC	A, B[*n*][1], C[2], F, M[3], O, OU, S*x*{-*y*|,*n*}, T*x*, U[1]
MA	A, B, C[2], D, E, F, I[1], M, N, P, Q, S
S	A, B[(*n*)], C[2], D, F, H, L(*x*), N, P, Q[(*n*)], R*x*{-*y*| *y*}, T[(*x*)], U[(*x*)], @, /
SORT	A, C[2]
TOT	A, C[2], E, H, I, J(*x*), N, O, R*n*, S, =x, *

Table A-1 summarizes these options.

Table A-1. Options for 10 Common Manual Functions

Option	Purpose	Functions
A	Process all line types.	BF CAL CHG DATE F LOC MA SORT S TOT
B	Build index.	BF
B[*n*][1]	Back up *n* lines after locate.	LOC
B	Blend issuing and receiving reports.	MA
B[(*n*)]	Stop search after *n*th find. Default = first find.	S
C[2]	Character set control (case sensitive).	BF CHG F LOC MA SORT S TOT
C	Conditionally display specific result lines.	CAL
D	Omit match or search information lines from result.	MA S
E	Display last occurrence of item found.	

continued

Table A-1. Options for 10 Common Manual Functions (cont.)

Option	Purpose	Functions
E	Erase fields (fill with spaces) if value = 0.	CAL
E	Count entries.	TOT
F	Process all line types; locate or change full character string.	CHG LOC
F	Do not fill move fields on a no-match condition.	MA
F	Search for floating-point numbers.	S BF
H	Display only header lines from first report in multiple report search.	S
H	Cumulate horizontally.	TOT
I[*n*]	Use index in report *n*. Default = report 2.	BF
I	Produce integer results.	CAL
I[1]	Issuing report on display.	MA
I	Ignore header restrictions.	TOT
J(*x*)	Numerically justify result value to *x*: *x* = c, l, r, x, or z.	CAL TOT
K	Verify that reports are sorted in ascending order.	BF
K[*n*]	Initialize value label to *n*.	CAL
L	List value label names or values in result.	CAL
L(*x*)	Omit line type *x* from result	S
M[3]	Treat first character of target string as line type designator.	CHG LOC
M	Display only matched lines in result.	MA

Table A-1. Options for 10 Common Manual Functions (cont.)

Option	Purpose	Functions
N	Create separate line per item and item count in result.	BF
N[*n*]	Substitute numeric value *n* for nonnumeric fields. Default = 0.	CAL TOT
N	Display lines not meeting match or search parameters.	MA S
O	Create result containing items found.	BF CHG LOC
O	Omit data lines from result; include headers, value labels or totals.	CAL TOT
OU	Create update result.	CHG LOC
P	Include period lines in result (valid only with N option).	BF
P	Issuing and receiving reports are presorted.	MA
P	Process paragraphs.	S
Q	Quick-find a single item.	BF
Q	Specifies reports are sorted and will not be verified.	MA
Q[(*n*)]	Stop scan after *n*th paragraph. Default = first paragraph; use with the P option.	S
R*x*{-*y*|,*y*}[4]	Scan range of reports: reports *x* through *y*; scan reports *x*,*y*.	BF F S
R*n*	Round answers to nearest *n*.	CAL TOT
S	Scan each report separately.	BF
S*x*{-*y*|,*n*}	Start scan at line *x* through line *y*; scan *n* lines.	CHG LOC
S	Display matched or found lines in issuing report or search parameter order.	MA
S	Place subtotals in vertical operation fields.	TOT
S(s)	Case sensitivity.	CAL

Table A-1. Options for 10 Common Manual Functions (cont.)

Option	Purpose	Functions
T	Include processed and unprocessed lines in result.	CAL
T*x*	Set *x* to transparent character.	CHG LOC
T	Convert time in field to decimal hours and move field.	DATE
T[(*x*)]	Include last *x* type line in result. Default = tab line.	S
U	Set update lock.	BF
U[1]	Resume scan beyond lines on display.	LOC
U[(*x*)]	Search within data unit; include unit in result. Default = tab line.	S
V	Process only equations whose result values are calculated from valid data.	CAL
W	Determine day of the week.	DATE
X	Exclude invalid values in MIN, MAX, SUM, AVG, VMIN, VMAX, VSUM, and VAVG computations.	CAL
n	Specify *n* workdays in week.	DATE
@	Find or search for blank characters (spaces).	BF F S
/	Find/search for slant as data.	BF F S
=*x*	Change column 1 to *x*.	TOT
*	Omit error flag (*) in subtotaling operations.	TOT
*	Flag invalid results with asterisk.	CAL

1. Not applicable in MAPPER runs.
2. Option C varies with function.
3. LCH = CHANGE manual function; M option not applicable for manual CHANGE and LOCATE functions.
4. For BF (Binary Find), only R*x* and R*x-y* are allowed.

Appendix A
Summary of Statements and Options

You may find additional information specific to your MAPPER system by signing on and entering **help,readme**.

This appendix contains the following:

- MAPPER run statements
- Field and subfield abbreviations
- Options for 10 common run statements

Run Statements

This part of the appendix contains a list of all MAPPER run statements. See "Field and Subfield Abbreviations" in this appendix for descriptions of the abbreviations used in these formats.

Reminders for Run Statement Syntax

Remember these points when formulating run statements:

- Separate fields with spaces.
- Separate subfields with commas.
- Refer to a current result as -0.
- Double check your MAPPER run statement syntax.
- If not specifying options, use apostrophes ('') in the *o* (options) field.
- Place apostrophes before and after the following items:
  - A space if it does not terminate a field
  - A comma if it does not terminate a subfield
  - A slant (/) if it does not indicate multiple parameters

Run Statement Formats

Following is a list of all the available run statements:

@ADD,*ic,id,ir,rc,rd,rr* .
@ADR,*c,d*[,*r,lab vtitle*] . (RPT$ or STAT1$ = new report number.)
@ART *exp vrslts* .
@ASR[,*rmt?,xmt?,ci,di,ri,hi?,in,co,do,ho?,out,lab*] *fn* [*args*] .
@AUX,*c,d,r,sn,dev*[,*dlnos?,f,,dhdgs?,d1char?,lsp,,,sl,spcc*] .

@BFN,*c,d*[,*r,l,lab*] *o cc ltyp,p* [*vrpt,vlno*] .
@BLT,*c,d,r*[,*lab*] .
@BR[,*c,d,r*] *run*[,*vld*] .
@BRG[,*c,d,q*] .
@BRK[,*c,d,q*] .

@CAB,*c,d* .
@CAH,*c,d,r* .
@CAL,*c,d,r*[,*l,q,lab*] *o cc ltyp,p eq* [*vrslts*] .
@CALL[,*c,d,r*] *lab* ([*v,v,...*]) .
@CAR .

@CAU,*c,d,r*[,*l,q,lab*] *o cc ltyp,p eq* [*vrslts*] .
@CER .
@CHD[,*c,d,r,rel?*] *lab* .
@CHG *v* {*exp*|*vld*} . or @CHG *rw v*[,*v...,v*] .
@CLK .
@CLT,*c,d,r*[,*lab*] .
@CLV[,*stvno,q*] .
@CMP,*c1,d1,r1*,[*lin1*],*c2,d2,r2*[,*lin2,q,lab*] *o cc1 ltyp1,p1 cc2 ltyp2,p2*
Δ[*vlno1,vcol1,vlno2,vcol2*] . (Δ indicates a space.)
@CMU .
@CSR .

@DAT,*c,d,r o cc ltyp,p* .
@DC *eq vrslts* .
@DCR,*c,d,r,key*[*-level,,dspscram?*] .
@DCU .
@DDI,*sltnm*[,*fnum,lab*] [*vname,vsz,vdtyp,vasz*] .
@DEC[,*n*] *v*[,*v,...v*] .
@DEF[,*o,lab*] *setv*{,|Δ}*testv* . (Δ indicates a space.)
@DEL .
@DEV,*sn*[,*dev,,lab*] .
@DFU[,*lab*] *c,d,r*[,*c,d,r,...c,d,r*] .
@DIR[,*lab*] *name* [*vcabinet,vdrawer,vrpt,vhirptr*] .
@DLR,*c,d,r*[,*lab*] .
@DOE,*c,d*[,*dsp*] .
@DRW,*c,d*[,*lab vcpl,,,,vnxrd,vhirptd,vlnd,vrptsd,vrlmt,vllmt*] .
@DSG,*c,d,r*[,*display,interim?*] .
@DSM,*c,d,r,lmsg*[,*tabp,erase?,interim?,pdq,dc,dd,dr,dspl,dspf*] .
@DSP,*c,d,r*[,*l,tabp,f,interim?,hold,msg80*] .
@DSX,*c,d,r*[,*l,tabp,f,,hold,msg80*] .
@DUP,*c,d,r*[,*rc,rd vtitle*] . (RPT$ or STAT1$ = new report number.)
@DVS[,*c,d,r,lab*] *field*[,*field,...,field*] *v*[,*v,...,v*] .

@ECR,*c,d,r,key*[*-level,hdgs?,,dspscram?*] .
@ESR[{,*q*|,*-q*}] .
@EXT .

@FCH,[*c,d*],*sltnm*[,*l,q,lab*] *o fdes* [*vrslts*] .
@FDR,*c,d*[,*r,l,q,lab*] *o cc ltyp,p* [*vrpt,vlno*] .
@FIL,*c,d,r*[,*mapperf?,hdgs?,lab*] *fn* .
@FMT[,*c,d,r*] *field*[,*field,...,field*] .
@FND,*c,d*[,*r,l,lab*] *o cc ltyp,p* [*vrpt,vlno*] .

@GOC,*c,d,r*[,*lab*] *ulvl?*[,*notxt?,ige?,igcz?,fxcz?*] *optmz?*[,*outrslt?,*
unp? vlinesi,vlineso,vco,vbuffz] .
@GS,*c,d,r*[,*lab*] *maxy*[,*o,ige?,unp?,aga?,expand?,ighitxt?,outrslt?,plotter?*] *sfx*[,*offx,*
offy,]*sfy angle*[,*absx,absy vci,vco,vminx,vmidx,vmaxx,vminy,vmidy,vmaxy*] .
@GTO {*lab*|END[,*n,n,n*]|LIN [+]*n*|LIN *-n*|RPX *r*} . (+ or - is the number of lines following or preceding the present line.)

@HOF .
@HRD,*ic,id,ir,rc,rd*[,*rr,lab vmsg*] .
@HRN[,*lab*] "*run statements*" [*vmsg*] .
@HSH *v=vld(min-max)*[,*v=vld(min-max)...,v=vld(min-max)*] .
@HST,*site*[,*rmu,rmd,rmpw,trnrpt,scl,col,mtr*] .
@HWR,*ic,id,ir,rc,rd,rr*[,*lab vmsg*] .

@IDU,*c,d*[,*q,user,sdate,endate,srpt,endrpt vrpts,vlines,vrptsd,vhirptd*] .
@IF[,C] *val1 op val2* [{,*val3*|& *op val3*}] *stmt1* . ; [*stmt2*] .
@INC[,*n*] *v*[,*v,...v*] .
@IND,*c,d*[,*q,lab*] .
@INS *vld substrv* .
@ITV[,*lab*] *v*[,*v...,v*] .

@JUV,*o v*[,*v,...v*] .

@KEY .

@LCH,*c,d,r*[,*l,lab*] *o cc tgtstr/replstr* [,*vlines,vrpt*] .
@LCV[,*lab*] *o v* {*v*|*tgtstr*}[/*replstr vpos,voccs*] .
@LDA[,*o*] *nametypesize*[*n*]=*vld*[,*vld...,vld*] .
@LDV[,*o*] *v=vld*[,*v=vld,...,v=vld*] . or @LDV,*o v*[,*v...,v*] .
@LFC *v* .
@LFN[,*c,d,r,tics?,lab*] *cc v*[,*v,...,v*] .
@LGF .
@LGN[,*user,psw*] .
@LLN,*c,d,r*[,*lab*] *vlines* .
@LN+,*c,d,r,lb4,q*[,*predfl*] .
@LN-,*c,d,r,l,q* .
@LNA,*c,d,r,l*[,*q,b*] .
@LNG,*n* .
@LNI,*c,d,r,lb4*,[*x*],*l*[,*q*] .
@LNK *run*[,*vld*] .
@LNM,*c,d,r,lb4*,[*x*],*l*[,*q*] .
@LNP,*c,d,r,lb4*[,*b*] .
@LNX,*c,d,r,l,x*[,*q*] .
@LNY,*c,d,r,l*[,*q,b*] .
@LOC,*c,d,r*[,*l,lab*] *o cc tgtstr* [*vcol,vlno,vrpt*] .
@LOG .
@LOK,*c,d,r*[,*lab*] .
@LSM,*msgno*[,*lab*] *vmsg* .
@LZR,*c,d,r*[,*lab vlines,vcpl,vhdgs,,,vdept,vuser,vrpw,vwpw,vlgn*] .

@MAU,*ic,id,ir,rc,rd,rr*[,*lab*] *o icc iltyp,ip rcc rltyp,rp* .
@MCH,*ic,id,ir,rc,rd,rr*[,*lab*] *o icc iltyp,ip rcc rltyp,rp* .
@MSG *vld* .

@NET,*net-id*[,*site-id,rmu,rmd,rmpw,trnrpt,mtr,lab*] .
@NOF .
@NRD,*ic,id,ir,rc,rd,rr*[,*lab vmsg*] .
@NRM "*cmd*" .
@NRN[,*lab*] "*run statements*" [*vmsg*] .
@NRT "*cmd*" .
@NWR,*ic,id,ir,rc,rd,rr*[,*lab vmsg*] .

@OK[,*lab,vrsp*] .
@OTV[,,*scl,col*] *vld* .
@OUM,*ic,id*[,*ir,if,rc,rd,rr,rf,title*] .
@OUT,*c,d,r,l,q*[,*outl,tabp,erase?,interim?,pdq,protect,fxmt?,outsp?,blink?*] .
@OUV[,,*scl,col*] *vld* .

@PEK[,*startv,q,level*] .
@PNT .
@POK[,*startv,q,level*] .
@POP[,*startv,q,level*] .
@PRT,*c,d*[,*r,dlnos,f,prtsite,cys,all?,lsp,dstn,,hdgs?*] .
@PSH[,*startv,q,level*] .

@RAR{,*c,d,r lab* | *lab*} .
@RDB .
@RDC,*c,d,r*[,*l,q,ltyp,lab*] *cc vdata* . (Place next line in output area.)
@RDL,*c,d,r,l*[,*lab*] *cc vdata* .
@REL .
@REP[,*ic,id,ir*],*rc,rd,rr* [*vtitle*] .
@RER{,*c,d,r lab* | *lab*} .
@RET,*c,d*[,*mapperf?,hdgs?,lab*] *fn* .
@RETURN .
@RFM,*ic,id,ir,rc,rd,rr o icc ,ip rcc ,rp* .
@RLN[,*l,lab*] *cc vdata* .
@RMV[,*level*] .
@RNM[,*c,d,r*] *-n* .
@RPW{,*pw1* |,*pw1,pw2* |,,*pw2*} .
@RRN[,*lc,ldr,lr*] *run*[,*vld*] *rms,rmu,rmd*[,*rmpw,rmc,rmdr*] .
@RSL,*c,d,r* .
@RSR{,*c,d,r lab* | *lab*} .
@RTN,*rc,rd,rr* .
@RUN {*run*[,*vld*] | "*cmd*"} .

@SC[,,,,,*tabp*] *o scmnd* . (format 1)
@SC,*c,d,r*[,*l,q,tabp*] *o* [*fldtxt*] . (format 2)
@SCH[,*date,time*] *rst* .
@SEN,*c,d,r,sn*[,*sl,ack?,lab*] .
@SFC[,*c,d,r*] *vld* .
@SNU,*c,d,r,user*[,*dept,sl,ack?,lab*] .
@SOR,*c,d,r o cc ltyp,p* .
@SQL[,*c,d,r,lab,sltnm*] ['*sqlstmt*'] [*vls,vfds*] .
@SRH,*c,d*[,*r,l,q,lab*] *o cc ltyp,p* [*vlines,vls,vrpt*] .
@SRR,*c,d,r o cc ltyp,p* .
@SRU,*c,d*[,*r,l,q,lab*] *o cc ltyp,p* [*vlines,vls,vrpt*] .
@STN[,*sn,lab vuser,vdepn,vdept,vttyp,vvsiz,vhsiz*] .
@STR,*c,d,r*[,*sl,ifc,logon,psw*] .
@SUB,*c,d,r*[,*lab*] *o cc ltyp,p vrslts* . (Place next line in output area.)

@TOT,*c,d,r*[,*lab*] *o cc ltyp,p* [*vrslts*] .

@ULK .
@UNX[,*c,d,logon,pw,lab*] *o cmd* [*args*] .
@UPD .
@USE *name=v*[,*name=v*,...,*name=v*] .

@WAT[,M,*lab*] *ms* .
@WRL,*c,d,r,l*[,*ntuid?,wpw*] *cc ltyp,vld* .

@XCH[,*startv,q,level*] .
@XIT .
@XQT{,*lab* | *vld*} .
@XUN .

@**cmd* .

Field and Subfield Abbreviations

Following are the abbreviations used in the preceding run statements:

absx	Absolute X rotation value
absy	Absolute Y rotation value
ack?	Request acknowledgment
aga?	Assume graphics active, Y/N
all?	All reports in drawer, Y/N
angle	Rotation angle
args	Arguments
b	Buffer label
blink?	Change <> to blinkers, Y/N
c	Cabinet number
c1	Cabinet to compare
c2	Cabinet to compare with the first cabinet
cc	Column-character positions
cc1	Column-character positions to compare in the first report
cc2	Column-character positions to compare in the second report
ci	Cabinet to send as input
cmd	Command
co	Cabinet in which to replace the result
col	Column number
cys	Number of copies
d	Drawer
d1	Drawer to compare
d2	Drawer to compare with the first drawer
date	Date
dc	Cabinet number of report to display (DSM statement)
dd	Drawer of report to display (DSM statement)
delim	Variable content delimiter
dept	Department number
dev	Device name or type
dhdgs?	Delete headings, Y/N
di	Drawer to send as input
display	Display text, graphics, both (A = text, G = graphics, M = mixed)
dlnos?	Delete line numbers, Y/N
do	Drawer in which to replace the result
dr	Report number of report to display (DSM statement)
dspf	Format of report to display (DSM statement)
dspl	Line number of report specified to display (DSM statement)
dspscram?	Display scrambled data, Y/N
dstn	Destination print model
d1char?	Delete first character, Y/N

Field and Subfield Abbreviations

endate	Ending date
endrpt	Ending report number
eq	Equations
erase?	Erase screen, Y/N
exp	Arithmetic expressions
expand?	Handle expanded syntax, Y/N
f	Format of report
fdes	ORACLE - Field designators indicating which fields to retrieve
field	Report field
fldtxt	Field text
fn	File name
fnum	ORACLE - Field number of the field for which information is requested
fxcz?	Fixed character size, Y/N
fxmt?	Forced transmit, Y/N
hdgs?	Include headings, Y/N
hi?	Append headings to input file, Y/N
ho?	Append headings to output file, Y/N
hold	Number of lines on the display screen to hold
ic	Issuing cabinet
icc	Issuing report column-character positions
id	Issuing drawer
ifc	Name of interface to pass the runstream to
igcz?	Ignore character size, Y/N
ige?	Ignore errors, Y/N
ighitxt?	Ignore high text, Y/N
iltyp	Issuing report line type
in	INTNAME of the input file of the program executed
interim?	Interim display, Y/N
ip	Issuing report parameters
ir	Issuing report
ivcol	Issuing report column number
ivlno	Issuing report line number
key	Key used to encode or decode a report
l	Line number in the report
lab	Label to go to
lb4	Line number before (start after this line)
lc	Local cabinet
ldr	Local drawer
level	Previously saved level
-level	Level of encryption on encoded reports
lin1	Line number in the first report to begin comparison
lin2	Line number in the second report to begin comparison
lmsg	Line number of message
logon	Logon to operating system

lr	Local report
lsp	Line spacing
ltyp	Line type
ltyp1	Line type to compare in the first report
ltyp2	Line type to compare in the second report
mapperf?	MAPPER format, Y/N
maxy	Maximum Y value
min-max	Range of numbers
ms	Milliseconds
msg80	Message (up to 80 characters long)
msgno	Message number
mtr	Monitor transferred data, Y/N
multi	Multiple reports, Y/N
n	Number
name	Data name to define
nametypesize	Variable array name, then type and size allowed for each member of array
net-id	Network identifier of remote system
notxt?	Eliminate all text, Y/N
ntuid?	No time/user-id, Y/N
o	Options
offx	Offset value for X components
offy	Offset value for Y components
op	Operator (relational)
optmz?	Optimize result, Y/N
out	INTNAME of the output file returned to the MAPPER system
outl	Line on terminal where output begins
outrslt?	Display graphics result on screen, Y/N
outsp?	A = send actual data B = perform cursor position Space = perform cursor positioning if data includes 5 or more spaces
p	Parameters
p1	Parameters for the first report
p2	Parameters for the second report
pdq	Push-down quantity (how many lines)
plotter?	Add or delete plotter primitives, Y/N
predfl	Predefined line reference number
protect	Protected format
prtsite	Print site
pthfn	Full path and file name
pw	Password
q	Quantity (how many)

Field and Subfield Abbreviations

r	Report number
r1	Report to compare
r2	Report to compare with first report
rc	Receiving cabinet
rcc	Receiving report column-character positions
rd	Receiving drawer
rel?	Release control to run, Y/N
replstr	Replacement string
ri	Report to send as input
rltyp	Receiving report line type
rmc	Remote cabinet
rmd	Remote department number
rmdr	Remote drawer
rmpw	Remote user password
rms	Remote site letter
rmt?	Program uses remote file, Y/N
rmu	Remote user-id
rp	Receiving report parameters
rr	Receiving report
rst	Run statement to queue
run	Run name
run statements	Run statements to be executed
rv	Receiving variable
rvcol	Receiving report column number
rvlno	Receiving report line number
rw	Reserved word
scl	Screen line at which to start the display
scmnd	Screen commands
sdate	Starting date
setv	Set variable
sfx	Scaling factor, X axis
sfy	Scaling factor, Y axis
site-id	Site identifier of remote MAPPER system
sl	Site letter
sltnm	ORACLE - Specify a specific SELECT name for DDI
sn	Station number
spcc	Control character used to specify printer device controls
sqlstmt	ORACLE - SQL statement
srpt	Starting report number
startv	Starting variable number
stmt	Another run statement
substrv	Variable substring
tabs?	Tab characters, Y/N
tabp	Tab position
testv	Test variable
tgtstr	Target string
tics?	Enclose field name in apostrophes, Y/N
time	Time
title	Title (up to 12 characters)
trnrpt	Report number of translation table

ulvl?	Upper-level chart, Y/N
unp?	Unpack result, Y/N
user	User-id
v	Variable name
val	Value
vasz	ORACLE - Variable to capture the actual size of the field
vbuffz	Variable with buffer size
vcabinet	Variable with cabinet number
vci	Variable with number of input characters scanned
vco	Variable with number of characters to send out
vcol	Variable with column number
vcol1	Variable with column number in the first report
vcol2	Variable with column number in the second report
vcpl	Variable with number of characters per line
vdata	Variable with data
vdepn	Variable with department name
vdept	Variable with department number
vdrawer	Variable with drawer letter
vdtyp	ORACLE - Variable to capture data type on a DDI return
vfds	ORACLE - Variable to return name of a field on a DDI statement
vhdgs	Variable with number of heading lines
vhirptd	Variable with highest report number in drawer
vhirptr	Variable with higher report number in range
vhsiz	Variable with horizontal screen size (character positions)
vld	Variables, literal data, or both (may include reserved words also, see individual statement descriptions)
vlgn	Variable with language number
vlines	Variable with number of lines
vlinesi	Variable with number of input lines scanned
vlineso	Variable with number of output lines in result
vllmt	Variable with line limit for the drawer
vlnd	Variable with number of lines in a drawer
vlno	Variable with line number
vlno1	Variable with line number in the first report
vlno2	Variable with line number in the second report
vls	Variable with number of lines scanned
vmaxx	Variable with maximum X value
vmaxy	Variable with maximum Y value
vmidx	Variable with midpoint X value
vmidy	Variable with midpoint Y value
vminx	Variable with minimum X value
vminy	Variable with minimum Y value
vmsg	Variable with a message
vname	ORACLE - Variable to return the name of a field on a DDI statement
vnxrd	Variable with next available report in the drawer
voccs	Variable with number of occurrences
vpos	Variable with character position
vrlmt	Variable with report limit for the drawer
vrpt	Variable with report number
vrpts	Variable with number of reports found
vrptsd	Variable with number of reports in a drawer

vrpw	Variable with read password
vrslts	Variable with results
vrsp	Variable containing a message response
vsz	ORACLE - Variable to capture maximum size of a field for a DDI statement
vtitle	Variable with the title of the new report
vttyp	Variable with terminal type code
vuser	Variable with user-id
vvsiz	Variable with vertical screen size (rows per screen)
vwpw	Variable with write password
wpw	Write password
x	Times (number of times)
xmt?	User must press **Transmit** before returning to MAPPER software, Y/N

Options for 10 Common Run Statements

This list summarizes the options for 10 common run functions.

BFN A, B, C[2], E, F, I[*n*], K, N, O, P, Q, R*x*{-*y*|*y*}[4], S, U, @, /
CAL A, C, E, I, J(*x*), K[*n*], L, N[*n*], O, R*n*, S(s), T, V, X, *
DAT A, n, T, W
FND A, C[2], R*x*{-*y*|*y*}, @, /
LCH A, C[2], F, M[3], O, OU, S*x*{-*y*|,*n*}, T*x*
LOC A, B[*n*][1], C[2], F, M[3], O, OU, S*x*{-*y*|,*n*}, T*x*, U[1]
MCH A, B, C[2], D, E, F, I[1], M, N, P, Q, S
SOR A, C[2]
SRH A, B[(*n*)], C[2], D, E*n*, F, H, L(*x*), N, P, Q[(*n*)], R*x*{-*y*|*y*}, T[(*x*)], U[(*x*)], @, /
TOT A, C[2], E, H, I, J(*x*), N, O, R*n*, S, =x, *

Table A-1 summarizes the options for 10 commonly used run statements.

Table A-1. Options for 10 Common Run Statements

Option	Purpose	Functions
A	Process all line types.	BFN CAL LCH DAT FND LOC MCH SOR SRH TOT
B	Build index.	BFN
B[*n*][1]	Back up *n* lines after the found line.	LOC
B	Blend issuing and receiving reports.	MCH
B[(*n*)]	Stop search after *n*th find. Default = first find.	SRH
C[2]	Character set control (case sensitivity).	BFN LCH FND LOC MCH SOR SRH TOT
C	Conditionally display specific result lines.	CAL
D	Omit match or search information lines from result.	MCH SRH
E	Display last item only if the target item appears more than once.	BFN

continued

Table A-1. Options for 10 Common Run Statements (cont.)

Option	Purpose	Functions
E	Do not move blank fields from the issuing report.	MCH
E	Erase fields (fill with spaces) if value = 0.	CAL
E	Count entries.	TOT
E*n*	Estimate *n* number of lines in result.	SRH
F	Process all line types; locate or change full character string.	LCH LOC
F	Do not fill move fields on a no-match condition.	MCH
F	Search for floating-point numbers.	SRH
F(*n*,*n*...)	Specifies the order of sorted fields when searching multiple fields.	BFN
H	Display heading lines only from first report in multiple report search.	SRH
H	Cumulate horizontally.	TOT
I[*n*]	Use index in report *n*. Default = report 2.	BFN
I	Produce integer results.	CAL
I[1]	Issuing report on display.	MCH
I	Ignore heading restrictions.	TOT
J(*x*)	Numerically justify result value to *x*: *x* = c, l, r, x, or z.	CAL TOT
K	Verify that reports are sorted in ascending order.	BFN
K[*n*]	Initialize value label to *n*.	CAL
L	List value label names or values in result.	CAL
L(*x*)	Omit line type *x* from result	SRH
M[3]	Treat first character of target string as line type designator.	LCH LOC
M	Display only matched lines in result.	MCH

Table A-1. Options for 10 Common Run Statements (cont.)

Option	Purpose	Functions		
T	Include processed and unprocessed lines in result.	CAL		
Tx	Set *x* to transparent character.	LCH	LOC	
T	Convert time in field to decimal hours and move field.	DAT		
T[(x)]	Include last *x* type line in result. Default = tab line.	SRH		
U	Set update lock.	BFN		
U[1]	Resume scan beyond lines on display.	LOC		
U[(x)]	Search within data unit; include unit in result. Default = tab line.	SRH		
V	Process only equations whose result values are calculated from valid data.	CAL		
W	Determine day of the week.	DAT		
X	Exclude invalid values in MIN, MAX, SUM, AVG, VMIN, VMAX, VSUM, and VAVG computations.	CAL		
n	Specify *n* workdays in week.	DAT		
@	Find or search for blank characters (spaces).	BFN	FND	SRH
/	Find/search for slant as data.	BFN	FND	SRH
=x	Change column 1 to *x*.	TOT		
*	Omit error flag (*) in subtotaling operations.	TOT		
*	Flag invalid results with asterisk.	CAL		

1. Not applicable in MAPPER runs.
2. Option C varies with function.
3. LCH = Change manual function; M option not applicable for manual Change and Locate functions.
4. For BFN (Binary Find), only R*x* and R*x*-*y* are allowed.