UNISYS

MAPPER® System Run Design

Operations Reference Manual

Volume 1: Usage

May 1990

Printed in U S America
7831 9274–000

Priced Item

UNISYS

MAPPER® System Run Design

Operations Reference Manual

Volume 1: Usage

May 1990

Printed in U S America
7831 9274–000

Priced Item

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual, living or otherwise, or that of any group or association is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financiai or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded using the Business Reply Mail form in this document, or remarks may be addressed directly to Unisys Corporation, MAPPER Product Information, P.O. Box 64942 MS: 4792, St. Paul, Minnesota, 55164-0942, U.S.A.

Page Status

Page	Issue
Volume 1	
iii through xvi	Original
Contents tab	Original
xvii through xxxi	Original
xxxiii through xxxiv	Original
Section 1 tab	Original
1-1 through 1-6	Original
Section 2 tab	Original
2-1 through 2-16	Original
Section 3 tab	Original
3-1 through 3-8	Original
Section 4 tab	Original
4-1 through 4-44	Original
Section 5 tab	Original
5-1 through 5-20	Original
Section 6 tab	Original
6-1 through 6-18	Original
Appendixes tab	Original
A-1 through A-21	Original
B-1 through B-8	Original
C-1 through C-4	Original
D-1 through D-15	Original
E-1 through E-9	Original
F-1 through F-3	Original
G-1 through G-2	Original
Volume 2	
iii through iv	Original
Contents tab	Original
v through xxii	Original
Section 7 tab	Original

Page	Issue
7-1 through 7-21	Original
A-E tab	Original
7-23 through 7-119	Original
F-P tab	Original
7-121 through 7-233	Original
R-Z tab	Original
7-235 through 7-364	Original
Glossary tab	Original
Glossary-1 through Glossary-44	Original
Index tab	Original
Index-1 through Index-25	Original

About This Manual

Purpose

This manual provides complete descriptions, formats, and examples for MAPPER run statements, as well as instructions for designing runs. By using this manual, the run designer should be able to write and update complete runs, obtain quick access to statement syntax, and learn efficiency techniques by following examples.

Scope

This manual contains reference information about run statements. It is not meant to teach run design; it is designed to provide quick reference to run statement information.

Audience

This manual is a reference for the user who has completed the *MAPPER System Run Design Operations Training Guide.* It provides information for the user who understands system fundamentals but needs a source of procedures and information specific to writing runs.

Prerequisites

Before using this manual, you should be comfortable with MAPPER manual functions and the layout of the MAPPER database. You should understand basic run design concepts and you should know how to write basic runs. For more information, refer to the *Run Design Training Guide.*

How to Use This Manual

To aid your understanding of this manual and your use of MAPPER software, certain style conventions are used. Following is a description of how this manual handles special characters, system-specific information, examples, keyboard key names, uppercase letters, italics, and color.

After reading this section, please refer to Appendix E for a description of the key names used in this manual.

If you have just completed the *Run Design Training Guide*, review the information in Volume 1, Usage, before beginning to write your own runs. Then use Volume 2, Run Statements, for reference. Ask your MAPPER system coordinator for the location of example runs on your system, such as DEMO, EDIT, and MARK.

Syntax

The format of a MAPPER run statement consists of these conventions:

- The call is capitalized (for example, CHG). However, you can type it in either uppercase or lowercase letters.
- Fields and subfields are italicized whenever they call for variable data. Variable data is information you supply according to the explanation that follows the statement.

  Note: *Field and subfield abbreviations are listed and described in Appendix A.*

- Fields or subfields enclosed in brackets are optional. In this example, field2, subfield1, and subfield2 are optional:

  field1 **[*field2 subfield1,subfield2*]**

  Whenever you make an entry in an optional subfield, you must type all intervening commas. For example:

  @aux,0,b,2,123,cop,y,,,y,,2 .

- Braces around items separated by a vertical bar mean that you may choose from among the items listed, for example:

  {*item1* | *item2*}

Special Characters

The following characters have special meanings in this document:

[ ] Used around optional entries. If you select the entry, do not type the brackets, just the character or characters inside them.

| Represents the tab character in some screens and formats. Otherwise, the tab is displayed as a center dot (·).

▮ Represents the cursor on your screen. This shows where you type information.

♦ Represents the start-of-entry (SOE) character on your screen.

{ } Used to show a selection of required fields.

| Delimits items enclosed in brackets or braces. You choose one of the items to enter, but you do not enter the brackets, braces, or vertical bars.

Δ Indicates a required space in the format of a command. This character is used where the spaces in the format are not shown clearly.

System-Specific Information

Because MAPPER software is available on many different hardware systems, certain portions of this document may contain information that applies only to one system. One library is used for all systems for these reasons:

- Seeing information that differs with each system helps you port applications from one system to another.
- Having one set of manuals rather than a separate set for each system reduces the cost to you.

This system-specific information can be recognized by its green color, a triangle placed next to the text, and a phrase identifying which hardware system the information applies to.

The following example is a piece of text that applies only to the BTOS II MAPPER System:

▽ **BTOS:** Only six languages (1-6) are supported.

Abbreviations for Different Hardware Systems

The following list includes the abbreviations used next to triangles in the MAPPER library, along with the full names of the hardware systems.

Abbreviation	System Name
A Series	A Series MAPPER System
BTOS	BTOS II MAPPER System
1100	OS 1100 MAPPER System
PC MAPPER	Personal Computer MAPPER System
U Series	U Series MAPPER System
UNIX®	MAPPER systems that run on UNIX systems not supplied by Unisys

Examples

The examples in this document appear in lowercase bold letters. Unless otherwise specified, you can type them in lowercase or uppercase letters.

Whenever possible, examples are created and tested against the demonstration database that comes with your system. You can try these examples yourself and receive the same results as shown in this document.

Caution

Be careful when trying examples that update the demonstration database. Do not use functions that add, delete, or change lines while working with report 2B0, 1C0, or 1D0. Use duplicate reports so other users start with the same basic information and get the same results as shown in this document.

UNIX is a registered trademark of AT&T Information Systems.

Screen Illustrations

To help you understand concepts and procedures, this guide uses screen illustrations that closely resemble actual screen displays. However, you may encounter some differences between the illustrations and screen displays. These differences are due to variations among MAPPER systems, among terminal types, and among characteristics of the databases of each MAPPER system. These are some of the differences you might find:

- Vertical bars (|) shown between fields in screen illustrations may or may not appear on your screen display, depending on your MAPPER system and terminal settings. Also, the SOE character (♦) may be a different symbol on your screen.
- Function key bars may contain different key names. For example, if you are using an OS 1100 MAPPER System, the SOE key does not appear on the function key bar.
- Key names within screen text may be different. For example, a screen illustration might mention the **Transmit** key; on the actual screen, it is displayed as XMIT, F5, or whichever key is applicable to your terminal.
- Title lines and date lines may be positioned differently and contain different information. For example, on an OS 1100 MAPPER System, the title line might begin with an indicator called a save flag (such as @991231); save flags are not shown on the screen illustrations in this document.
- The report numbers, dates, times, and user-ids shown on screen illustrations may be different from those you actually see on your screen.

Obtaining Online Help for Keys

To obtain a list of some of the most common generic key names used in the documentation, press **KeyHlp** from the first screen that appears when you enter the MAPPER system.

Some MAPPER systems also provide a list of actual key sequences to use on your keyboard for various key actions. If this feature is available, you will see an entry in the first **KeyHlp** screen called **KeyMap**. Press the keys shown to the right of the word **KeyMap** to display this list of actual key sequences.

If **KeyMap** does not appear on the first **KeyHlp** screen, specific information for your keyboard is not available online. Refer to the printed documentation for your terminal.

You can also use the following methods to obtain the list of generic key names:

Method 1

1. Press **Help** until you reach the main help menu.
2. Tab to "Key Assignments for Documentation Key Names."
3. Transmit.

Method 2

1. After signing on to the MAPPER system, move the cursor to the control line.
2. Type **help,key**.
3. Transmit.

Uppercase Letters

These items appear in uppercase letters:

- MAPPER function and run statement calls (for example, CHG)
- MAPPER runs (for example, NAME run)
- Reserved words (for example, DATE1$)

Italics

Italics indicate the following values that you supply:

- The italicized letter N (*n* or *N*) stands for a numeral (*nn* stands for two digits, *nnn* for three, and so on).
- Other italicized letters (for example, *x* and *y*) indicate user-supplied variables.

Color

The color blue is used to indicate formats and anything you are instructed to type. For example:

To start the ICAL run, enter `ical` on the control line.

The color green, combined with a triangle in the margin, indicates that the information is specific to a certain system. The color stops at the end of the system-specific information. For example:

BTOS: Only six languages (1-6) are supported.

Organization

This manual is divided into two volumes. Volume 1: Usage, contains Sections 1 through 6, and seven appendixes, and Volume 2: Run Statements, contains Section 7.

Section 1. Overview of MAPPER Run Design

This section provides a brief description of MAPPER runs, explaining what they are, why they are used, and how you use them.

Section 2. Formulating Run Statements

This section introduces you to run statement formats and guidelines. It also explains how to use labels and special characters in run statements.

Section 3. Using Data Naming

This section describes data naming and how to use it.

Section 4. Variables, Reserved Words, and Constants

This section describes variables and reserved words and explains how to use them in runs. It also includes information on how to use predefined constants.

Section 5. Getting Online Assistance

This section describes how to use the HELP run and other online run design aids.

Section 6. Designing and Debugging Runs

This section discusses MAPPER features used to build runs (including planning, registering, and debugging runs) and methods for writing runs more efficiently and easily.

Appendix A. Summary of Statements and Options

This appendix contains a list of all run statements along with the field and subfield abbreviations used in them. It also contains a summary of options for some commonly used MAPPER statements.

Appendix B. Reserved Words

This appendix provides a table of all reserved words.

Appendix C. Control Commands for Transferring Data

This appendix lists the data control commands available that let you control the format of the data in the files being transferred.

Appendix D. Character Sets and Sorting Orders

This appendix contains tables of character sets and sorting orders. It also explains character set processing on the OS 1100 MAPPER System.

Appendix E. Using Your Keyboard and Mouse

This appendix describes keys available on the function key bar, keyboard key names used in the documentation, and use of the MAPPER software mouse.

Appendix F. Developing Source-Protected Applications

This appendix describes application development features of the MAPPER system.

Appendix G. Setting up Local Code in Your MAPPER System

This appendix describes how you can add MAPPER functions to your own site and then access them.

Section 7. Run Statements

This section presents the MAPPER run statements alphabetically, with formats, field descriptions, and examples. Some MAPPER statements are documented in the online help system only. These are flagged with an asterisk (*) in Table 7-1.

Related Product Information

Because MAPPER software contains features for every kind of user, there is an entire library of manuals available. But that does not mean every manual is right for you. Which manuals you need depends on the way you use the software. For example, to perform basic MAPPER functions, you need only the *Manual Functions Training Guide*, but to configure and manage the system for a group of users, you need several manuals.

The following chart shows which manuals fit your needs. For some tasks, several manuals are marked. Before performing these tasks, you need to understand the material in all the manuals marked.

Note: *Some of the documents in this chart may not be included in the MAPPER library for your system. See "For More Information" for instructions on how to obtain a list of documents in your MAPPER library.*

If you want to ↓ / Then read →	Manual Functions Training Guide	Manual Functions Reference	Run Design Training Guide	Run Design Reference	Color Graphics Guide	Word Processing Guide	Installation Guide	Administration Guide
Read an overview of MAPPER software	●							
Learn to use fundamental MAPPER functions	●							
Design a database and create reports	●							
Know about manual functions and how to use them	●	●						
Learn the fundamentals of run design	●		●					
Know about run statements and how to use them	●	●	●	●				
Design a run-driven application	●	●	●	●				
Learn how to use MAPPER color graphics	●				●			
Learn how to use MAPPER word processing						●		
Install MAPPER software							●	●
Configure and manage the MAPPER system	●	●	●	●	●	●	●	●

For More Information

To obtain a list of the complete title, the document number, the short title, and a brief description of each manual in the MAPPER library and any other related Unisys documents, use one of these methods:

Method 1

1. Press **Help** until you reach the main help menu.
2. Tab to "Documents in the MAPPER Library."
3. Transmit.

Method 2

1. After signing on to the MAPPER system, move the cursor to the control line.
2. Type **help,doc.**
3. Transmit.

For a List of New Features

To obtain a list of new features for this release of MAPPER software, use one of these methods:

Method 1

1. Press **Help** until you reach the main help menu.
2. Tab to "New Features and Enhancements for this Release."
3. Transmit.

Method 2

1. After signing on to the MAPPER system, move the cursor to the control line.
2. Type **help,new.**
3. Transmit.

Contents

Volume 1: Usage

About This Manual v

Section 1. Overview of MAPPER Run Design

What Is a Run? 1-2
Why Use Runs? 1-2
Run Statements 1-3
Run Control Reports 1-3
Creating and Executing Runs 1-4
Creating a New Run 1-4
Executing a Run 1-4
Starting a Run 1-5
Stopping a Run 1-5
Handling Errors 1-5

Section 2. Formulating Run Statements

Run Statement Format 2-2
Using Multiple Statements 2-3
Example of a Run Statement 2-4
Guidelines for Formulating Run Statements 2-5
Entering Statements 2-5
Terminating Lines 2-6
Using Reserved Words in Run Statements 2-6
Examples Using Reserved Words 2-7
Using the Output Area 2-8
Using Labels in Runs 2-9
Relative Line Numbers in the Lab (Label) Subfield 2-10
Using a Label Table 2-10
Specifying Reports to Process 2-12

Specifying Columns to Process 2-13
Maximum Number of Report Fields 2-13
Using Special Characters 2-14
Specifying Multiple Parameters in Run Statements 2-14
Separating Multiple Expressions and Decisions 2-15
Using Multiple Lines for One Run Statement 2-15
Denoting Literal Data in Run Statements 2-16

Section 3. Using Data Naming

Naming Cabinets, Drawers, and Reports 3-2
Names in Variables 3-3
Naming Results 3-3
Naming Fields 3-4
Report Headings and the Heading Divider Line 3-4
Field Names 3-5
Field Order 3-6
Field Names in Variables 3-6
Naming Partial Fields 3-6
Field Size Variable Definition 3-7
Converting to Field Names 3-8
Selecting Fields to Display 3-8
Efficiency Considerations 3-8

Section 4. Variables, Reserved Words, and Constants

Some Terms You Should Know 4-2
What Is a Variable? 4-2
What Is an Array? 4-2
What Is a Reserved Word? 4-2
What Is a Constant? 4-3
Selecting a Variable Name 4-4
Named Variables 4-4
Numbered Variables 4-4
Named Vs Numbered Variables 4-5
Using Named and Numbered Variables on OS 1100 MAPPER Systems 4-5

Assigning Names to Numbered Variables 4-5
Converting Named Variables to Numbered Variables 4-5
Understanding Variable Types and Sizes 4-6
Default Justification of Variables 4-8
No Type Verification 4-8
Selecting a Variable Type and Size 4-9
Determine the Size 4-9
Defining and Initializing Variables 4-10
Methods for Defining a Variable 4-10
Using the LDV Statement 4-11
Using the CHG Statement 4-11
Using Other Run Statements 4-11
Redefining Variables 4-11
Using Variables in Run Statements 4-12
Referring to Substrings 4-12
Trailing Characters 4-12
Unknown Number of Trailing Characters 4-13
Using Contents to Act as Name or Number 4-13
Using Scientific Notation 4-14
Changing the Contents of Variables 4-15
Using the LDV Statement 4-15
Using the INC and DEC Statements 4-15
Using the ART Statement 4-15
Using the CHG Statement 4-16
Testing the Contents of Variables 4-17
Using the DEF Statement 4-17
Using the IF Statement 4-17
Using the LCV Statement 4-17
Using the VARIABLE Run to Check Contents 4-18
Using a Variable Table 4-19
Using the BVT Run to Build Variable Tables 4-19
Working with Variable Limits 4-22
Maximum Number of Variables Allowed 4-22
Run Registration 4-22
Maximum Number of Characters 4-23
String Space Limits 4-23

Techniques for Handling Variable Limits 4-23
Writing Modular Subroutines 4-23
Passing Variable Arrays 4-24
Clearing Numbered Variables 4-24
Using the Variable Stack 4-24
Capturing Input 4-26
Screen Input 4-26
Initial Input Parameters 4-26
Prompting the User for Input 4-26
Using the Output Area 4-27
Accepting User Input 4-27
Using INPUT$ 4-28
Using INSTR$ 4-29
Using INVAR$ 4-30
Using INVR1$ 4-31
Using INMSV$ with the OUM Statement 4-31
Using ICVAR$ with the CHD Statement 4-32
Using FKEY$ with the KEY and CHD Statements 4-32
Capturing Initial Input Parameters 4-33
Passing Input to the Run 4-33
Examples Using Variables 4-35
Using Reserved Words 4-38
Reserved Words in the Output Area 4-38
Using Predefined Constants 4-39
Defining Constants 4-39
Including Defined Constants 4-43

Section 5. Getting Online Assistance

Displaying Information about MAPPER Functions (HELP) 5-2
Creating, Modifying, and Removing APT Tables (BAT) 5-6
Creating Input Screens and Menus (SCGEN) 5-8
Displaying Horizontal Column Positions (CC) 5-10
Displaying Field Column-Characters (FCC) 5-11
Displaying Run Statement Formats (FORM) 5-12
Using a Function Mask to Get Format (FORMC) 5-13
Creating Run Statements (MARS) 5-14

Generating Runs (RUN) 5-15
RUN Run Function Key Bar 5-16
Displaying Reports and Results in Your Generated Run 5-16
Registering Run Statements as a Run 5-17
RUN Run Limitations 5-17
Analyzing Your Run (RUNA) 5-19

Section 6. Designing and Debugging Runs

Planning and Registering Your Run 6-2
Handling Reports and Results 6-5
The Output Area 6-6
Results 6-7
The Relationship Between Output Area and Results 6-7
Debugging Your Run 6-8
Interactive Debugging 6-8
Online Help System 6-8
Checkpoint Displays 6-9
Run Debug (RDB) Statement 6-9
Register Error Routine (RER) Statement 6-9
Using Conditional Statements 6-10
Branching 6-10
Conditional Branching 6-10
Looping 6-11
Using Subroutines 6-12
Writing Efficient Runs 6-13
Run Control Reports 6-13
Analysis and Registration 6-14
Loading Variables 6-15
Statements and Functions 6-15
Updating Reports 6-16
Logic 6-17
Batch Processing 6-18

Appendix A. Summary of Statements and Options

Run Statements A-2
Reminders for Run Statement Syntax A-2
Run Statement Formats A-3

Field and Subfield Abbreviations A-10
Options for 10 Common Run Statements A-17

Appendix B. Reserved Words

Reserved Words in the Output Area B-2
Commonly Used Variable Types and Sizes of Reserved Words B-2

Appendix C. Control Commands for Transferring Data

Data Control Commands C-2

Appendix D. Character Sets and Sorting Orders

ASCII Character Set D-2
Fieldata (Limited Character Set) D-5
Using the C(S) Option D-7
Character Set Processing on the OS 1100 MAPPER System D-9
LCS, FCS, and FCSU D-9
Character Hierarchy D-9
Using the C(x) Option D-10
Using the C(L) and C(F) Options D-10

Appendix E. Using Your Keyboard and Mouse

Function Keys E-2
Online Help for Keys E-7
Use of the MAPPER Mouse E-8
Installing the Mouse on Your PC E-8
Guidelines E-9

Appendix F. Developing Source-Protected Applications

Developing an Application F-1
How to Design a Run-Timed Application F-1

Appendix G. Setting up Local Code in Your MAPPER System

Writing Your Functions G-1
Accessing Your Functions G-2

Volume 2: Run Statements

Section 7. Run Statements

Run Statements by Name 7-2
Run Statements by Topic 7-8
Locating, Changing, Comparing, Sorting, and Reformatting Data 7-8
Manipulating Reports and Results 7-9
Manipulating Screen Displays 7-10
Obtaining Information about the Database, Display, and Runs 7-11
Calculating 7-12
Printing Results or Reports 7-13
Manipulating Variables 7-13
Controlling Runs 7-14
Manipulating and Updating Report Lines 7-16
Sending and Receiving Messages, and Handling Devices 7-17
Interfacing the Operating System, Other Applications, and Other MAPPER Systems 7-18
ADD (Append Report) ..... Online
ADR (Add Report) ..... Online
ART (Arithmetic) 7-23
Operators 7-24
Formulating Arithmetic Expressions 7-24
Multiple Expressions 7-25
Internal Computation 7-25
Negative Numbers 7-26
Changing the Hierarchy of Expressions 7-26
Arithmetic and Trigonometric Functions 7-27
ASR (A Series Run) 7-28
AUX (Auxiliary) 7-31
BFN (Binary Find) 7-33
Using the IBFN Run to Create a BFN Statement 7-39
BLT (Build Label Table) ..... Online
BR (Background Run) 7-40
BR (Background Run): OS 1100 7-42
BRG (Break Graphics) ..... Online

BRK (Break) 7-44
CAB (Cabinet Switch) ..... Online
CAH (Cache Report) ..... Online
CAL (Calculate) 7-46
Priority of Operations 7-50
True/False Conditions 7-52
Date and Time Processing 7-54
Character String Processing 7-54
Using the ICAL Run to Create a CAL Statement 7-54
CAL Statement Examples 7-55
CALL (Call Subroutine) 7-58
CAR (Clear Abort Routine) ..... Online
CAU (Calculate Update) ..... Online
CER (Clear Error Routine) ..... Online
CHD (Command Handler) 7-65
CHG (Change Variable) 7-68
Using Expressions 7-69
Guidelines for Addition and Subtraction 7-69
Processing Octal Variables 7-70
CLK (Clear Link) ..... Online
CLT (Clear Label Table) ..... Online
CLV (Clear Variables) ..... Online
CMP (Compare Report) 7-72
CMU (Commit Updates) ..... Online
CNT (Count) 7-78
Using the ICNT Run to Create a CNT Statement 7-84
CNT Statement Example 7-85
CPY (Copy) ..... Online
CSR (Clear Subroutine) ..... Online
DAT (Date) 7-86
Using the IDAT Run to Create a DAT Statement 7-90
DC (Date Calculator) 7-91
Date and Time Formats 7-92
DCPY (DDP Copy) ..... Online
DCR (Decode Report) 7-95
DCRE (DDP Create) ..... Online
DCU (Decommit Updates) ..... Online
DDI (Data Definition Information) ..... Online

DEC (Decrement Variable) ..... Online
DEF (Define) 7-97
DEL (Delete) ..... Online
DEV (Device) ..... Online
DFU (Defer Updates) ..... Online
DIR (Directory) 7-100
DIS (Diskette) ..... Online
DLL (Downline Load) ..... Online
DLR (Delete Report) ..... Online
DPUR (DDP Purge) ..... Online
DRW (Drawer) 7-102
DSF (Display Form) 7-320
DSG (Display Graphics) ..... Online
DSM (Display Message) 7-104
DSP (Display Report) 7-107
DSX (Display Report and Exit) ..... Online
DUP (Duplicate Report) ..... Online
DVS (Define Variable Size) 7-110
ECR (Encode Report) 7-112
EL- (Element Delete) 7-115
ELT (Element) 7-117
ESR (Exit Subroutine) ..... Online
EXT (Extract) ..... Online
FCH (Relational Aggregate Fetch) ..... Online
FDR (Find and Read Line) 7-121
FIL (Create File) 7-125
FKY (Function Key) 7-129
FMT (Format) 7-131
FND (Find) 7-133
Using the IFND Run to Create an FND Statement 7-137
GOC (Generate Organization Chart) ..... Online
GS (Graphics Scaler) ..... Online
GTO (Go To) 7-138
HOF (Host Sign-off) ..... Online
HRD (Host Read) ..... Online
HRN (Host Run) ..... Online
HSH (Hash) 7-140

HST (Host Sign-on) ..... Online
HWR (Host Write) ..... Online
IDU (Index User) 7-141
IF (If Conditional) 7-143
INC (Increment Variable) ..... Online
IND (Index) 7-148
INS (Insert Variable) ..... Online
ITV (Input Variable) 7-149
JUV (Justify Variable) 7-151
KEY (Function Key Input) 7-154
LCH (Locate and Change) 7-155
LCV (Locate and Change Variable) 7-158
LDA (Load Variable Array) 7-166
LDV (Load Variable) 7-169
Loading Multiple Variables 7-171
Packing Variables 7-172
Loading Variables Based on Content 7-172
LDV Examples with Various Options 7-173
LFC (Load Format Characters) 7-175
LFN (Load Field Name) 7-176
LGF (Logoff Relational System) ..... Online
LGN (Logon Relational System) ..... Online
LLN (Last Line Number) ..... Online
LMG (List Merge) ..... Online
LN+ (Add Line) ..... Online
LN- (Delete Line) ..... Online
LNA (Append Line) ..... Online
LNG (Language) ..... Online
LNI (Insert Line) ..... Online
LNK (Link to Another Run) 7-178
LNM (Move Line) ..... Online
LNP (Put Line) ..... Online
LNX (Duplicate Line) ..... Online
LNY (Yank Line) ..... Online
LOC (Locate) 7-180
LOG (Log for Analysis) ..... Online
LOK (Update Lock) ..... Online
LSM (Load System Message) 7-183

LZR (Line Zero) 7-184
MAU (Match Update) ..... Online
MCH (Match) 7-187
MSG (Message to Control) 7-192
NET (Network Sign On) 7-194
NOF (Network Off) 7-196
NRD (Network Read) 7-197
NRM (Network Remote) 7-198
NRN (Network Run) 7-200
NRT (Network Return) 7-202
NWR (Network Write) 7-203
Receiving Report Entries 7-203
OK (Acknowledge Message) 7-205
OS2 (Operating System Interface) 7-207
OUM (Output Mask) 7-209
OUT (Output) 7-214
Displaying Information At Another Terminal 7-220
If the User Is At the Terminal 7-221
If the User Is Not at the Terminal 7-221
Four- and Five-to-One Output 7-221
Color Characters 7-226
Emphasis Characters 7-227
Four-to-One Screens 7-228
OUV (Out Variable) 7-231
PEK (Peek Variables) ..... Online
PNT (Refresh Screen) ..... Online
POK (Poke Variables) ..... Online
POP (Pop Variables) ..... Online
PRT (Print) 7-232
PSH (Push Variables) ..... Online
QCTL (Queue Control) ..... Online
QREL (Release Message) ..... Online
QRSP (Send Response Message) ..... Online
QSND (Send Message, No Response) ..... Online
QSNR (Send Message, Expect Response) ..... Online
RAM (Relational Aggregate Modify) ..... Online
RAR (Register Abort Routine) 7-235
RDB (Run Debug) 7-238
RDB Commands and Function Keys 7-239

RDB (Run Debug): OS 1100 7-244
RDB Commands and Function Keys: OS 1100 7-245
RDC (Read Continuous) 7-250
RDL (Read Line) 7-254
REH (Retrieve from History) 7-257
REL (Release Display) ..... Online
REP (Replace Report) ..... Online
RER (Register Error Routine) 7-258
RET (Retrieve File) 7-261
RET (Retrieve File): OS 1100 7-263
RETURN (Return Call Routine) ..... Online
RFM (Reformat Report) 7-265
RLN (Read Line Next) 7-267
RMV (Remove Variables) ..... Online
RNM (Rename) ..... Online
RPW (Read Password) 7-270
RRN (Remote Run) ..... Online
RS (Run Status) 7-272
RSI (Remote Symbiont Interface) 7-273
RSL (Create Result Copy) ..... Online
RSR (Run Subroutine) 7-275
RTN (Return Remote) ..... Online
RUN (Run Start) 7-278
SC (Screen Control) 7-280
Reserved Words 7-284
Error Status Reserved Words 7-285
Screen Commands 7-285
Cursor Control Commands 7-288
Screen Editing Commands 7-289
Field and Attribute Commands 7-290
Text Handling Commands 7-307
Screen Printing Commands 7-312
Setup Commands 7-312
Special FKEY Actions 7-316
Displaying a Form and Returning Control to the Run 7-320
SCH (Schedule) 7-324
SEN (Send Report) 7-325

SFC (Set Format Characters) 7-327
SNU (Send Report to User) 7-328
SOR (Sort) 7-330
Using the ISOR Run to Create an SOR Statement 7-332
SQL (Submit SQL Statement) ..... Online
SRH (Search) 7-333
Using the ISRH Run to Create an SRH Statement 7-337
SRR (Sort and Replace Report) ..... Online
SRU (Search Update) ..... Online
STN (Station Information) 7-338
STR (Start) 7-340
IBM Start Requirements 7-342
SUB (Subtotal) 7-343
 **TCS (Tape Cassette)** ..... Online
TOT (Totalize) 7-347
Using the ITOT Run to Create a TOT Statement 7-351
TRC (Trace) ..... Online
ULK (Unlock) ..... Online
UNX (UNIX Interface) 7-352
UPD (Update) ..... Online
USE (Use Variable Name) 7-355
WAT (Wait) 7-356
WDC (Word Change) ..... Online
WDL (Word Locate) ..... Online
WPR (Word Process) ..... Online
WRL (Write Line) 7-358
XCH (Exchange Variables) ..... Online
XIT (Sign off MAPPER Software) ..... Online
XQT (Execute) 7-360
XUN (Exit MAPPER System) 7-363
***cmd (Local Code)** 7-364

Glossary

Index

Figures

4-1.	Fields in Variable Tables	4-20
6-1.	Sample Flow Chart	6-2
6-2.	Processing a Report or Result	6-5
6-3.	One Run Using Several Reports and Results	6-6
6-4.	Relationship between Output Area and Results	6-7

Tables

4-1.	Variable Types and Sizes	4-7
4-2.	Default Justification	4-8
A-4.	Options for 10 Common Run Statements	A-18
B-1.	Reserved Words	B-3
C-1.	Data Control Commands	C-2
D-1.	ASCII Character Set	D-2
D-2.	Limited Character Set	D-5
D-3.	Sorting with the C(S) Option	D-8
D-4.	Sorting without the C(S) Option	D-8
D-5.	LCS Order Using the C(x) Option	D-11
D-6.	FCS Order Using the C(x) Option	D-13
E-1.	Function Key Descriptions	E-2
E-2.	Screen Location of Keys on the Function Key Bar	E-6
7-1.	Run Statements by Name	7-2
7-2.	ART: Arithmetic Operators	7-24
7-3.	ART: Priority of Arithmetic Operations	7-25
7-4.	ART: Arithmetic and Trigonometric Functions	7-27
7-5.	CAL: Priority of Arithmetic Operations	7-50
7-6.	CAL: Priority of Relational Operations	7-51
7-7.	CAL: AND and OR True/False Conditions	7-52
7-8.	Characteristics for the Expanded M	7-223
7-9.	Characteristics for the Expanded N	7-225
7-10.	Five-to-One Color Codes	7-227
7-11.	Emphasis Characters	7-228

7-12.	RDB Commands and Function Keys	7-239
7-13.	RDB Commands and Function Keys: OS 1100	7-245
7-14.	Cursor Control Commands	7-288
7-15.	Screen Editing Commands	7-289
7-16.	Special Commands	7-307

Section 1
Overview of MAPPER Run Design

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

This manual contains reference information about new and existing run statements. By using this manual, you should be able to write and update complete runs, obtain quick access to statement syntax, and learn efficient techniques by following examples.

This section includes:

- What is a run?
- Creating and executing runs

What Is a Run?

A run is a series of statements in a report that specify step-by-step instructions for MAPPER software to follow. Available capabilities include the following:

- Generating reports and results
- Executing other MAPPER operations
- Creating unique report formats
- Making logical decisions
- Manipulating data in MAPPER reports

You type run statements in a run control report. Runs are identified by their names. To initiate a run, enter the run name on the control line of your screen; MAPPER software automatically interprets statements starting at line 3 of the run control report and carries out the instructions contained in the run.

Ask your MAPPER system coordinator for the location of example runs on your system, such as DEMO, EDIT, and MARK.

Why Use Runs?

Runs replace a series of manual functions that have to be performed repeatedly. They reduce the risk of input mistakes and save time. Runs also let you do some things that are impossible to do using only manual operations.

Run Statements

Runs are composed of run statements, each instructing MAPPER software to perform a specific operation. Each must be written in a certain format and must follow certain conventions, as described in Section 2.

In general, each statement does three things:

- Identifies the operation to be performed
- Identifies the data on which the operation is to be performed
- Specifies the manner in which the results are to be presented

Each statement uses a run function call, a short abbreviation used to request an operation (for example, SRH for the Search function). These calls are listed and described in detail in Section 7.

Run Control Reports

Run statements are composed and stored as a run in a run control report. Each report stores one run.

Each MAPPER cabinet can contain one or more drawers that are reserved for run control reports, depending on the needs of your department. These reports are similar to other MAPPER reports; they have a control line and a date line, and you can update their contents using manual updating functions.

As with other MAPPER reports, run control reports are identified by their cabinet number, drawer letter, and report numbers. Refer to the *Run Design Training Guide* for sample run control reports.

Creating and Executing Runs

This section describes how to create a new run and how to execute it.

Creating a New Run

Creating a new run involves the following steps:

1. Planning the run so it solves the problem or performs the desired operation effectively and efficiently
2. Naming and registering a run control report, where the run statements are written and tested
3. Writing the individual run statements that constitute the run
4. Testing and debugging the run statements until the run performs the desired operations without error
5. Submitting the run to the MAPPER system coordinator (a person who manages MAPPER software) for analysis and registration

Executing a Run

Runs are executed from the control line, just like manual functions. If the run contains an error, it stops and displays the system message and the run control report where the error was encountered, provided you are the author of the run. You can also interrupt the run if you need to.

The basic instructions for starting and stopping a run and handling errors that cause a run to stop during its execution are described in this section.

Starting a Run

To start a run, enter the run name in any position on the control line of any report or on the active screen. Note that if a run name is the same as a manual function call, the manual function is executed, not the run. See the MAPPER system coordinator for more information.

Note: *If you receive a message informing you MAPPER software is unable to process your request, the run name is either incorrect or is not registered correctly.*

Stopping a Run

The run stops automatically at the end of its execution or whenever it encounters an error.

To stop the run manually during its execution, press **Abort** (or the equivalent key on your keyboard). See Appendix E for information on key names used in this manual.

Handling Errors

Four things happen when an error is encountered during a run:

- The run stops.
- The portion of the run containing the error is displayed if you were the last person to update the run.
- A system message appears in the control line of the display.
- All results are no longer accessible unless a Register Error Routine (RER) subroutine is in the run, or unless you are executing the run in Run Debug (RDB) mode. See RER and RDB in Section 7.

To correct the error:

- Read the system message. Press the **Help** function key to obtain online information about the error for which the message is displayed. See Section 5 for more information about online documentation.
- At this point, you can press **Help** again for more detailed online information or press the **Return** function key to redisplay your run.
- Make the necessary modifications to your report. When it is corrected, you can execute it again.

Section 2
Formulating Run Statements

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

In run control reports, lines containing MAPPER run statements begin with an at sign (@) and follow a specific format. This section describes how to formulate run statements, how to use labels and label tables, and how to use special characters, for example, for specifying multiple parameters, in your runs.

This section includes:

- Run statement format
- Guidelines for formulating run statements
- Using labels in runs
- Specifying reports to process
- Specifying columns to process
- Using special characters

Run Statement Format

Run statements are instructions you enter into a run control report to create a run. A line that contains run statements begins with an at sign (@) and follows a specific format. The following is a common MAPPER run statement format:

```
@label:call,report options fields line-type,
parameters variables . comment
```

Following is a description of each portion of the sample statement format:

Item	Description
@	A symbol indicating the beginning of a run statement line.
label	A number identifying a statement line in a run. See "Using Labels in Runs" in this section.
call	The run function call, which is an abbreviation for the full name of the operation.
report	The cabinet, drawer, and report number or name to process. See "Specifying Reports to Process" in this section.
options	Options for specifying how to perform the operation. To see a list of options available for a particular statement, press **Op&Prm** while in the online help topic for that function or run statement.
fields	The report fields to process. Specify either the column-character positions (such as 2-2) or the field name (such as 'StCd'). See "Specifying Columns to Process" in this section.
line-type	The line type to process.
parameters	Items of information that indicate specific values or fields to process. To see a list of parameters available for a particular statement, press **Op&Prm** while in the online help topic for that function or run statement. In statements that perform searches, such as Find (FND), you can place a slant (/) within apostrophes in this subfield to process a partial field.

continued

continued

Item	Description
variables	Variables to capture specific information. See Section 4 for more information.
. comment	A space, a period, and another space, followed by an optional comment about the run statement line. The MAPPER system stops processing the line when it encounters a period; therefore, you should use a period after each run statement line.

Using Multiple Statements

You can enter multiple statements on the same line. In these cases, enter the at sign only on the first statement on the line. Enter the space-period-space and optional comment only after the last statement on the line. For example:

```
@if user$ = jdoer gto 002 ; gto 001 . check user
```

Example of a Run Statement

This Search (SRH) statement uses data from report 1C0:

```
@007:srh,0,c,1 d 'producttype' □,blackbox7 <finds>i5,<lines>i6 .
```

Following is a description of each portion of the sample SRH statement:

Item	Description
@	Symbol indicating a run statement line.
007:	Label (007) and separator (:).
srh	Run function call.
0,c,1	Cabinet number, drawer letter, and report number.
d	D option.
'producttype'	Field to search.
□	Line type to process. (You can enclose the tab character in apostrophes to remind yourself it is there if it does not display on your screen.)
blackbox7	Parameter indicating the value to search for.
<finds>I5	Variable to capture the number of finds.
<lines>I6	Variable to capture the number of lines searched.

Guidelines for Formulating Run Statements

This section describes guidelines to follow when formulating run statements.

Entering Statements

Follow these guidelines when entering run statements:

- Type an @ in column 1.
- Enter multiple run statements on one line, separated by a space. Use just one @ per line.
- Separate fields with spaces. Separate subfields with commas.
- Optional fields are shown enclosed in brackets ([ ]) in the formats. Do not enter the brackets in your own run statements.
- Required fields are those that are not enclosed in brackets in the format. Be sure to use all required fields. Enter two apostrophes (' ') if you are not entering options in the *o* (options) field.
- If you omit any optional subfields, include commas in their places, unless they are at the end of the field.
- The maximum number of variables you can use as input to a statement is 40, unless otherwise stated in the description of the individual statement.
- You can specify fields to process in the *cc* subfield in any order, regardless of their order in the report. However, you must list the parameters in the same order. For example, the next two statements create the same result, even though the fields to process are inverted:

```
@srh,0,c,1 '' 2-5,16-1 □,black,a .
@srh,0,c,1 '' 16-1,2-5 □,a,black .
```

- In run statements that require an issuing and receiving report, designate the issuing report first and the receiving report second.
- **1100:** You cannot process your own run control report.
- Run statements that access or lock reports (such as DFU, LOK) cannot access or lock their own run control report.

Terminating Lines

The following items terminate a line:

Space-period-space
@.
@label.
DEFINE statements
INCLUDE statements
Several following run statements, for example:

Call Subroutine (CALL)	Out Variable (OUV)
Display Message (DSM)	Output (OUT)
Display Report (DSP)	Remote Run (RRN)
Execute (XQT)	Run Subroutine (RSR)
Exit Subroutine (ESR)	Screen Control (SC)
Link to Another Run (LNK)	Wait (WAT)

(If a statement terminates a line, it is noted in the documentation for that statement.)

Using Reserved Words in Run Statements

A reserved word is a system-maintained variable. See Section 4 for more information on reserved words.

You can use reserved words directly — where you might otherwise use variables — in these run statement subfields:

- *c* Cabinet
- *d* Drawer
- *r* Report
- *l* Line number
- *f* Format
- *p* Parameters

Examples Using Reserved Words

If you use the reserved word EDRW$, which contains the numeric drawer number of the run control report, you can omit the cabinet subfield. This example displays one screen from the drawer in which the run resides:

```
@out,edrw$,5,1,scnv$ .
```

To make your statement easier to read and more efficient, do not initialize a variable with the reserved word first.

You can also use substrings of reserved words when you reference them directly, just as you can use substrings of variables. For example, this statement logs the users of a run:

```
@wrl,edrw$,v10,v11 2-11,15-6,23-5 *,user$,date1$,time$(1-5) .
```

The user-id, current date, hour, and minute are written in the report in v10, on the line in v11, in the drawer in which the run resides.

Whenever you directly reference a reserved word, be sure to consider the field size. For example, in the above WRL statement, 11 characters (2-11) were allowed for the user-id to ensure enough information was written.

Use the DEF statement with the S option to find out the length of a reserved word. See DEF in Section 7.

Using the Output Area

The output area is a temporary scratch area that you build in your run control report to hold information. For more information about the output area, see Section 6.

Do not use an at sign or colon (@ or :) in the first character position of any output area line.

To specify the literal representation of variables and reserved words in the output area, enclose them within apostrophes (for example, 'v1' or 'DATE1$').

1100: On OS 1100 MAPPER Systems, the literal representation of reserved words always appears in the output area. To place the value of reserved words in the output area, load variables with the reserved words and place the variables in the output area.

Using Labels in Runs

Labels are numbers you use to identify run statement lines. Use them in these situations:

- At the start of a run statement to match a label specified in the *lab* subfield of another run statement.
- In logical If Conditional (IF) and Go To (GTO) statements for transferring control to a section of the run starting at the specified label.

Formats

@label: (followed by a run statement)

@label .

where *label* can be 1 to 199 (or 1 to 399 if your system is set up to handle 399 labels). Each label must be unique; duplicates are not allowed.

For easier run maintenance, you can use three- or four-character labels, zero-filled (for example @003 or @0003). You can also indent unlabeled lines for legibility. For example, four-character labels and six-character indenting would look like this:

```
@0001:inc <nameit>
@     if <nameit> gt 5,(0199)
            .
            .  (more run statements)
            .
@0199:dsp,-0 .
```

This convention does make it easier to maintain runs; however, it is slightly less efficient.

Relative Line Numbers in the Lab (Label) Subfield

In addition to using label numbers in the *lab* subfield of run statements, you can specify a relative line number to go to.

Format

```
LIN[+]n
```

where [+]*n* jumps to the line that is equal to the current line plus *n* lines. When the system encounters continued lines (reverse slant [\] at the end of a line), it considers the current line as the last line of the continuation sequence.

Specify *n* using an integer or using a variable that contains an integer. If the variable contains a negative number, execution continues at the current line minus the absolute value of the variable.

For easier maintenance, avoid jumping more than one or two lines forward, and avoid jumping backward (**LIN-***n*).

Using a Label Table

When a run statement contains a label, MAPPER software may have to search each line in a report for the label. To speed up the search, you can build label tables that point directly to the line you want.

To build a label table, use the Build Label Table (BLT) function or the BLT run statement. Note that the BLT function and run statement create a result.

The lines in a label table tell the system which lines contain labels. The BLT function or run statement automatically inserts these definition lines in your run control report in the following format:

```
:Llabel-number=line-number, ...
```

Example

```
:L22=13,33=26,44=39
```

Use label tables only in runs that have been debugged and tested. Also, use them only in larger, more stable production runs, but not in runs that change frequently.

When you add or delete lines from a run control report, simply rebuild the label table with the BLT function to reflect the line changes.

Specifying Reports to Process

Many run statement formats contain a *c,d,r* field in which you specify the cabinet, drawer, and report to process. In this field, you can use the cabinet number, drawer letter, and report number, or a data name.

In some run statements, one or more of the *c,d,r* subfields is optional. This statement uses all three subfields and processes report 1D0:

```
@srh,0,d,1 '' 'cust code' □,amco .
```

This example uses only the *c* (cabinet) and *d* (drawer) subfields:

```
@srh,0,b '' 'producttype' □,blackbox7 .
```

To specify a data name, use the name of a cabinet, drawer, or report as defined in the system directory. Enclose the data name in apostrophes. This example uses a report named Order Status:

```
@srh,'order status' '' 'cust code' □,amco .
```

For more information about using data names, see Section 3.

Specifying Columns to Process

Many run statement formats contain a *cc* field in which you specify the columns of data to process. In this field, you can use column-character positions or field names.

To specify column character positions, indicate the starting column number and the number of characters to process. For example, this statement searches the 4-character field beginning in column 26:

```
@srh,0,d,1 '' 26-4 □,amco .
```

To use field names, use the names from the report headings (enclosed in apostrophes). This example uses field names and performs the same operation as the previous example:

```
@srh,0,d,1 '' 'cust code' □,amco .
```

For more information about using field names, see Section 3.

1100: In most run statements, you cannot define adjacent or overlapping fields (one exception, for example, is the Read Line [RDL] statement).

Maximum Number of Report Fields

The maximum number of report fields you can process is 64, unless otherwise stated in the description of the individual statement.

1100: On OS 1100 MAPPER Systems, the maximum number of report fields you can process is 39, unless otherwise stated in the description of the individual statement.

Denoting Literal Data in Run Statements

Use apostrophes (') to enclose literal data containing spaces, slants, or commas in the parameters field of a run statement. (Literal data that does not contain spaces, slants, or commas do not need to be enclosed in apostrophes, unless it is a field name or data name.)

Examples

```
@srh,0,d,1 d 61-17 □,'digital corp' .

@srh,0,d,1 / 'customer' □,'union steel/sulfr' .

@srh,0,c,1 f 'produc cost' □,'13,500' .
```

Section 3
Using Data Naming

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

Data naming is a run design tool that allows you to identify fields in a run statement using the field names of a report rather than the column-character positions.

You can also assign meaningful names to reports, drawers, and cabinets. Use these names in place of report numbers, drawer letters, and cabinet numbers in both run statements and manual functions.

This section describes the required syntax for named reports, drawers, cabinets, and fields. You can mix standard run syntax and data naming syntax within the same run or even within the same run statement.

The system directory contains information about all named reports, drawers, and cabinets. To retrieve information about a particular name, use the NAMES run or the Directory (DIR) run statement. See DIR in Section 7 and the NAMES run in the online help system (HELP,NAMES). For information on updating the system directory, see the NAME run in the online help system (HELP,NAME).

This section includes:

- Naming cabinets, drawers, and reports
- Naming fields

Naming Cabinets, Drawers, and Reports

Before using a named report, drawer, or cabinet, you must enter it in the system directory with the NAME run.

Names must start with an alphabetic character (A to Z) and contain from 1 to 16 characters, which can include either uppercase or lowercase letters.

You can use any characters in your report, drawer, or cabinet names except the caret (^), semicolon (;), slant (/), comma (,), space, and tab. For example, orderstatus and order-status are both acceptable (and unique) names.

1100: On OS 1100 MAPPER Systems, you can use any characters in your report, drawer, or cabinet names. However, only alphanumeric characters (A to Z or 0 to 9) are considered when comparing the name to the system directory; all other characters are ignored. For example, orderstatus, order-status, and order status are all acceptable names and considered to be the same.

You can also assign a name to a range of reports in a drawer. With the Binary Find (BFN), Find (FND), and Search (SRH) statements, the system automatically adds the R option, which specifies a range of reports, to the run statement if you use a name that defines a range of reports.

1100: The previous paragraph also applies to the Count (CNT) statement.

A report name replaces all three run statement subfields. A drawer name replaces both the *c* and *d* subfields; a cabinet name replaces only the *c* subfield.

In the run statement, enclose the names of reports, drawers, and cabinets in apostrophes (').

For example, assuming the name orderstatus represents report 1D0, these statements both search the same report. Notice that the name replaces the *c,d,r* subfields.

```
@srh,0,d,1 ...

@srh,'orderstatus' ...
```

In this example, the statement searches report number 2 in a drawer named orders:

```
@srh,'orders',2 ...
```

Since data names vary from site to site and user to user, the formats and examples in this manual use the cabinet, drawer, and report designations (*c*,*d*,*r*).

Names in Variables

Enclose variables in apostrophes if they contain report, drawer, or cabinet names. Do not place any other characters, such as spaces, within the apostrophes.

You can use any variable type, including a variable containing the name of another variable; however, do not use substrings. See Section 4 for information on variables.

This example searches the report named orderstatus. Notice that no spaces or other characters are between the apostrophes in '<repname>':

```
@ldv <repname>h18='orderstatus' .
@srh,'<repname>'  ...
```

Naming Results

Wherever you specify the current result (-0) or a renamed result (-1 to -8), you can omit the cabinet and drawer (*c* and *d*) subfields.

For example, instead of using the first statement, you can use the second statement to do the same thing:

```
@dsp,0,b,-0 .

@dsp,-0 .
```

Naming Fields

You can use field names from report headings in run statements in place of column-character positions. The system uses the headings of the report you are processing or headings from report 0 if you process the entire drawer.

Using field names makes run statements easier to read and maintain. If a field moves within a report or the size of the field changes, you do not have to change the reference to it.

For example, these two statements perform the same search and have identical results:

```
@srh,0,d,1 '' 26-4 □,amco .
@srh,0,d,1 '' 'cust code' □,amco .
```

Report Headings and the Heading Divider Line

When a run reads the first field name in a statement, it scans the first 16 lines of the report for a heading divider line.

The run derives the starting columns and field sizes from the grouping of equal signs on the heading divider line, extracting field names from up to two asterisk lines above the line.

For example, the run derives the field names listed below from the sample headings:

Sample headings:	Field names:	Column-characters:
*St.Status.By. Product .	St Cd	2-2
*Cd. Date .In. Type .	Status Date	5-6
*==.======.==.=========.	By In	12-2
	Product Type	15-9

If the report heading contains three or more asterisk lines, the run recognizes only the last two. For example, the run derives the field names listed below from the headings shown:

Sample headings:	Field names:
* Monthly. Annual .	
*Interest.Discount.	Interest Rate
* Rate . Status .	Discount Status
*========.========.	

Therefore, if you use named fields for reports with three or more asterisk heading lines, keep the important information in the last two lines.

Field Names

When specifying field names in run control reports, enclose them in apostrophes (') and, since they are not case sensitive, enter them in either uppercase or lowercase letters. Field names can contain up to 32 characters and must be unique within the heading; if duplicate names exist, the run uses the leftmost field.

You can use any characters in your field names except parentheses. The run ignores spaces, so, for example, the run considers 'custcode' and 'cust code' identical names.

1100: On OS 1100 MAPPER Systems, the run also ignores special characters. So, for example, the run considers 'custcode', 'cust-code', and 'cust code' identical names.

You can also abbreviate field names by omitting trailing characters, as long as the character string that remains is unique to that field.

For example, to make the run statement shorter and more efficient, search the Cust Code field like this:

```
@srh,0,d,1 '' 'cust' □,amco .
```

Field Order

You can list multiple named fields in any order; they do not have to be in the same order as they appear in the report as long as the parameters are in matching order.

For example, these two statements perform the same search and have identical results:

```
@srh,0,d '' 'st cd','cust code' □,or,amco .
@srh,0,d '' 'cust code','st cd' □,amco,or .
```

Field Names in Variables

Enclose variables in apostrophes if they contain field names. Do not place any other characters, such as spaces, in apostrophes. You can use any variable type, including a variable containing the name of another variable; however, do not use substrings of variables. For more information on variables, see Section 4.

This example searches the Cust Code field for Amco. Notice that no spaces or other characters are between the apostrophes in '<fldname>':

```
@ldv <fldname>h18='cust' .
@srh,0,d,1 '' '<fldname>' □,amco .
```

Naming Partial Fields

Run statements can also use field names to scan partial fields for specific starting or ending characters. To process a partial field, enclose the relative starting column position and number of characters of the partial field in parentheses after the field name.

This statement searches the first character of the Product Type field:

```
@srh,0,d,1 '' 'product type(1-1)' □,b .
```

To process a named field from a specific column up to the end of the field, specify the starting column and define the number of characters as zero.

For example, this statement searches the Product Type field starting at the sixth column in the field for the remainder of the field:

```
@srh,0,d,1 '' 'product type(6-0)' □,box1 .
```

To process only the trailing portion of a named field, specify a starting column of zero and the number of characters to process.

This example searches the last character of the Order Number field:

```
@srh,0,d,1 '' 'order number(0-1)' □,s .
```

Field Size Variable Definition

When using an input parameter to process a report field, the variable used for input must match the field size. The input field size on a screen may also need to match a corresponding report field.

You can use the Define Variable Size (DVS) run statement to create variables equal to the size of the report fields (see DVS in Section 7). The run defines the size of the variable when it executes. Any input parameter or screen using that variable adjusts to changes in the size of the field.

With the Read Continuous (RDC), Read Line (RDL), Read Line Next (RLN), and Subtotal (SUB) run statements, you can let the run automatically define the variable sizes as the field sizes from which the data is read. See RDC, RDL, RLN, and SUB in Section 7.

For example, this statement lets the run define the size of <data>:

```
@rdl,0,b,2,6 'cust' <data>h .
```

Converting to Field Names

Use the Load Field Name (LFN) run statement to convert an existing run to one that uses named fields (see LFN in Section 7). You can also use it to translate column position data, such as the values captured using the Output Mask (OUM) run statement, into field names (see OUM in Section 7).

The LFN statement loads variables with the names of report fields that correspond to the column-character positions supplied in the statement.

Selecting Fields to Display

Use the Format (FMT) statement to select which fields you want to display in a following DSP, OUT, or OUM statement (see FMT in Section 7).

Efficiency Considerations

This section on efficiency considerations applies only to the OS 1100 MAPPER System. It applies to other MAPPER systems only if the report is cached.

When a run encounters a field name, it reads the whole report heading, which requires one additional I/O access; however, since it already has read the heading, it does not need to read it again for other field names in the same run statement.

It also does not need to read it again for succeeding run statements that specify the same report and fields, or a result derived from it.

In this example, the run reads the headings for the Search (SRH) statement, but does not read them for the Sort (SOR) or Totalize (TOT) statements:

```
@srh,0,d,1 '' 'st cd' □,or .
@sor,-0 '' 'order number' □,1 .
@tot,-0 s 'order number','ord qty' □,s,+ .
```

Since many statements reprocess the result of a previous statement, the overhead of reading report headings is distributed and the impact minimized on any individual statement.

Section 4
Variables, Reserved Words, and Constants

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

Variables and reserved words are important aspects of run design, so you should know how to use them properly and efficiently. This section defines variables and reserved words and describes how to use them in run design.

This section includes these topics:

- Some terms you should know
- Selecting a variable name
- Understanding variable types and sizes
- Selecting a variable type and size
- Defining and initializing variables
- Using variables in run statements
- Changing the contents of variables
- Testing the contents of variables
- Using a variable table
- Working with variable limits
- Capturing input
- Examples using variables
- Using reserved words
- Using predefined constants

Some Terms You Should Know

This subsection describes some of the terms you should know to understand how to use variables in MAPPER runs.

What Is a Variable?

A variable is a named storage area. In a run, you can store data in variables and retrieve data from variables. You can also use variables in run statements to supply values for fields and subfields.

Before you can use a variable in a run, you must define it. You define a variable by declaring its name, type, and size. You can also assign an initial value to the variable.

What Is an Array?

An array is a series of one or more variables of the same type and size. Although an array can contain many variables, you assign only one name to the entire array. Each variable within the array is called a member of the array. For information on arrays, see LDA in Section 7.

What Is a Reserved Word?

A reserved word is a system-maintained variable. In your run control report, you can use reserved words to retrieve information maintained by the system. For example, you can use the reserved word DATE1$ to retrieve the current date; you can use the reserved word USER$ to retrieve the user-id of the current user. For a list of all reserved words, see Appendix B.

What Is a Constant?

A constant is a named value. You define constants by assigning a logical name to a value. You can then use the name rather than the value in run statements. Using constants makes a run easier to read, develop, debug, and maintain. For information on defining and using constants, see "Using Predefined Constants" in this section.

Selecting a Variable Name

You can name a variable using either a character string or a number. Variables that have been named with a character string are called named variables; variables that have been named with a number are called numbered variables.

Named Variables

To name a variable with a character string, use this format:

<*name*>

where ***name*** is a 1- to 12-character string, and the less than (<) and greater than (>) signs are required. You can use any characters to name the variable, including spaces and special characters. Here are some examples of named variables:

<counter1> **<commission>** **<2001>**

1100: On OS 1100 MAPPER Systems, use alphanumeric characters (A-Z and 0-9) for the name.

Numbered Variables

To name a variable with a number, use this format:

V*n*

where ***n*** is a number from 1 to 199 (or 1 to 399 if your system is set up to handle 399 variables). Ask your coordinator for the maximum number of variables allowed at your site.

You must precede ***n*** with the letter V. Here are some examples of numbered variables:

v9 **v23** **v7**

Named Vs Numbered Variables

You can use named and numbered variables in the same run without conflict. However, using only named variables makes a run easier to read, develop, debug, and maintain.

Using Named and Numbered Variables on OS 1100 MAPPER Systems

This section on using named and numbered variables applies only to the OS 1100 MAPPER System.

You can use named variables anywhere you would use numbered variables in a run. When you define a named variable, the system assigns it the lowest unused variable number; therefore, the system assigns the first variable in a run to V1, the second to V2, and so on. When the system assigns a name to a variable, you can no longer refer to it by its number.

If you intend to mix named and numbered variables in the same run, use the Use Variable Name (USE) run statement to assign a variable name to a specific variable number.

Note: *Named variables are slightly less efficient than numbered variables, so you may want to use numbered variables in logic-intensive runs. In most runs, however, their cost is insignificant.*

Assigning Names to Numbered Variables

If you want to use a name for a variable that has been assigned with a number, you can use the USE run statement to equate a name with a numbered variable. You can then refer to the variable using either the name or the number. See USE in Section 7.

Converting Named Variables to Numbered Variables

If you want to convert all variables in a run from named to numbered variables, you can do so using the CVT (Convert Variable Table) run. See "Using a Variable Table" in this section.

Understanding Variable Types and Sizes

A variable type indicates the kind of data stored in the variable. The MAPPER system provides several data types:

- I Integer (whole numbers)
- F Fixed-point (numbers with a fractional part)
- A Alphanumeric (any characters)
- H Hollerith (any characters)
- S String (any characters)
- **1100:** O Octal (octal numbers)

The type determines not only the kind of data allowed in the variable, but also the maximum size for the variable and, except for OS 1100 Systems, the manner in which the system loads data into the variable.

Table 4-1 lists each variable type, along with its size limit and default initial value. See also ART, CHG, and IF in Section 7 for more information on how those statements handle the different types of variables.

Table 4-1. Variable Types and Sizes

Variable Type	Description
A (Alphanumeric)	Characters = all characters Maximum size = 16 Default initial value = blanks
F (Fixed-Point)	Characters = all numbers, signs (+ and -), and decimal (.) Maximum size = 18 (including sign and decimal) Default initial value = zero The maximum size of the fractional portion is 16 characters. When characters are added to the left of the decimal, causing the contents to exceed 18 characters, the fractional portion is truncated. For systems other than OS 1100 MAPPER Systems, note that precision of F type numbers is machine-dependent. See the C compiler documentation for your computer.
H (Hollerith)	Characters = all characters Maximum size = 18 Default initial value = blanks
I (Integer)	Characters = numbers, signs (+ and -) Maximum size = 16 (including sign) Default initial value = zero
1100: O (Octal)	Characters = octal numbers (0 through 7) Maximum size = 12 Default initial value = zero
S (String)	Characters = all characters Maximum size = 256 Default initial value = blanks

Default Justification of Variables

This section on default justification does not apply to the OS 1100 MAPPER System.

Table 4-2 lists each variable type along with its default justification.

Table 4-2. Default Justification

Variable Type	Justification
A (Alphanumeric)	If alphanumeric, left-justified; if all numeric, right-justified
F (Fixed-Point)	Right
H (Hollerith)	Left
I (Integer)	Right
S (String)	Left

No Type Verification

When you load data into a variable, the MAPPER system does not verify that the data you load matches the type of the variable. For example, you can load alphabetic characters into an I (integer) type variable, and the system will not issue a system message.

You are responsible for verifying that the contents of the variable match its type. See "Testing the Contents of Variables" in this section.

Selecting a Variable Type and Size

To select a type for a variable, determine whether you need to perform arithmetic operations on the data that the variable stores.

If you need to perform arithmetic operations on the data, define the variable as an I or F type variable. Use I for whole numbers (positive and negative); use F for fixed-point decimal (positive and negative).

If you do not need to perform arithmetic operations on the data, you can define the variable as an A, H or S type:

- The MAPPER system checks the contents of A type variables to determine how to process the data. For this reason, A type variables are the least efficient way to store numeric and nonnumeric data.
- H type variables are the most efficient way to store nonnumeric data up to 18 characters in length.
- S type variables are the only way to store data greater than 18 characters in length.

Determine the Size

Before defining the variable size, determine the maximum number of characters you need to store the data. Then, define the variable size to accommodate that number of characters. It is important to accurately estimate the size of the variable needed to store the data.

Defining and Initializing Variables

To define a variable, you declare the name, type, and size of the variable on any run statement in which a variable is allowed.

Regardless of the run statement you use to define the variable, you must use this format to specify the name, type, and size:

name type size [. *fsize*]

where *name* is the variable name, *type* is the variable type, and *size* is the character length of the variable.

For F type variables, *size* is the total size of the variable including the decimal point. Use *fsize* to specify the number of characters to the right of the decimal. For example, <dollars>f6.2 is an F type variable that is six characters long. Two of the six characters are to the right of the decimal.

Methods for Defining a Variable

There are three ways to define a variable in a MAPPER run:

- Using the Load Variable (LDV) run statement
- Using the Change Variable (CHG) run statement
- Using a run statement that provides a field for a variable to receive data

Using the LDV Statement

The LDV run statement is an efficient way to define a variable and set an initial value for the variable. For example:

```
@ldv <dollars>f6.2=109.99 .
@ldv <prodtype>h9=blackbox1 .
@ldv <counter>i2=1 .
```

See LDV in Section 7 for more information and examples.

Using the CHG Statement

With the CHG run statement, you can perform simple calculations on variables and capture user input. For example:

```
@chg <calc1>i10 <calc> + 50 .
@chg <commission>i18 <sales> * <percent> .
@chg input$ <name>s30,<address>s30 .
```

See CHG in Section 7 for more information and examples.

Using Other Run Statements

You can define a variable in any run statement that provides a field for specifying a variable to receive data.

For example, the following Find (FND) statement defines a variable containing the line number of the find:

```
@fnd,0,2,b '' 'st-cd' |,ip ,<line>i6 .
```

Redefining Variables

Once you have defined a variable, you do not need to include the type and size each time you use the variable unless you want to redefine it.

To redefine a variable, specify a new type and size for the variable on any run statement that allows a variable. When you redefine a variable, the original definition and contents are discarded.

Using Variables in Run Statements

Once you have defined a variable, you can use the variable to supply a value for a run statement field or subfield. To use a variable in a run statement, specify the name of the variable. This example uses <cab>, <drawer>, and <rep> to specify the report to process:

```
@find,<cab>,<drawer>,<rep> '' 'st-cd' |,ip ,<line> .
```

Referring to Substrings

You can refer to part of the contents of a variable, called a substring. To refer to a substring of a variable, use this format:

name(*position-characters*)

where *name* is the name of the variable, *position* is the starting position of the substring, and *characters* is the number of subsequent characters to include in the substring.

For example, to refer to the first three characters of <phone>, specify the following:

```
<phone>(1-3)
```

Trailing Characters

To refer to a substring that includes a specific number of trailing characters, use this format:

name(0-*n*)

where *name* is the name of the variable and *n* is the number of characters at the end of the variable that you want to include in the substring. For example, to refer to the last four characters of <phone>, specify the following:

```
<phone>(0-4)
```

Unknown Number of Trailing Characters

To refer to a substring that starts at a specific position and includes all the remaining characters in the variable, use this format:

<*name*>(*p*-0)

where ***name*** is the name of the variable and ***p*** is the position of the first character. For example, to refer to the substring of v50 that starts with the 6th character and includes all the remaining characters, specify the following:

v50(6-0)

Using Contents to Act as Name or Number

To use the contents of one variable to act as the name (or number) of another variable, use one of the following formats:

<<*name*>>

VV*n*

For named variables, ***name*** is the name of the variable whose contents you wish to use for the actual variable reference; for numbered variables, ***n*** is the number of the variable whose contents you wish to use for the actual variable reference.

Example

The following statement uses the contents of one variable as the name of an actual variable reference:

```
@ldv <one>h3=two . load <one> with the characters two
@ldv <<one>>i2=2 . load variable <two> with 2
```

Using Scientific Notation

You can load A, F, and I type variables with a number expressed in scientific notation. That is, the number is expressed as a fractional part called the mantissa and a power of ten called the characteristic. Here is the format for expressing a number in scientific notation:

*mantissa*E*characteristic*

where *mantissa* is the fraction part of the number and *characteristic* is the power of ten. Here are some examples:

12e5 = 1,200,000

.1234e-3 = .0001234

When a number being loaded exceeds the size of a variable, the MAPPER system automatically translates the number into scientific notation.

If the number still does not fit into the variable, the system loads the variable with asterisks (*).

Changing the Contents of Variables

The MAPPER system provides many run statements for setting and changing the contents of a variable. This section presents the run statements most commonly used to change the contents of variables.

The examples that follow show variables being changed instead of being initially defined; therefore, the type and size are not included following the variable name.

Using the LDV Statement

The LDV run statement is the most efficient method to use for setting and changing the contents of a variable to a specific value. Here is an example:

```
@ldv <report>=54,<error>='unknown report' .
```

See LDV in Section 7 for more information.

Using the INC and DEC Statements

The Increment Variable (INC) and Decrement Variable (DEC) run statements are the most efficient methods to use for incrementing and decrementing the value of a numeric (type I or F) variable. Here are some examples:

```
@inc <counter> .
@dec <counter> .
```

See INC and DEC in the online help system (HELP,@INC and HELP,@DEC).

Using the ART Statement

The Arithmetic (ART) run statement is the most efficient method to use for setting the value of a numeric (I or F type) variable to the result of a complex arithmetic expression. Here is an example:

```
@art (2/3)*<cost> <expense> .
```

See ART in Section 7.

Using the CHG Statement

Use the CHG run statement to set the value of a numeric (I or F type) variable to the result of a simple arithmetic expression or to capture input data. Here is an example of using the CHG run statement to set the value of a variable with a simple arithmetic operation:

```
@chg <total> <subtot1> + <subtot2> .
```

See CHG in Section 7.

Testing the Contents of Variables

You can use the Define (DEF), If Conditional (IF), and Locate and Change Variable (LCV) run statements to test the contents of a variable.

You can also use the VARIABLE run to see how the type of variable and the method used for loading or modifying the variable affect its contents.

Using the DEF Statement

With the DEF run statement, you can determine the defined name, type, and size of a variable. You can also determine information about the contents of the variable, such as:

- Number of significant characters
- Type of data stored in the variable
- Size of the data if it were packed

See DEF in Section 7.

Using the IF Statement

With the IF run statement, you can use relational operators to compare the value of a variable to one or more values. See IF in Section 7.

Using the LCV Statement

With the LCV run statement, you can find and optionally replace strings in a variable. You can also specify a label at which to continue execution when a find is made. Using the LCV statement to compare two values and then direct the flow of execution based on the result of the compare is more efficient than using an IF/GTO sequence.

See LCV in Section 7.

Using the VARIABLE Run to Check Contents

The VARIABLE run demonstrates how a variable type affects its contents. To use the run, type **variable** on the control line and transmit.

The system prompts you to enter a value. The system then loads that value into a series of variables of different types and displays the contents of those variables.

This example shows the VARIABLE run screen and a value to test:

```
Enter a value ♦·100.00


            The VARIABLE run displays the contents of variables
              after initialization and arithmetic operations
```

After you transmit, a screen similar to the following appears:

```
Line♦·                Roll♦·                                  RESULT (-1)


Variable values for entry (100.00)

 @CHG INPUT$ V1      @CHG INVAR$ V2      CHG V3 V1 + 1        @INC V1
=================  =================  =================  =================
V1A6   = (100.00)  V2A6   = (100.00)  V3A6   = (   101)  V1A6   = (   N/A)
V1I6   = (   100)  V2I6   = (   100)  V3I6   = (   101)  V1I6   = (   101)
V1F6.3 = (100.00)  V2F6.3 = (100.00)  V3F6.3 = (101.00)  V1F6.3 = (101.00)
V1H6   = (100.00)  V2H6   = (100.00)  V3H6   = (100.01)  V1H6   = (   101)

                          Press F1 to enter another value
```

If you want to test another value, press **Resume**.

Using a Variable Table

A variable table lists all the variables used in a run. For each variable, the table marks the statement at which the variable was defined, and all subsequent references to the variable.

Procedure

To build a variable table, follow these steps:

1. Display the run control report for which you want a table of variables.
2. Execute the BVT run.
3. When the result is displayed, either press **Resume** to replace the original run control report with the result on display, or duplicate the result into another run control report.

Using the BVT Run to Build Variable Tables

Use the BVT (Build Variable Table) run to build or rebuild a table that displays the location of all the variables in your run control report. You can also use it to name variables so you can easily convert to or from named variables. The variable table is displayed at the end of your run control report.

To use the BVT run, display your run control report and enter one of these requests:

BVT	Builds a variable table at the end of your run control report and displays it as a result.
BVT,Q	Lists only where the variables are defined (for example, v1a3 rather than v1).
CVT	Converts v-type variables (such as v1) to named variables (such as <name>) using a previously built variable table from the end of your run control report.
CVT,N	Converts all named variables to v-type variables.
CVZ	Converts all v-type variables to three-character variables (for example, v1 to v001).

Note that each of the previous requests also calls the Build Label Table (BLT) function, which builds or rebuilds the label table.

When you rebuild a variable table, the new table is matched with the existing table to preserve any user-defined names or comments.

Building or rebuilding a variable table produces the information shown in Figure 4-1 as a result at the end of your run control report.

```
.VARIABLE TABLE
*    Name    .Vnum.Sq.            Line Numbers            .     Comment
*============.====.==.====================================.===============
```

Figure 4-1. Fields in Variable Tables

Following is a description of the fields in the variable table:

Field	Description
Name	Variable name (<name>) to equate to the v-type number. Default = N000.
Vnum	V-type number to equate to the variable name (<name>).
Sq	Sequence number used to match and save user comments.
Line Numbers	Line number where the variables are located. The line numbers containing defined variables are flagged with asterisks.
Comment	Any user-supplied comments.

Follow these guidelines when using the BVT run:

- When you use the BVT or BVT,Q request to build or rebuild a variable table locating a named variable, the run first checks for an existing table to determine a v-type variable with which to associate the named variable. The run does not read USE run statements to determine a v-type variable (see USE in Section 7). For example, **@use name=v199** does not necessarily associate <name> with v199.

- When you use the CVT or CVT,N request to convert variables, you are not notified when a variable-variable is converted. For example, if you convert vv199 to <<name>>, it may not execute correctly.
- When you use the CVT, CVT,N, or CVZ request, and the size of the new variable causes the run statement to extend beyond the end of the line, the original report is displayed at that line number. No changes are made to the run control report until you change that run statement line.

Working with Variable Limits

This section describes some of the variable limits you need to be aware of when you write your runs.

Note: *If you plan to port runs to other MAPPER systems, for example, porting A Series runs to an OS 1100 MAPPER System, make sure the runs comply with the limitations on variable usage for that system.*

Maximum Number of Variables Allowed

The MAPPER system enforces a system limit on the total number of variables that can be used concurrently in any run. The system limit is between 199 and 399 variables, depending on how your system is set up. Ask your coordinator for the maximum number of variables allowed at your site.

Run Registration

This section on run registration does not apply to the OS 1100 MAPPER System.

During run registration, the MAPPER coordinator specifies the maximum number of variables used in the run. The number can be less than or equal to the system limit on variables.

Consult with your coordinator to determine an appropriate limit on the total number of variables for your run. Note that runs registered with fewer variables execute more efficiently than those registered with a great number of variables.

Maximum Number of Characters

This section on maximum number of characters does not apply to the OS 1100 MAPPER System.

In addition to the limit on the number of variables per run, the coordinator also enforces a limit on the total number of characters you can use concurrently for all variables. Usually, this limit is set to 6,616 characters. However, the coordinator can increase the limit to a maximum of 20,000 characters of variable storage space per run.

String Space Limits

This section on string space limits applies only to the OS 1100 MAPPER System.

In addition to the limit on the number of variables per run, the system enforces a limit of 4096 characters for S type (string) variables. Array variables also use string space, regardless of their type.

Techniques for Handling Variable Limits

This section describes some methods of conserving both the number of variables used in a run and the amount of variable storage space used.

Writing Modular Subroutines

This section on writing modular subroutines applies only to the OS 1100 MAPPER System.

Organize your run into subroutines in such a way that no subroutine uses more than the maximum limit of variables. Then, use the Call Subroutine (CALL) and Return Call Routine (RETURN) run statements to execute the subroutines.

For variables used by more than one subroutine, you can pass and return up to 40 variables to a called subroutine using CALL and RETURN statements.

Passing Variable Arrays

Although an array can contain from 1 to 199 (or 399) variables, an entire array counts as only one variable in the total number of variables you can pass to a subroutine on the CALL statement. See LDA in Section 7.

Clearing Numbered Variables

If you are having problems with storage limits, you can clear the variables once you have finished using them. Use the Clear Variable (CLV) run statement.

Using the Variable Stack

The MAPPER system initializes and maintains a variable stack for each executing run. The variable stack is a storage area that consists of from 0 to 10 levels. You can save and retrieve a set of variables in each stack level.

By saving, clearing, and then retrieving sets of variables, you can define and maintain up to ten times the maximum number of variables allowed per run.

These run statements manipulate the variable stack:

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

Example

Following is an example of a run statement sequence you can use to save and retrieve a set of variables:

1. Use a PSH statement to save a set of variables to the stack.
2. Use a CLV statement to clear the definitions and contents of the variables in the run.
3. Execute a subroutine.
4. Use a POP statement to retrieve the set of variables from the stack.

Reserved Word

The reserved word STACK$ contains a number from 0 to 10 to indicate the most recent stack level to which the run has saved a set of variables.

Capturing Input

This section describes how to capture screen input and initial input parameters.

Screen Input

Screen input is any input a user enters on the screen during run execution. The MAPPER system provides several methods for capturing screen input. To capture screen input, you use run statements that prompt the user for input and statements that accept input into the run.

Initial Input Parameters

Initial input parameters are a list of from one to forty parameters that you can pass to a run at the time the run starts executing, either from a statement that starts a run or from the run user's manual run call. The run to which the parameters are passed uses a CHG INPUT$ statement to accept the input.

Prompting the User for Input

You can use an Out Variable (OUV) run statement followed by an Input Variable (ITV) statement to display variables, literals, or the contents of reserved words on the screen.

In the display, you can include tab characters to delimit fields on the screen in which you want the user to enter data. The ITV statement suspends the run until the user presses a function key or transmits.

To accept the input into your run, use a CHG INPUT$ statement after the OUV statement. See OUV and ITV in Section 7.

You can also use Screen Control (SC) commands to build menus and screens for users to enter data. Here are some of the features of the SC statement:

- You can design more attractive displays by using colors and boxes.
- You can build menus to overlay other screen displays and allow the user to erase the display.

- You can define up to 10 function keys and a function key bar to display the key options.
- You can provide context-sensitive help for input fields.

See SC in Section 7 for detailed information and examples.

Using the Output Area

You can build a display in the output area, move the output area to the -0 result using the Break (BRK) run statement, and then display the result using the Output (OUT) run statement.

See BRK and OUT in Section 7 and "The Output Area" in Section 6 for more information.

Accepting User Input

To accept the input a user enters on a screen into a run, use the CHG run statement with one of the following reserved words:

INPUT$
INSTR$
INVAR$
INVR1$
INMSV$ with the OUM run statement
ICVAR$ with the CHD run statement
FKEY$ with the ITV, KEY or CHD run statements

Using INPUT$

When using INPUT$ to capture data, remember these things:

- Use a CHG INPUT$ statement to load from one to forty variables with the current input. The current input can be initial input parameters or input entered by the user on the screen.
- Always use a CHG INPUT$ statement after prompting the user for input.
- The data loaded into the variables starts at the first tab character on the screen display: the data after the first tab character is loaded into the first variable, the data after the second tab character is loaded into the second variable, and so on.

For information on capturing initial input parameters, see "Capturing Initial Input Parameters" in this section.

Example

This example uses the reserved word INPUT$ to enter data from the screen. A numbered explanation follows it.

```
1. @brk,0,a .
2. Enter appropriate data and press Transmit .
3. □       ,Enter start date in format yymmdd
4. □       ,Enter end date in format yymmdd
5. Place cursor here ->□ , and press Transmit .
6. @brk out,-0,2,6,1,1,y .
7. @chg input$ <stdate>i6,<endate>i6 .
```

1. The first Break (BRK) statement defines the next output area (that is, the lines that follow) as cabinet 0, drawer A. (See "Handling Reports and Results" in Section 6 for an explanation of the output area.)

2-5. Place these lines in the output area.

6. The second BRK statement places the preceding lines from the output area into the -0 result. The OUT statement displays the new -0 result on the screen.

7. CHG INPUT$ loads <stdate> with the start date and <endate> with the end date that is entered.

Using INSTR$

When using INSTR$ to capture data from the screen, remember these things:

- Use a CHG INSTR$ statement to load one or more variables with lines of input from the screen.
- Always use a CHG INSTR$ statement before prompting the user for input.
- The data loaded into the variables starts at the first line on the screen or at the last SOE character before the cursor. The input for each variable ends at the end of each line. The data in the first line is loaded into the first variable, the data in the second line is loaded into the second variable, and so on.
- The maximum number of variables you can load is equal to the number of lines on the screen.
- Do not use INSTR$ with formatted or protected screens.

Example

This example loads variables with two lines of data on the screen:

```
@chg instr$ <linea>s80,<lineb>s80 .
```

<linea> contains eighty characters, starting with the first character following the last SOE character. If there is no SOE character on the screen, it contains the first eighty characters starting at the home position. <lineb> contains up to eighty characters starting with the first character on the next line.

Using INVAR$

When using INVAR$ to capture data from the screen, remember these things:

- Use a CHG INVAR$ statement to load from one to forty variables with input delimited by tabs.
- Always use a CHG INVAR$ statement before prompting the user for input.
- The data loaded into the variables starts at the first tab character: the data after the first tab character is loaded into the first variable, the data after the second tab character is loaded into the second variable, and so on.
- The length of the input field is determined either by a tab or by the length of the receiving variable on the CHG statement.

Example

This example uses the reserved word INVAR$ to enter data from the screen:

```
@chg invar$ <inp1>s17,<inp2>i3,<inp3>i6 .
@brk .
Enter description □                    ,
Enter quantity □    ,
Enter date in format yymmdd □        ,
@brk out,-0,2,3,1,1,y .
```

In this example, after the user enters the solicited information, the run continues at the statement that follows the OUT statement. <inp1>, <inp2>, and <inp3> contain the information the user entered.

Using INVR1$

INVR1$ is similar to INVAR$, but it allows you to load several fields into one variable. When using INVR1$ to capture data from the screen, remember these things:

- Use a CHG INVR1$ statement to load from one to forty variables with input.
- Always use a CHG INVR1$ statement before prompting the user for input.
- The length of the variable determines how many characters, including tab characters, are loaded from the screen. Characters beyond the variable length are discarded.
- Do not use INVR1$ with formatted or protected screens.

Example

In this example, after the user enters the solicited information, the run continues at the statement that follows the OUT statement. <name> contains both the first and last name the user entered. Note that protected format is not used.

```
@chg invr1$ <name>s25 .

            First          Last

Enter name □              □
@brk out,-0,2,4,1,1,y .
```

Using INMSV$ with the OUM Statement

When using INMSV$ to capture data, remember these things:

- Use a CHG INMSV$ run statement to capture input entered on a function mask display.
- Always use a CHG INMSV$ run statement before prompting the user to enter function mask information with the OUM statement.

For more information, see OUM in Section 7.

Using ICVAR$ with the CHD Statement

When using ICVAR$ to capture data, remember these things:

- Use a CHG ICVAR$ run statement to capture input entered on the control line during a display.
- Before prompting the user for input, use a Command Handler (CHD) run statement and then the CHG ICVAR$ run statement.
- The ICVAR$ reserved word captures information only when the user transmits from the control line.

Example

This example displays a report and loads <controlline> with input the user entered from the control line:

```
@chd 100 .
@chg icvar$ <controlline>s80 .
@dsp,0,b,2 .
```

Using FKEY$ with the KEY and CHD Statements

When using FKEY$ to capture function key input, remember these things:

- Use the Function Key Input (KEY) run statement to capture function key input; use the CHD and KEY statements to capture function key input from the control line. The system puts the input in the reserved word FKEY$.
- Before prompting the user for input, use a KEY run statement; if you want to accept function key input when the cursor is located at the control line, also use a CHD run statement.

Example

This example requests function key input and displays the current -0 result:

```
@key .
@out,-0,4,1 .
```

Following these statements, the run can test FKEY$ for its contents and proceed accordingly.

Capturing Initial Input Parameters

A run can accept up to 40 initial input parameters. To define the input you want the run to accept, use a CHG INPUT$ statement that specifies a list of variables to capture the input.

Usually, the CHG statement for capturing initial input is the first statement in the run. However, the only requirement for placement of the CHG statement is that it must be executed before any other input is accepted in the run.

Example

The following statement is the first statement in a run control report. It captures initial input parameters in <report> and <option>:

```
@chg input$ <report>s20,<option>s10 .
```

Passing Input to the Run

Initial input parameters are passed to a run on the call that starts the run.

To pass initial input parameters to a run, use this format:

name[,*input1* ,*input2* . . . ,*input40*]

where *name* is the name of the run and *input1* through *input40* are from one to forty parameters to pass to the run.

The first input specified is loaded into the first variable listed on the CHG INPUT$ statement; the second input specified is loaded into the second variable listed on the CHG INPUT$, and so on.

Examples

Here is an example of how a user passes the initial input parameters myreport and sort to the UPDATE run:

```
update,myreport,sort
```

You can also pass the same information to the run on any run statement that starts a run. For example:

```
@run update,myreport,sort .

@lnk update,myreport,sort .
```

Examples Using Variables

This section uses one Display Report (DSP) statement to help explain the reasons for using variables in run statements. It also demonstrates several ways to initialize variables depending on the situation. The following DSP statement serves as the basis for all of the following hypothetical situations:

```
@dsp,v1,v2,v3 .
```

This DSP statement requests that the cabinet number in v1, the drawer in v2, and the report number in v3 be displayed.

Before using this DSP statement, you could have initialized the variables v1, v2, and v3 in any number of ways. The possibilities include the following:

- You could load the variables earlier in the run so that the information, if moved to another cabinet, needs to be changed only once in your run control report, at the place where you initialized v1, v2, and v3. This is especially useful if the report is processed repeatedly in the run.
- You could write the run to capture the information entered on the screen by the run user, for example:

```
@brk .
Enter the location of the report you
want to display and press Transmit

Cabinet□   , Alphabetic drawer□ , Report□   ,
@brk out,-0,2,6,1,1,y,,,i .
@chg input$ v1i3,v2a1,v3i4 dsp,v1,v2,v3 .
```

V1 is initialized as the cabinet entered, v2 as the alphabetic drawer entered, and v3 as the report number entered on the screen. Use v1, v2, and v3 to display the report the user requested.

- You could load v1, v2, and v3 as a result of another statement that automatically loads variables with pertinent information, for example:

```
@ldv v1i3=0,v2a1=d fnd,v1,v2 '' 22-3 □,'  1' v3i5 .
@dsp,v1,v2,v3 .
```

  First you loaded v1 and v2 with the cabinet and drawer in which to find some data. The FND statement produces v3, which contains the report number of the first find. Use v1, v2, and v3 to display this report.

- You could load the variables with a combination of screen information and internal run loading (LDV), as in this example:

```
@brk .
Enter the code to find and where to look
(Cabinet 0, drawer b report)

   Status Code□   , Report number□   ,
@brk out,-0,2,4,1,1,y,,,i .
@chg input$ v4a2,v3i4 ldv v1i1=0,v2a1=b .
@fnd,v1,v2,v3 '' 2-2 □,v4  ,v5i5 .
@dsp,v1,v2,v3,v5 .
```

  First you loaded v1 and v2 with the cabinet and drawer in which to find some data, as in the last example, but you let the run user request the data to find (captured in v4) and the report in which to scan for the data (captured in v3). The FND statement produces v5, which contains the line number of the first find. Use v1, v2, v3, and v5 to display the report requested at the line where the find was made.

- You can use variables as counters that the run increases whenever logically necessary, then later checks, as in this sequence of statements:

```
@    ldv v1i3=0,v2a1=d,v3i4=1,v4i5=6 .
@    lzr,v1,v2,v3 v5i5 . v5=nr.lines
@    fnd,v1,v2,v3,v4,196 '' 22-3 □,'   1' ,v4 .
@    dsp,v1,v2,v3,v4 . v4=line
@    inc v4 if v4 not > v5 gto lin -2 .
@196: .
No more finds.
@    gto end .
```

V4 is the counter for the line on which to start the scan, as well as the line number of the find. V5 is the number of lines in the report. As v4 increases, the find process begins further into the report.

Use v1, v2, v3, and v4 to display the report at each line where a find is made. The run user resumes the run to continue displaying the found lines until v4 becomes greater than v5, indicating the entire report has been processed.

You can use loops and counters like these in many other ways. In the same example, for instance, you could execute a find across reports and display the correct report at the appropriate line by using more variables and checks.

Using Reserved Words

A reserved word is a character string that is reserved by MAPPER software for specific use in a MAPPER run.

You can initialize a variable with the value of a reserved word using CHG and LDV statements. For example, this CHG statement initializes v2 as one less than the current vertical position of the cursor:

```
@chg v2i2 curv$ - 1 .
```

This LDV statement initializes <cabinet> to contain the cabinet number, <drawer> to contain the drawer number, and <report> to contain the report number of the last report or result processed or on display:

```
@ldv,w <cabinet>i4=cab$,<drawer>i6=drw$,<report>i4=rpt$ .
```

You can also use reserved words to specify values for fields and subfields in run statements.

For more information, see CHG and LDV in Section 7; for a list of reserved words, see Appendix B.

Reserved Words in the Output Area

The system inserts the value of any reserved words placed in the output area, not the reserved word itself.

1100: To place the value of reserved words in the output area on OS 1100 MAPPER Systems, load variables with the reserved words and place the variables in the output area.

Using Predefined Constants

You can predefine values for any of the fields and subfields in the run statements you intend to use, including the location of the database (cabinet, drawer, report), fields to process (column-character positions), the actual parameters that indicate how you want the data processed, and more.

The Define Constant (DEFINE) and Include Report (INCLUDE) statements enable you to define constants to specific values in one place, making it easier to modify such items as database location, fields to process, and function parameters during the design and debug phase of run design. To make changes to items such as these, simply modify the define constants themselves. You no longer need to go through one or more run control reports looking for every occurrence of the item to change.

The DEFINE and INCLUDE statements are especially useful when working with complex applications, or when several run designers are involved in designing an application.

Defining Constants

Use a DEFINE statement to define values for any of the fields and subfields in a run statement, including items such as the following:

- Labels (for example, @010:)
- Run function calls (for example, SRH)
- Location of the database (*c*, *d*, *r*)
- Fields to process (*c c*)
- Function parameters (*p*)
- Output area data
- Variable names

Note: *Before placing your run into production, use the BLT function to convert all DEFINE statements to their defined values.*

Format

:DEFINE *constant value*

Following is a description of the call and the fields:

Field	Description
:DEFINE	Define Constant call. You may use either :D or :DEFINE.
constant	Up to 18 characters, beginning with an alphabetic character, for the constant you intend to use instead of the actual value. Valid characters include alphabetic characters, the numbers 0 to 9, and the underscore character (_). Constant names are case sensitive.
value	Up to 18 characters including any combination of variable names, reserved words, or literal data to use in run statements by using the associated constant.

Follow these guidelines when using the DEFINE statement:

- Place all DEFINE statements at the beginning of your run control report.
- You can define only one constant in each DEFINE statement. Constant names are case sensitive.
- You can define up to 400 constants in a run. If the system encounters duplicate constants, it uses the value of the constant from the last DEFINE statement.
- When defining a literal value that contains spaces, enclose the literal value in apostrophes (') or quotation marks ("). To define a literal value that contains an apostrophe, use two apostrophes (' ') and enclose the entire value in quotation marks. See the examples in this subsection.
- When the system detects a quotation mark in the defined value, it includes everything up to the next quotation mark, unless the value exceeds the maximum size allowed, or the system encounters the end of the line.

- When several runs use common variables or data to initialize constants to the correct value, define the common constants in a separate report and use the INCLUDE statement within each run control report to include the constants. See "Including Defined Constants" in this section.
- You can define a constant to contain a variable name. To initialize the variable (using the define constant), enclose the variable type and size in apostrophes. For example:

```
:define drawer v102
@ldv drawer'a1'=b .
```

- When defining related groups of data, define the main item first with a standard DEFINE statement (:DEFINE). Define subordinate items in lines following the main item using a colon followed by a space. See the online help system (HELP,DEFINE) for examples.
- Do not place other run statements on the same line after a DEFINE statement. Other run statements following it on the same line are ignored. Put the next run statement on a new line.

Notes: *1. Using DEFINE statements requires an additional 16,384 (16K) bytes of memory to execute your run. Before placing your run into production, use the BLT function to convert all defined constants to their defined values.*

2. The BLT function converts the included and defined constants to their actual values, removes comments, and creates a result. Save this result in a report other than your source run control report. Use the new report as the production version. Your source run retains the defined constants and comments for easier maintenance.

3. When converting the defined constants, if the BLT function encounters lines that would expand past the end of the line, it automatically inserts continuation characters (\ ) and wraps the statement appropriately. Because of this feature, avoid using @gto lin+n or @gto lin-n statements in runs containing defined constants.

Example 1: Define Report, Field Columns, and Search Parameters

Define search_rpt as the cabinet, drawer, report, and line number to use on the SRH statement, prod_type as the columns to process, and item as the search parameter:

```
:define search_rpt 0,b,2,5
:define prod_type  15-9
:define item       greenbox8
@srh,search_rpt d prod_type  ,item <lines>i4,<scan>i4 .
```

After you debug the run and execute the BLT function, the production run contains the following run statement:

```
@srh,0,b,2,5 d 15-9  ,greenbox8 <lines>i4,<scan>i4 .
```

Example 2: Define Report, Field Columns, and Find Parameters

Define inventory as the report to process, order_num and cust_code as the field columns, and target as the find parameter:

```
:define inventory  0,d,1
:define order_num  5-6
:define cust_code  26-4
:define target     99951s
@fnd,inventory '' order_num  ,target ,<line>i3 .
@rdl,inventory,<line> cust_code <customer>h .
```

After you fully debug the run and execute the BLT function to convert the defined constants to their actual values in the run control report, the production run contains the following run statements:

```
@fnd,0,d,1 '' 5-6  ,99951s ,<line>i3 .
@rdl,0,d,1,<line> 26-4 <customer>h .
```

Example 3: Define Field Columns, Parameter, and Variable

Define a field to process (field), a search parameter that contains leading spaces (target), and a variable in which to store the number of lines found (snum):

```
:define field  56-8
:define target "'△△△△△100'"
:define snum   v100
@srh,0,c,1 d field □,target snum'i5' .
```

After you execute the BLT function to convert the defined constants to actual values, the run contains the following statement:

```
@srh,0,c,1 d 56-8 □,'△△△△△100' v100i5 .
```

Including Defined Constants

Use INCLUDE statements to add all defined constants from another report to your current run.

Note: *Before placing your run into production, use the BLT function to convert all constants defined in the included report to their actual values.*

Format

```
:INCLUDE,c,d[,r].
```

Following is a description of the call and the subfields:

Field	Description
:INCLUDE	Include Report call. You may use either :I or :INCLUDE.
c,d,r	Report to include. If you do not specify a report, the system uses report 0 in that drawer.

Follow these guidelines when using the INCLUDE statement:

- Place all INCLUDE statements at the beginning of your run control report.
- You can specify only one report on each INCLUDE statement.
- See also the guidelines listed under "Defining Constants" in this subsection.

Examples

Include DEFINE statements residing in report 15E0:

```
:include,0,e,15 .
```

Include DEFINE statements residing in report 14I0:

```
:include,0,i,14 .
```

Section 5
Getting Online Assistance

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

This section contains information about online help and MAPPER runs you can use to help you write your own runs.

This section includes:

- Displaying information about MAPPER functions (HELP)
- Creating, modifying, and removing APT tables (BAT)
- Creating input screens and menus (SCGEN)
- Displaying horizontal column positions (CC)
- Displaying field column-characters (FCC)
- Displaying run statement formats (FORM)
- Using a function mask to get format (FORMC)
- Creating run statements (MARS)
- Generating runs (RUN)
- Analyzing your run (RUNA)

Displaying Information about MAPPER Functions (HELP)

There are two methods of accessing help topics: the Help function key and the HELP run.

Accessing Help through the Help Function Key

The Help function key displays information about the task you are performing. You can access help this way by pressing the **Help** function key whenever it is displayed on the function key bar.

A short help screen, menu help, or full-screen online documentation providing information about your task is displayed.

Accessing Help through the HELP Run

The HELP run gives you direct access to the main help menu or to a topic you specify. Enter one of the following formats on the control line. Note that pressing **Quit** from the HELP run always displays the active screen.

1100: On OS 1100 MAPPER Systems, pressing **Quit** from the HELP run redisplays the report or result that was previously on display.

Format 1

```
HELP
```

The main help menu is displayed.

Format 2

```
HELP,topic
```

where:

topic	The code that appears to the right of the topic title on the main help menu or in the lower right corner of each help screen. The code is often the call for the manual function or run statement. The list of all topic codes appears in the help index, available from the main help menu.

The first screen of the topic is displayed.

The following table lists examples of certain help calls and the help topics they display:

Example	Information Displayed
help,pr	Print (PR) function
help,@prt OR help,run,prt	Print (PRT) run statement
help,MGUPIN	Details about the system message designated by the code MGUPIN
help,variables	General information about using variables

Obtaining Online Help for Keys

To obtain a list of some of the most common generic key names used in the documentation, press **KeyHlp** from the first screen that appears when you enter the MAPPER system.

Some MAPPER systems also provide a list of actual key sequences to use on your keyboard for various key actions. If this feature is available, you will see an entry in the first **KeyHlp** screen called **KeyMap**. Press the keys shown to the right of the word **KeyMap** to display this list of actual key sequences.

If **KeyMap** does not appear on the first **KeyHlp** screen, specific information for your keyboard is not available online. Refer to the printed documentation for your terminal.

You can also use the following methods to obtain the list of generic key names:

Method 1

1. Press **Help** until you reach the main help menu.
2. Tab to "Key Assignments for Documentation Key Names."
3. Transmit.

Method 2

1. After signing on to the MAPPER system, move the cursor to the control line.
2. Type **help,key**.
3. Transmit.

Obtaining Online Help for Using the Help System

To obtain help for the online help system, select that topic from the main help menu. Access the main help menu by repeatedly pressing **Help** or by entering **help,help** on the control line.

The online documentation topic about help describes in detail the help system and how to use it. This topic includes how to print topics, locate help topics, use menu access for help, and navigate from topic to topic.

Creating, Modifying, and Removing APT Tables (BAT)

Use the BAT (Build APT Table) run to create, modify, or remove an Application Power Tools (APT) table at the end of your run control report. An APT table is a list of all cabinets, drawers, and reports your run accesses. The table is useful when you use the LISTS run to list all runs and subroutines that access a particular report. This capability lets you determine which runs would be affected by a change to a report.

Note: *The BAT run is part of the APT feature of MAPPER software. For more information about the APT features, see the* Manual Functions Reference. *For information about the LISTS run and how you use it with the BAT run, enter* `apt` *on the control line, tab to* **LISTS**, *and press* **Help**.

Prerequisites

The following prerequisites apply to the BAT run:

- The run control report must be on display in an even-numbered cabinet.
- You must be the last one to update the run control report.
- The run must be registered with an application in your department data dictionary.

How to Access the Run

Follow this procedure to use the BAT run:

1. Display the run control report.
2. Type `apt` on the control line and transmit.
3. Select **BAT** from the APT menu.
4. Fill in the BAT run form. For help while filling in the form, press the **Help** key.

You can also enter `bat` on the control line to display the BAT menu. For help in choosing a selection, press the **Help** key.

Guidelines

If you or other users plan to use the LISTS run to check which runs access a specific report, it is important to keep the APT table current. When a change is made to your run such that it accesses a different cabinet, drawer, or report, use the BAT run to modify the APT table.

To make sure that all runs within your application contain APT tables, use the VALIDATE run. For more information, enter **apt** on the control line, tab to the **VALIDATE** selection, and press **Help**.

For More Information

For more information about the BAT run, enter **apt** on the control line, tab to **BAT**, and press **Help**. For more information about the data dictionary and the runs available with Application Power Tools (APT) feature of MAPPER software, see the *Manual Functions Reference*.

Creating Input Screens and Menus (SCGEN)

Use the SCGEN run to create input screens and menus using a menu-driven interface. With the SCGEN run, you design screens using a visual layout that resembles the actual output screen. You can also include fields in your screens that are already defined in the data dictionary. The SCGEN run automatically creates the Screen Control (SC) code for you to use in your run.

Note: The SCGEN run is part of the APT feature of MAPPER software. For more information about the data dictionary and other APT features, see the Manual Functions Reference.

How to Access the Run

Follow this procedure to use the BAT run:

1. Type `apt` on the control line and transmit.
2. Select **SCGEN** from the APT menu.
3. Select items from the subsequent menus and fill in the forms. For help in completing the menus and forms, press the **Help** key.
4. When the screen design is final, press **Build.**

You can also enter `scgen` on the control line and transmit to display the SCGEN menu. For help in choosing a selection, press the **Help** key.

Outcome

When the screen design is final and you press **Build**, the screen control code is created and placed into a report in drawer A of cabinet 0. Your run can then use the SC run statement to access that report.

For More Information

For more information about the SCGEN run, enter `apt` on the control line, tab to **SCGEN**, and press **Help**. For more information about the data dictionary and the runs available with Application Power Tools (APT) feature of MAPPER software, see the *Manual Functions Reference*.

Displaying Horizontal Column Positions (CC)

The CC run gives a horizontal column position count for a drawer.

The column numbers appear in lines 2 and 3 of the screen. The CC run displays the correct column numbers regardless of the format on display.

To start the CC run, display a report and enter:

```
cc
```

Following is an example of the screen displayed if you start the CC run with report 2B0 on display:

```
Line♦·1          Roll♦·-                                        2B0

         1111111111222222222233333333334444444444555555555566666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890
.DATE   01 JUN 90  05:01:00      REPORT GENERATION   NEWUSER
.Production Status Report                          Corporate Production
*St.Status.By. Product .Serial.Produc.Order.Cust.Produc.Produc. Ship .Ship .Spc.
*Cd. Date .In.  Type   .Number. Cost .Numbr.Code. Plan .Actual. Date .Order.Cod.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
·IP·831224·LS·BLACKBOX1·436767·      ·84389·AMCO·831223·831224·      ·     ·     ·
·IP·831225·LS·BLACKBOX1·436768·      ·84390·AMCO·831223·831225·      ·     ·     ·
·IP·831219·LS·BLACKBOX2·637071·      ·84353·INTR·831218·831219·      ·     ·     ·
```

With the column counts on display, you can perform SOE updates or roll through your report. If you shift or change formats, the column counts are removed from the display.

1100: On OS 1100 MAPPER Systems, the column counts are not removed if you shift or change formats. Press **Paint** to remove them.

Displaying Field Column-Characters (FCC)

The FCC (Field Column-Characters) run displays field headings, the position of the first character in each field, and the size of each field.

To start the FCC run, display a report and enter:

```
fcc
```

Following is an example of the screen displayed if you start the FCC run with report 2B0 on display:

```
Line♦·1          Roll♦··                                        RESULT
.DATE   01 JUN 90  05:01:10     REPORT GENERATION   NEWUSER
.CABINET (0) DRAWER (B) REPORT (2) CHARACTERS (80)
*St.Status.By. Product .Serial.Produc.Order.Cust.Produc.Produc. Ship .Ship .Spc.
*Cd. Date .In.  Type   .Number. Cost .Numbr.Code. Plan .Actual. Date .Order.Cod.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 2-2       12-2         25-6   32-6   39-5  45-4 50-6   57-6   64-6   71-5  77-3
    5-6       15-9
                       ..... END REPORT .....
```

Take note of the column-character positions. If you want, print the result for future reference.

To redisplay the original report, press **Resume**.

Displaying Run Statement Formats (FORM)

The FORM run displays the format of run statements (fields and subfields). It fills in all open function calls in the report with abbreviations for the fields and subfields.

To execute the FORM run, enter the run function calls you want the formats for in your run control report, and enter:

```
form
```

Following is an example of starting the FORM run with some function calls entered:

```
Line♦·1          Roll♦·form                                        59E0
.DATE   01 JUN 90  05:02:00  RID   59E   01 JUN 90  MAPPER
*Run Function Data: Example of use of the FORM run
*=============================================================================.
@SRH
@SOR
@MCH
@DSP

                    ..... END REPORT .....
```

The FORM run inserts the run statement formats directly into the run control report, as shown:

```
Line♦·1          Roll♦·-                                           59E0
.DATE   01 JUN 90  05:03:00  RID   59E   01 JUN 90  MAPPER
*Run Function Data: Example of use of the FORM run
*=============================================================================
@srh,c,d[,r,l,q,lab] o cc ltyp,p [vlines,vls,vrpt] .
@sor,c,d,r o cc ltyp,p .
@mch,ic,id,ir,rc,rd,rr[,lab] o icc iltyp,ip rcc rltyp,rp .
@dsp,c,d,r[,l,tab,f,interim,hold,msg] .
```

Using a Function Mask to Get Format (FORMC)

The FORMC run creates run statements for functions that use function masks, as well as for Read Continuous (RDC), Read Line (RDL), and Write Line (WRL).

To execute the FORMC run, follow these steps:

1. Enter the function call in a run control report including all necessary cabinets, drawers, and report numbers. For example:

   @mch,0,b,1,0,c,1 .

2. Roll the line with the desired statement to the top of the screen.
3. Enter **formc** on the control line.

The function mask or masks appear on your screen. Fill in the options and parameters and modify the mask, if necessary, just as you would for the equivalent manual function.

For RDC, RDL, and WRL, fill in the variables in the desired fields.

The system writes the statements in your run control report. Modify the statements as needed.

To omit the subfields that are unique to runs (such as start line and line quantity) from the generated statement, enter **formc,*x*** where ***x*** is *any* alphabetic character (used only as a flag indicating subfields should be omitted).

To specify formats for functions using double function masks, enter **formc,*f1*,*f2*** where ***f1*** is the format for the issuing report, and ***f2*** is the format for the receiving report.

To omit the subfields that are unique to runs (such as start line and line quantity) from the generated statement for functions using double function masks, enter **formc,*f1*,*f2*,*x*** where ***f1*** and ***f2*** are optional formats and ***x*** is any alphabetic character (used as a flag indicating subfields should be omitted).

Creating Run Statements (MARS)

The MARS (Make a Run Statement) run creates MAPPER run statements and places them in a run control report. If you do not have a run control report, the MARS run adds one for you.

The MARS run is especially useful for capturing functions you use repeatedly in a run control report. You can name the run and execute it at any time. Call your MAPPER system coordinator for more information.

The MARS run prompts you for the information it needs to write the run statements. It writes the statements in the run control report as it creates them.

To execute the MARS run, enter:

mars[,*rd*]

or

mars*[,*rd*]

where *rd* is the report number and drawer (if you already have a run control report), and * means use field names instead of column-character positions in affected statements.

You get a menu of functions. You can do either of the following:

- Tab to the function you want and press **Transmit**.
- Enter the function call at the top of the menu after **Enter call**.

If you need help, enter:

mars,help

If you have the MARS run menu on your screen and you want help with any function, tab to the call you want help with and enter **?**.

Generating Runs (RUN)

Use the RUN run to generate run statements automatically from manual functions you perform at your terminal. The RUN run generates these statements transparently as you execute the manual functions.

The RUN run is especially useful for MAPPER users who do a sequence of repeated tasks on the same reports. By using the RUN run, a user who has no knowledge of run design can easily automate these basic tasks.

For example, suppose that every day you need to search a report for a particular item, sort the result, add the values in a column, and print the result. You can have the RUN run generate run statements for each of these tasks so that all you need to type in is one word, the run name.

Format

```
RUN[*]
```

The RUN run creates run statements using column-character positions in the *cc* field of run statements. To create run statements using field names instead of column-character positions, use the asterisk (*).

Procedure

After you start the RUN run, follow these steps to continue the processing:

1. Do the manual functions for which you want run statements generated.
2. Use the keys listed on the RUN run function key bar to control the operation of the RUN run. See "RUN Run Function Key Bar" in this subsection for details.
3. Press **Exit** or type a caret (^) and transmit when you are ready to register the statements as a run or when you are ready to execute the run statements. See "Registering Run Statements as a Run" in this subsection.

Your terminal is placed under control of the RUN run. If no report is on display, the Automatic Run Generation active screen appears; otherwise the run redisplays the report.

RUN Run Function Key Bar

Use the keys listed on the RUN run function key bar to control the operation of the RUN run. Following are descriptions of the keys on the function key bar:

Key	Description	
DspRun	Displays the generated run statements in a result.	
Re-Dsp	Displays the report you were processing. Use this key after you use the **DspRun** key to continue executing other manual functions after looking at the generated run statements.	
AcScrn	Displays the Automatic Run Generation active screen.	
Exit	Displays the Automatic Run Generation Terminated screen, where you can register the run or use the following function keys to display or execute the run statements:	
	DspRun	Displays the generated run statements in a result.
	Run	Executes the run statements.
		Note: *For each Display (DSP) statement in the run, press* **Resume** *to continue processing the next statement.*
Quit	Terminates the RUN run and displays the active screen.	

Displaying Reports and Results in Your Generated Run

The RUN run does not generate a Display (DSP) statement for each function result, even though you see the results as you execute the functions. It generates a DSP statement only after the final result in the run.

If you want to display each result in the process, use the manual Display Report (D) function. For example, if you want to search and sort a result and display both results, type `d -` after each result appears and then transmit.

Registering Run Statements as a Run

To register your run statements as a run, in the Automatic Run Generation Terminated screen, type the letter of the drawer where you want the run to reside and transmit. The RUN run does the following:

- Duplicates the result containing the run statements in the drawer you specify in your current cabinet.
- Registers the run with a run name of your user-id. The run is registered so that only your user-id can execute it.

1100: On OS 1100 MAPPER Systems, if your user-id begins with a number (for example, 7NEWUSER), you can still execute it by entering `run 7newuser`.

- Writes over the run registration of any previously generated run with the same run name. To retain the old run, have the coordinator reregister the old run with a new name (the old run is still intact, only its name is no longer available).

Note: *Automatic run registration may be restricted on some systems. See your MAPPER system coordinator if you have problems.*

RUN Run Limitations

You can use the following manual functions while under control of the RUN run, but the run does not generate run statements:

L Restores the control line; same as typing `pnt` and transmitting.

RSM Resumes; same as pressing **DspRun**.

If you select a function that is not available to the RUN run, a system message appears.

The RUN run generates run statements only for statements that have equivalent manual functions. For example, the Load Variable (LDV) statement does not have an equivalent manual function.

The following functions generate run statements but terminate automatic run generation:

X Generates a Sign off MAPPER Software (XIT) statement and displays the Automatic Run Generation Terminated screen.

^ Generates a Release Display (REL) statement. The run executes, releases the screen, and displays the Automatic Run Generation Terminated screen.

You cannot generate output screens or capture user input from any other source (for example, by using the reserved words INPUT$ or INVAR$).

You cannot reuse a function mask after a system message appears on the control line. Instead, you must reenter the function call, complete the function mask a second time, and transmit.

You cannot use the SOE Update function during run generation. However, when the generated run is executing, you can update reports in the usual way.

To update displayed reports, use the Change (CHG) function or an updating function such as the Search Update (SU) function followed by the Update (UPD) function. You can also use a line change function such as the Add Line function.

Analyzing Your Run (RUNA)

This section does not apply to the Personal Computer MAPPER System.

The RUNA run identifies certain inefficient run design techniques. The RUNA run is not an absolute or total test of run design quality, but it does give you a good indication whether your run is acceptable. Carefully review any indications of inefficient techniques identified in the analysis and try to correct them. If you have a problem correcting something, call the MAPPER system coordinator.

Even if your run passes the RUNA run analysis satisfactorily, it may not be ready for use in production. The MAPPER system coordinator must still approve your run.

The RUNA run has its own built-in online Help. To read it, enter:

```
runa
```

or

```
runa,help
```

(Note that if you enter `runa` without a log list on display, you get the online help for the RUNA run.)

Before executing the RUNA run against your own run, do the following:

- Make sure you have a Log for Analysis (LOG) statement at the beginning of the run but after the label table, if there is one.
- Replace the Release Display (REL) statement (if used) with a GTO END statement to keep the log list intact after the run completes.
- Execute your run.

Follow these steps after your run completes:

1. Press **Resume**.
2. When the log list appears on your screen, enter `runa` to get the RUNA analysis result.

   At this point, you can print the result, or go to the next step.
3. If you want even more details, press **Resume**.

The RUNA run appends a detailed explanation of recommended corrective guidelines to the end of the result. Print the result if you want to refer to it in the future or press **Resume** to return to the log list.

Section 6
Designing and Debugging Runs

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

This section helps you begin planning your run. It also explains how to handle reports and results, how to debug your run, and how to use conditional statements and subroutines. You also learn some methods for writing more efficient runs.

This section includes:

- Planning and registering your run
- Handling reports and results
- Debugging your run
- Using conditional statements
- Using subroutines
- Writing efficient runs

Planning and Registering Your Run

When you are ready to write a MAPPER run, follow this logical step-by-step procedure:

1. Plan the run. Determine which statements you want to execute and whether you are going to use any logical decisions, paths, or loops. It may be helpful to draw a flow chart, as shown in Figure 6-1, to map out the processing steps.

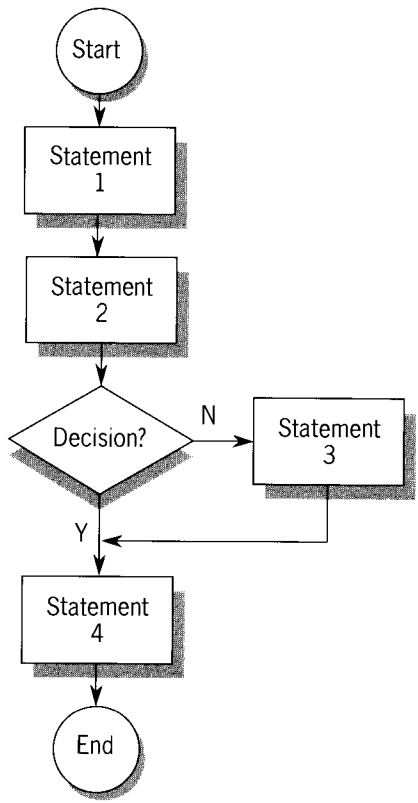


Figure 6-1. Sample Flow Chart

If you are unsure what effect a run statement has in a run, you can usually test the statement by running it separately. If it uses a manually executable processing function, test it manually first.

2. Register your run control report through your MAPPER system coordinator. First, execute the Drawer Table of Contents (T) function to see which drawer is available in your cabinet for MAPPER runs. Next, use the Add Report (AR) function to add a report in that drawer. Give the coordinator such information as report number, drawer, cabinets to be used, and your proposed run name. Explain the general plan of the run to the coordinator to assess its impact on the MAPPER system. If everything is acceptable, the coordinator registers the run for online debugging.
3. Enter the required run statements in the run control report. With manual updating, natural pauses between transmissions disperse the processing load; but in a run, where the statements are executed rapidly one after the other, virtually no pauses occur between the execution of statements. A run that executes several individual functions may put a severe load on the MAPPER system. You should consider the performance impact on the system and adjust run user response expectations accordingly. Consider using predefined constants to organize the information needed in your run (such as database location, parameters, and so on) in one place. See "Using Predefined Constants" in Section 4.
4. When the run is designed and ready for use, if you used predefined constants, be sure to execute the Build Label Table (BLT) function to convert the constants into actual values.
5. When a run is ready for production, execute the run with logging turned on. Have the coordinator assess the impact of your run on the MAPPER system by examining the log list. The coordinator may suggest improvements for your run.

PC MAPPER: Run logging is not available on Personal Computer MAPPER Systems.

After accepting your run and obtaining a final log list, the coordinator registers it by its name. The coordinator may restrict user accessibility, cabinet accessibility, time of execution, input/output quantity, logic line count, and station numbers for your run.

If you change your run significantly, the coordinator must reanalyze it.

6. **1100:** You should give the run control report a report password and a save flag in this format:

   .@YYMMDD

   The save flag must start in the first column of line 2 (the line below the date line) and must be a period line (notice the period beginning in column 1).

Handling Reports and Results

You can process several reports in a single MAPPER run. The simplest run acts on a report or result from a previously executed function and produces a result or updated report, as shown in Figure 6-2.

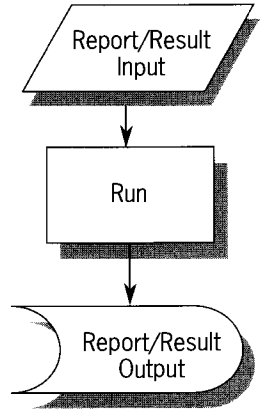


Figure 6-2. Processing a Report or Result

You can use several reports and results from different cabinets for input, provided your coordinator registered the cabinets for access. Figure 6-3 shows how you use reports and results from different cabinets in a run.

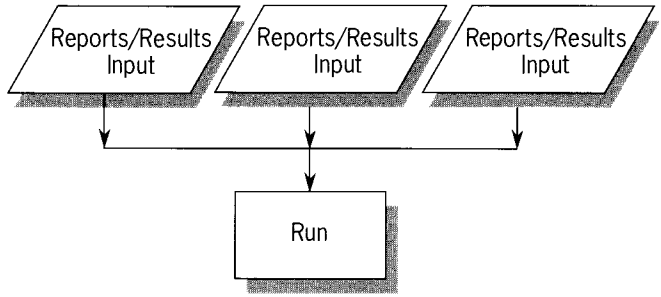


Figure 6-3. One Run Using Several Reports and Results

The Output Area

The output area is a temporary scratch area that you build in your run control report to hold information. This information is composed of output lines, which are lines of data that do not have at signs (@) or colons (:) in column one (and are not continuations of run statement lines). The output area may contain, for example, messages or special screen displays that you want to display later in the run.

To examine the output area at any time during the run, enter a GTO END statement. This displays the contents of the output area as a result.

You can use output area data as a result at any time by executing a Break (BRK) statement. The BRK statement places the output area into the current result and creates a new output area. You can then use the Display Report (DSP) or Output (OUT) statements to display the result. The new output area is empty, so you can place new data in it.

See BRK, DSP, and OUT in Section 7.

Results

A result is a temporary copy of data obtained by executing a MAPPER function or run statement. The current result is the latest result, and its report number is always -0. Only one current result (result -0) exists at any moment. In addition to the current result, a MAPPER run can save up to eight results for subsequent access. These are only saved for the duration of the run. To save up to eight results, rename them with a Rename (RNM) statement.

To access results, specify the result identifier (the renamed result, such as -1, or the current result, -0) in the appropriate subfields of a MAPPER run statement. To access a report or result on display before the run started, refer to it as -0 until your run creates another result.

The Relationship between Output Area and Results

Do not confuse the output area with a result. You create a result using a function or run statement, such as Search (SRH) or Totalize (TOT). You build the output area by adding output lines to your run control report. You can create an output area without affecting the current result or previously renamed results. You can also turn the output area into a result using the Break (BRK) or Break Graphics (BRG) statements.

Figure 6-4 shows how runs use the output area and results. For more detailed information about the output area and results, see the *Run Design Training Guide*.

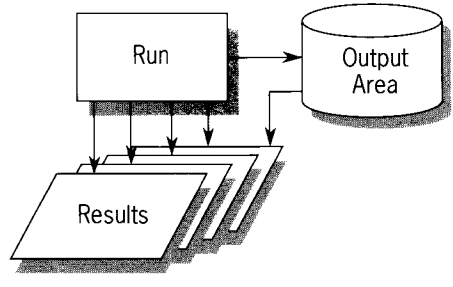


Figure 6-4. Relationship between Output Area and Results

Debugging Your Run

Debugging is an important step in creating a MAPPER run. For the run to perform as it should, any errors in the statements must be corrected.

Because run statements use a precise format and syntax, MAPPER software does not recognize any statements that deviate from these rules.

You can use any of the following methods to debug your run:

- Interactive debugging
- Online help system
- Checkpoint displays
- Run Debug (RDB) statement
- Register Error Routine (RER) statement

Interactive Debugging

When your run stops because of an error, you see a one-line system message and the erring run statement line at the top of the screen. You can interactively correct the error. This technique works well for simple errors, such as an incorrect drawer or a missing comma in a run statement.

With the run on display, correct the erring line, move the cursor to the next line and transmit. The change is added to your run control report. To test the corrected run, execute it again.

Online Help System

You can also use the online help system to debug your run. When the run stops because of an error, a system message is displayed on the control line.

If you need more information about the error, press the **Help** key. The system responds by displaying information about the error for which the system message appeared. After you read the information, you can press **Help** for more detailed online documentation or press the **Return** function key to return to the run control report where the error occurred.

Checkpoint Displays

If your run has several stages of processing, add DSP statements to display intermediate results. You can check the results to see if the previous run statements executed properly. If you write your run in modules, where each module performs a specific task, you can easily run each module with a checkpoint display to test it. As you debug the run, take the checkpoint displays out.

Run Debug (RDB) Statement

You can use the RDB statement to debug your run while it executes. This is different than interactive debugging because you can do the following:

- Display or change the value of any variable
- Examine reports and results
- Stop the run at a defined variable or label

See RDB in Section 7.

Register Error Routine (RER) Statement

To help you debug runs, use error subroutines to capture pertinent information such as the system message number and the failing function call, line number, and so on. Register an error routine with the RER statement. See RER in Section 7.

Using Conditional Statements

You can change the flow of a run using branching, conditional branching, looping, and subroutines.

Branching

To skip from one part of the run to another, use a Go To (GTO) statement. Use a GTO statement to jump to a specific label, to a line preceding or following the current line, or to a run in another run control report. See GTO in Section 7.

Since you cannot use a label more than once in the same run, it uniquely identifies a line. See "Using Labels in Runs" in Section 2.

Following is the format for the GTO statement for jumping to labels:

```
@GTO lab .
```

where *lab* specifies the label where the run should continue execution.

Example

Label the line containing the Change Variable (CHG) statement as 1 and jump to it:

```
@001:chg v4a3 abc .
     .
     . (more run statements)
     .
@gto 001 .
```

Conditional Branching

If you want the run to branch only in certain circumstances, use conditional branching. Conditional branching uses an If Conditional (IF) statement in the following format:

```
@IF val1 op val2 stmt1 . ;stmt2 .
```

Following is a description of the fields:

Field	Description
val1	A value, such as a variable.
op	The operator to use.
val2	A second value.
stmt1	The run statement to process if the comparison is true (omit the @ sign before the run function call for this statement).
stmt2	The run statement to process if the first condition is not true.

Example

In the following example, if v9 equals the letters abc, the run is to skip to the line that is labeled 15. If v9 does not equal the letters abc, the statement written after the semicolon (;) is to be processed (in this case, the run is to skip to the line labeled 20).

```
@if v9 = abc gto 015 ;gto 020 .
```

See IF in Section 7.

Looping

To repeat an operation more than once, you can loop rather than jumping back and forth in a run.

To set a loop, place a label at the beginning of the run statements through which you want to cycle. Then, use a GTO statement at the appropriate point in the run.

To loop a given number of times or until another condition is met, use an IF statement to build in a test within the loop. Then, when the condition is met, the IF statement may specify a label at which the run continues.

Using Subroutines

A subroutine is a set of run statements that forms a small run by itself. You can use it instead of a loop to repeat a process.

Use the Call Subroutine (CALL) statement to save the contents of all variables currently defined in the run and transfers control to an internal or external subroutine.

You can pass up to 40 variables to the subroutine. You can manipulate these variables within the subroutine and pass them back to the calling run without affecting any of the other currently defined variables.

1100: On OS 1100 MAPPER Systems, you can also pass parameters that the system does not change upon return to the calling run.

Format

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

1100:

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

See CALL in Section 7 for an explanation of CALL statement fields and subfields, and for details and examples.

You can also use the Run Subroutine (RSR) statment to transfer control to subroutines. See RSR in Section 7.

Writing Efficient Runs

The tips in this section can help you improve your run by reducing I/O requests and logic lines processed (LLPs).

Run Control Reports

Follow these guidelines regarding run control reports:

- Use a space-period-space to terminate the logic scan of a run statement.
- Place short comments after the space-period-space to document your run.
- Place as many run statements on one line as possible within logic loops.
- If you use DEFINE and INCLUDE statements when you develop the application, use the BLT function to convert the defined constants. Save the result in a different report and use it for the production run.
- Use the BLT function to build a label table even when not using DEFINE statements. This helps reduce overhead because whenever the run refers to a label, it can go directly to the label address without scanning the run control report for the label.
- If possible, use an 80-character (or narrower) drawer for your run control reports. In general, efficiency decreases as run control report width increases.
- With the BRK statement, you can set up your output area as greater than 80 characters (by specifying a wider drawer in the BRK statement), and you can write to the output area using variables, thus writing beyond column 80.

- **1100:** In addition to these guidelines for run control reports, also note that choosing the right character set for your database can help improve the performance of the runs that access it.

  For standard column-formatted reports where lowercase characters are not absolutely necessary, use full character set upper (FCSU) instead of full character set (FCS). The system searches and sorts much more efficiently with FCSU.

Analysis and Registration

Follow these guidelines regarding run analysis and registration:

PC MAPPER: The LOG statement and RUNA run are not available on the Personal Computer MAPPER System.

- Include a LOG statement to analyze your runs. When your run terminates normally, resume to display the LOG result; then execute the RUNA run to analyze your LOG results.

  Note: *After final debugging of your run, remove the LOG statement from your run control report.*

- The RUNA run identifies certain inefficient run design techniques. It is not intended to be a total test of the run design quality, but it does give you a good indication of whether your run might be acceptable.

  For an online explanation of the RUNA run, enter:

  ```
  runa,help
  ```

  See "Analyzing Your Run (RUNA)" in Section 5.

- Have your MAPPER system coordinator place frequently used runs near the beginning of the run registration report.

- **1100:** If your department has several hundred runs registered, have the coordinator sort the run registration report.

- If you think that several users might execute your run, duplicate the run and execute the different versions under different department numbers.

- You or your coordinator can use the LOGLA run to analyze the LOG function result, function by function. The LOGLA run displays the total I/Os, logic lines processed, and total lines processed.

Loading Variables

Follow these guidelines when loading variables:

- Define variables in the run statements instead of setting up a table of predefined variables at the start of a run. Predefining variables causes extra processing overhead.
- Use a Justify Variable (JUV) statement instead of a CHG statement to justify the contents of variables that contain numbers.
- Use a Load Variable (LDV) statement instead of a CHG statement to load variables with data; use an LDV statement to right-justify, left-justify, and center variables.
- Use the P option with an LDV statement rather than an IF/CHG loop to delete unnecessary leading or trailing spaces from variables.

Statements and Functions

Follow these general guidelines regarding statements and functions:

- If available for a function, use the D and H options whenever possible to eliminate detail and heading lines.
- Use Increment Variable (INC) or Decrement Variable (DEC) statements whenever possible to add to and subtract from variables.
- Use an Arithmetic (ART) statement for complex equations only.
- Use a CHG statement for simple arithmetic operations rather than an ART statement whenever possible. The CHG statement is an internal operation, processed by the run interpreter; the ART statement calls a separate MAPPER function (Arithmetic).
- In SRH statements, use the B option to stop a search after a find when you know there is only one item to be found.
- Sort the reports on the intended match fields before using a Match (MCH) statement; otherwise, the MCH statement first sorts the data, then matches it, then sorts it again to return it to its original state. Use the P or Q option when matching the presorted data.

- Use a label with the P option in the MCH statement so that the run continues at another routine if the data is unsorted. At this routine, you can sort the data and return to the MCH statement for another attempt.
- Use a Binary Find (BFN) statement to find a single item in a large report or drawer, and be sure the data is presorted on the field or fields being searched.
- Use a Calculate (CAL) statement instead of multiple ART and TOT statements if you need to perform several calculations. Use a TOT statement if you need to perform only one calculation.
- Use a Find and Read Line (FDR) statement followed by a Read Line Next (RLN) statement instead of a Find (FND) and Read Line (RDL) combination.
- **1100:** If available, use the E option to estimate the number of lines in the result if you expect the result to be larger than about 500 lines. This reserves space for the result beforehand and avoids the movement of the result from a smaller to a larger area as the system builds it.

Updating Reports

Follow these guidelines when updating reports:

- Write lines into results whenever possible. The Find/Write Line (FND/WRL) combination is more efficient than Search Update/Totalize (SRU/TOT).
- Add multiple blank lines to a report whenever you need more data lines; then update the added lines. Either find a blank line and write it, or write into a heading line the line number of the next line to write or the last one written.
- Write spaces into existing lines instead of deleting them with a Delete Line (LN-) statement so that MAPPER software does not need to write the entire report to mass storage.
- Enter **y** in the ***ntuld?*** field when using a WRL statement; it is 50 percent more efficient.

- Rename results instead of using pivot reports. (Pivot reports are created in the run, used as accumulators, then deleted at the end of the run.)
- When creating results in the output area, include a heading divider line before the first data line. When processing the result in subsequent functions, the system looks for the heading divider line to find the first data line. Including it in the result prevents the system from scanning the result unnecessarily in search of the first data line.

- **1100:** Compared to traditional LOK/WRL sequences for updating, deferred updating may cause extra lines to be processed and may increase response time. Follow these guidelines to minimize the impact on the system:
  - Try to keep all reports as small as possible.
  - Limit the amount of time the report is locked.
  - If possible, limit the number of Write Line (WRL) statements to one or two.

  Note: *If you must use several WRL statements on a report being accessed by other users, consider using the ccpy subfield in the WRL statement, which forces the system to write a complete copy of the report. Though this may cost additional I/O overhead to update the report, it can considerably decrease the cost of other users attempting to access the same report.*

Logic

Follow these guidelines to improve logic:

- Put as many IF, GTO, CHG, and LDV statements as possible on one line, especially if they are used in a loop against an RLN statement. Keep the most frequently used logic paths short.
- Loop on an RLN statement rather than on an RDL statement.

Batch Processing

This section on batch processing applies only to the OS 1100 MAPPER System.

Follow these guidelines for batch processing:

- When bringing large volumes of data from the batch environment into the MAPPER system, use one of the following methods to split the data into properly sized reports ahead of time:
  - Use the *l* and *q* subfields in the Retrieve File (RET) statement to retrieve a block of data from a very large file, or else have each report in a different element in a program file and retrieve the elements one at a time.
  - Instead of passing data in a BPRUN$ command, start a run through the batch port that sends a message to your station indicating that the data is ready to be retrieved, then manually retrieve the data from separate elements or start a new run to retrieve the data. This procedure frees the batch port for other users.
  - Use multiple SYM statements to bring in the data one report at a time.
- Use a RET statement to retrieve data from the batch environment instead of passing it through the batch port.
- When using a RET statement, enter the expected number of lines in the *q* subfield. This allocates sufficient storage space for the data.
- In applications where you are transferring identical data to the batch environment each time it is needed, use an Element (ELT) statement. The ELT statement transfers data directly to a program file where multiple batch jobs can access it.

Appendix A
Summary of Statements and Options

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

This appendix contains the following:

- 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 are the formats for run statements available in all MAPPER systems, including variations to the format of common statements (such as AUX and PRT). The Systems column denotes the systems to which the run statement or format variation applies.

Systems	Run Statement
All	@ADD,*ic,id,ir,rc,rd,rr* .
All	@ADR,*c,d*[,*r,lab vtitle*] . (RPT$ or STAT1$ = new report number.)
All	@ART *exp vrslts* .
A Series	@ASR[,*rmt?,xmt?,ci,di,ri,hi?,in,co,do,ho?,out,lab*] *fn* [*args*] .
1100	@AUX,*c,d,r,sn,dev*[,*dlnos?,f,tabs?,dhdgs?,d1char?,lsp,transp, unit,,spcc*] .
All except 1100	@AUX,*c,d,r,sn,*[*dev,dlnos?,f,,dhdgs?,d1char?,lsp,,,sl,spcc*] .
All	@BFN,*c,d*[,*r,l,lab*] *o cc ltyp,p* [*vrpt,vlno*] .
All	@BLT,*c,d,r*[,*lab*] .
1100	@BR[,*sn,lab*] *run*[,*vld*] .
All except 1100	@BR[,*c,d,r*] *run*[,*vld*] .
All	@BRG[,*c,d,q*] .
All	@BRK[,*c,d,q*] .
All	@CAB,*c,d* .
All except 1100	@CAH,*c,d,r* .
All	@CAL,*c,d,r*[,*l,q,lab*] *o cc ltyp,p eq* [*vrslts*] .
1100	@CALL[,*c,d,r*] *lab* ([*p,p,...*]) .
All except 1100	@CALL[,*c,d,r*] *lab* ([*v,v,...*]) .
All	@CAR .
All	@CAU,*c,d,r*[,*l,q,lab*] *o cc ltyp,p eq* [*vrslts*] .
All	@CER .
All	@CHD[,*c,d,r,rel?*] *lab* .
All	@CHG *v* {*exp*|*vld*} . or @CHG *rw v*[,*v,...,v*] .
All	@CLK .
All	@CLT,*c,d,r*[,*lab*] .
All	@CLV[,*startv,q*] .
All	@CMP,*c1,d1,r1,*[*lin1*],*c2,d2,r2*[,*lin2,q,lab*] *o cc1 ltyp1,p1 cc2 ltyp2,*
All	*p2* [*vlno1,vcol1,vlno2,vcol2*] .
All	@CMU .
1100	@CNT,*c,d*[,*r*] *o cc ltyp,p* [*vrslts*] .
1100	@CPY,*o ls,lq,lfn*[,*le,lv*] *rms*[,*rmq,rmfn,rme,rmv*] *rmu,rmd,rmpw*[,*l*] .
All	@CSR .

continued

continued

Systems	Run Statement
All	@DAT,*c,d,r o cc ltyp,p* .
All	@DC *eq vrslts* .
1100	@DCPY[,*lab*] *host,fn,rhost,fn,type*[,*pos,transl*] .
1100	@DCR,*c,d,r,key* .
All except PC MAPPER and 1100	@DCR,*c,d,r,key*[*-level,,dspscram?*] .
1100	@DCRE[,*lab*] *host,fn*[,*dev,init,gran,maxfs,vol,access*] .
All	@DCU .
1100, U Series, UNIX	@DDI,*c,d*[,*lab,edsp?,d/p,db*] *'id col tbl'* [*vstat,vcol,vqty*] .
All	@DEC[,*n*] *v*[,*v,...,v*] .
All	@DEF[,*o,lab*] *setv,testv* .
All	@DEL .
1100	@DEV,*sn*[,*dev,unit,lab*] .
All except 1100	@DEV,*sn*[,*dev,,lab*] .
All	@DFU[,*lab*] *c,d,r*[,*c,d,r,...,c,d,r*] .
All	@DIR[,*lab*] *name* [*vcabinet,vdrawer,vrpt,vhirptr*] .
1100	@DIS,*c,d*[,*r,f,lab*] *code,dev,fn*[,*ext,tabs?,hdgs?,transp?*] .
1100	@DLL,*c,d,r,trfadr,dev*[,*proglab*] .
All	@DLR,*c,d,r*[,*lab*] .
1100	@DPUR[,*lab*] *host,fn* .
1100	@DRW,*c,d*[,*lab vcpl,vcs,vmfno,vmfn,vnxrd,vhirptd,vlnd, vrptsd,vrlmt,vllmt*] .
All except 1100	@DRW,*c,d*[,*lab vcpl,,,,vnxrd,vhirptd,vlnd,vrptsd,vrlmt,vllmt*] .
1100	@DSF,*c,d,r,pn,tabp,o* .
1100	@DSG,*c,d,r*[,*display,interim?,,sn,lab*] .
All except 1100	@DSG,*c,d,r*[,*display,interim?*] .
All	@DSM,*c,d,r,lmsg*[,*tabp,erase?,interim?,pdq,dc,dd,dr,dspl,dspf*] .
All	@DSP,*c,d,r*[,*l,tabp,f,interim?,hold,msg*] .
All	@DSX,*c,d,r*[,*l,tabp,f,,hold,msg*] .
All	@DUP,*c,d,r*[,*rc,rd vtitle*] . (RPT$ or STAT1$ = new report number.)
All	@DVS[,*c,d,r,lab*] *field*[,*field,...,field*] *v*[,*v,...,v*] .
1100	@ECR,*c,d,r,key*[,*hdgs*] .
All except 1100 and PC MAPPER	@ECR,*c,d,r,key*[*-level,hdgs?,,dspscram?*] .
1100	@EL-[,*lab*] *qual,fn*[,*cyc,elt,ver*] .
1100	@ELT,*c,d,r*[,*lab*] *qual,fn*[,*cyc,elt,ver,mapperf?,hdgs?,cs,newcyc?*] .
All	@ESR[{,*q*|,*-q*}] .
All	@EXT .

continued

continued

Systems	Run Statement
1100, U Series, UNIX	@FCH,*c,d*[,*lab,o,adj,skip,q,edsp?,wrap,info,vert,sort,n-char, dst,sp,db,Fs*] *'id col tbl [WHERE clause]'* [,*c,d,r*] [*vstat,vcol,vqty*] .
All	@FDR,*c,d*[,*r,l,q,lab*] *o cc ltyp,p* [*vrpt,vlno*] .
All	@FIL,*c,d,r*[,*mapperf?,hdgs?,lab*] *fn* .
All	@FKY[,*c,d,r*] .
All	@FMT[,*c,d,r*] *field*[,*field,...,field*] .
All	@FND,*c,d*[,*r,l,lab*] *o cc ltyp,p* [*vrpt,vlno*] .
All except PC MAPPER	@GOC,*c,d,r*[,*lab*] *ulvl?*[,*notxt?,ige?,igcz?,fxcz?*] *optmz?*[,*outrslt?, unp? vlinesi,vlineso,vco,vbuffz*] .
All except PC MAPPER	@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*] .
All	@GTO {*lab*|END[,*n,n,n*]|LIN[+]*n*|LIN-*n*|RPX *r*} . (+ or - is the number of lines following or preceding the present line.)
U Series and BTOS	@HOF .
U Series and BTOS	@HRD,*ic,id,ir,rc,rd*[,*rr,lab vmsg*] .
U Series and BTOS	@HRN[,*lab*] *"run statements"* [*vmsg*] .
All	@HSH *v=vld,min-max* .
U Series and BTOS	@HST,*site-id*[,*rmu,rmd,rmpw,trnrpt,scl,col,mtr*] .
U Series and BTOS	@HWR,*ic,id,ir,rc,rd,rr*[,*lab vmsg*] .
All	@IDU,*c,d*[,*q,user,sdate,endate,srpt,endrpt vrpts,vlines,vrptsd, vhirptd*] .
All	@IF[,C] *val1 op val2* [{,*val3*|*& op val3*}] *stmt1* . ; [*stmt2*] .
All	@INC[,*n*] *v*[,*v,...,v*] .
All	@IND,*c,d*[,*q,lab*] .
All	@INS *vld substrv* .
All	@ITV[,*lab*] *v*[,*v,...,v*] .
All	@JUV,*o v*[,*v,...,v*] .
All	@KEY .
All	@LCH,*c,d,r*[,*l,lab*] *o cc tgtstr/replstr* [,*vlines,vrpt*] .
All	@LCV[,*lab*] *o v tgtstr*[*/replstr vpos,voccs*] .
All	@LDA[,*o*] *nametypesize*[*n*]=*vld*[,*vld,...,vld*] .
All	@LDV[,*o*] *v=vld*[,*v=vld,...,v=vld*] . or @LDV,*o v*[,*v,...,v*] .
All	@LFC *v* .
All	@LFN[,*c,d,r,tics?,lab*] *cc v*[,*v,...,v*] .

continued

continued

Systems	Run Statement
1100, U Series, UNIX	@LGF[,*lab,rb,db*] .
1100, U Series, UNIX	@LGN[,*lab,edsp?,c,d,db*] o[,*lang,user,pw vdbt*] .
All	@LLN,*c,d,r*[,*lab*] *vlines* .
All except BTOS and PC MAPPER	@LMG,*ic,id,ir,rc,rd,rr*[,*lab*] .
All	@LN+,*c,d,r,lb4,q*[,*predfl*] .
All	@LN-,*c,d,r,l,q* .
All	@LNA,*c,d,r,l*[,*q,b*] .
All	@LNG,*n* .
All	@LNI,*c,d,r,lb4,*[*x*],*l*[,*q*] .
All	@LNK *run*[,*vld*] .
All	@LNM,*c,d,r,lb4,*[*x*],*l*[,*q*] .
All	@LNP,*c,d,r,lb4*[,*b*] .
All	@LNX,*c,d,r,l,x*[,*q*] .
All	@LNY,*c,d,r,l*[,*q,b*] .
All	@LOC,*c,d,r*[,*l,lab*] *o cc tgtstr* [*vcol,vlno,vrpt*] .
All except PC MAPPER	@LOG .
All	@LOK,*c,d,r*[,*lab*] .
All	@LSM,*msgno*[,*lab*] *vmsg* .
1100	@LZR,*c,d,r*[,*lab vlines,vcpl,vhdgs,vcs,vupds,vdept,vuser,vrpw,vwpw*] .
All except 1100	@LZR,*c,d,r*[,*lab vlines,vcpl,vhdgs,,,vdept,vuser,vrpw,vwpw,vlgn*] .
All	@MAU,*ic,id,ir,rc,rd,rr*[,*lab*] *o icc iltyp,ip rcc rltyp,rp* .
All	@MCH,*ic,id,ir,rc,rd,rr*[,*lab*] *o icc iltyp,ip rcc rltyp,rp* .
1100	@MSG *vld* [*vrsp*] .
All except 1100 and PC MAPPER	@MSG *vld* .
All	@NET,*net-id*[,*site-id,rmu,rmd,rmpw,trnrpt,mtr,lab*] .
All	@NOF .
All	@NRD,*ic,id,ir,rc,rd,rr*[,*lab vmsg*] .
All	@NRM "*cmd*" .
All	@NRN[,*lab*] "*run statements*" [*vmsg*] .
All	@NRT ["*cmd*"] .
All	@NWR,*ic,id,ir,rc,rd,rr*[,*lab vmsg*] .
All	@OK[,*lab,vrsp*] .
PC MAPPER	@OS2[,*c,d,,,lab*] *o* '[*cmd*]' .
All	@OUM,*ic,id*[,*ir,if,rc,rd,rr,rf,title*] .

continued

continued

Systems	Run Statement
1100	@OUT,*c,d,r,l,q*[,*outl,tabp,erase?,interim?,pdq,protect,fxmt?,outsp,blink?,sn,lab*] .
All except 1100	@OUT,*c,d,r,l,q*[,*outl,tabp,erase?,interim?,pdq,protect,fxmt?,outsp?,blink?*] .
All	@OUV[,*scl,col*] *vld* .
All	@PEK[,*startv,q,level*] .
All except 1100	@PNT .
	@POK[,*startv,q,level*] .
	@POP[,*startv,q,level*] .
1100	@PRT,*c,d*[,*r,dlnos?,f,prtsite,cys,all?,lsp,banner,formsid,hdgs?*] .
All except 1100	@PRT,*c,d*[,*r,dlnos?,f,prtsite,cys,all?,lsp,dstn,,hdgs?*] .
All	@PSH[,*startv,q,level*] .
1100	@QCTL[,*c,d,r,lab*] *func*[,*p*] *v*[,...,*v*] .
1100	@QREL .
1100	@QRSP[,*c,d,r,l,q,,eol,tabs?,err,,lab*] .
1100	@QSND[,*c,d,r,l,q,,eol,tabs?,,sn,oq,acct,,lab*] *dnfunc,dnqu,*[*user,dept,pw,dnrun,vld*] .
1100	@QSNR[,*c,d,r,l,q,,eol,tabs?,,sn,tmo,acct,,lab*] *dnfunc,dnqu*[,*user,dept,pw,dnrun,vld*] *rspfunc*[,*rspqu,user,dept,pw,rsrun,vld vstat*] .
1100, U Series, UNIX	@RAM,*c,d,r*[,*lab,o,adj,dup/col,iqty,cqty,edsp?,cstat,istat,n-char,db*] *'table*[,*options*]*' vstat,vcol,vins,vcom,vnbq,vdpk*] .
All	@RAR{,*c,d,r lab* | *lab*} .
All	@RDB .
All	@RDC,*c,d,r*[,*l,q,ltyp,lab*] *cc vdata* . (Place next line in output area.)
All	@RDL,*c,d,r,l*[,*lab*] *cc vdata* .
1100	@REH,*c,d,r*[,*lab*] .
All	@REL .
All	@REP[,*ic,id,ir*],*rc,rd,rr* [*vtitle*] .
All	@RER{,*c,d,r lab* | *lab*} .
1100	@RET,*c,d*[,*lab*] [*qual*],*fn*[,*cyc,elt,ver,mapperf?,hdgs?,l,ststr,q*] .
All except 1100	@RET,*c,d*[,*mapperf?,hdgs?,lab*] *fn* .
All	@RETURN .
All	@RFM,*ic,id,ir,rc,rd,rr o icc ,ip rcc ,rp* .
All	@RLN[,*l,lab*] *cc vdata* .
All	@RMV[,*level*] .
All	@RNM[,*c,d,r*] *-n* .
All	@RPW{,*pw1* |,*pw1,pw2* |,,*pw2*} .

continued

continued

Systems	Run Statement
All	@RRN[,*lc,ldr,lr*] *run*[,*vld*] *rms,rmu,rmd*[,*rmpw,rmc,rmdr*] .
1100	@RS[,*o,run,sn*] .
1100	@RSI[,*site-id,tmo*] *user,pw*[,*acct,qual,fn,cyc,elt,ver,q*] .
All	@RSL,*c,d,r* .
All	@RSR{,*c,d,r lab* | *lab*} .
1100	@RTN,*rmc,rmd,rmr*[,*rmf*] *lc,lcpw,ldr*[,*lf*] .
All except 110	@RTN,*rmc,rmd,rmr* .
All	@RUN {*run*[,*vld*]|"*cmd*"} .
All	@SC[,,,,,,*tabp*] *o scmnd* . (format 1)
All	@SC,*c,d,r*[,*l,q,tabp*] *o* [*fldtxt*] . (format 2)
All except 1100	@SCH[,*date,time*] *rst* .
1100	@SEN,*c,d,r,sn*[,*ack?,lab*] .
All except 1100	@SEN,*c,d,r,sn*[,*sl,ack?,lab*] .
All	@SFC[,*c,d,r*] *vld* .
1100	@SNU,*c,d,r,user*[,*dept,,ack?,lab*] .
All except 1100	@SNU,*c,d,r,user*[,*dept,sl,ack?,lab*] .
All	@SOR,*c,d,r o cc ltyp,p* .
1100, U Series, UNIX	@SQL[,*lab,edsp?,c,d,cstat,db*] *'syntax'* [,*c,d,r*] *vstat* [,*vcol,vretn, vretn,...vretn*] .
All	@SRH,*c,d*[,*r,l,q,lab*] *o cc ltyp,p* [*vlines,vls,vrpt*] .
All	@SRR,*c,d,r o cc ltyp,p* .
All	@SRU,*c,d*[,*r,l,q,lab*] *o cc ltyp,p* [*vlines,vls,vrpt*] .
All	@STN[,*sn,lab vuser,vdepn,vdept,vttyp,vvsiz,vhsiz,vaspect, vcolor,vgraph*] .
1100	@STR,*c,d,r*[,*runid,acct*] .
All except 1100	@STR,*c,d,r*[,*sl,ifc,logon,pw*] .
All	@SUB,*c,d,r*[,*lab*] *o cc ltyp,p vrslts* . (Place next line in output area.)
1100	@TCS,*c,d*[,*r,f*],*dev,code*[,*unit,tach,transp?,tabs?,trk,eot?*,EOD, *dhdgs?,lsp,sn*] .
All	@TOT,*c,d,r*[,*lab*] *o cc ltyp,p* [*vrslts*] .
1100, U Series, UNIX	@TRC[,*c,d,r,db*] .
All	@ULK .
U Series	@UNX[,*c,d,logon,pw,lab*] *o cmd* [*args*] .
All	@UPD .
All	@USE *name=v*[,*name=v*,...,*name=v*] .
1100	@WAT *ms* .
All except 1100	@WAT[,M,*lab*] *ms* .

continued

continued

Systems	Run Statement
All except BTOS and PC MAPPER	@WDC,*ic,id,ir,rc,rd,rr*[,*l,col,lab*] .
All except BTOS and PC MAPPER	@WDL,*ic,id,ir,rc,rd,rr*[,*l,col,lab*] *vlno,vcol,vcolnxt* .
All except BTOS and PC MAPPER	@WPR,*c,d,r*[,*l,lab*] *wpcmd* .
1100	@WRL,*c,d,r,l*[,*ntuid?,wpw,ccpy?*] *cc ltyp,vld* .
All except 1100	@WRL,*c,d,r,l*[,*ntuid?,wpw*] *cc ltyp,vld* .
All	@XCH[,*startv,q,level*] .
All	@XIT .
All	@XQT{,*lab* | *vld*} .
All except 1100 and PC MAPPER	@XUN .
All except 1100 and PC MAPPER	@**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
access	Access type (PUB, PRI)
acct	Batch run account number
ack?	Request acknowledgment
adj	Adjust the size of decimal fields
aga?	Assume graphics active, Y/N
all?	All reports in drawer, Y/N
angle	Rotation angle
args	Arguments
b	Buffer label
banner	Banner/printout identification
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
ccpy?	Write a complete copy of the report, Y/N
ci	Cabinet to send as input
cmd	Command
co	Cabinet in which to replace the result
code	Mnemonic function code for cassette/diskette operation
col	Column number
cqty	COMMIT command frequency
cs	Character set
cstat	Alternate status code
cyc	File cycle
cys	Number of copies
d	Drawer
d1	Drawer to compare
d2	Drawer to compare with the first drawer
date	Date
db	Database name
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
dnfunc	Destination function

dnqu	Destination queue
dnrun	Destination MAPPER run or TIP transaction
do	Drawer in which to replace the result
d/p	Eliminate period lines containing descriptions of field abbreviations (*d*) or include descriptions (*p*).
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
dst	Specify whether duplicate data items are to be eliminated
dstn	Destination print model
dup/col	Duplicate primary keys in result
d1char?	Delete first character, Y/N
edsp?	Generate error display, Y/N
elt	Standard OS 1100 element name
endate	Ending date
endrpt	Ending report number
eol	End-of-line character
eot?	End of type, Y/N
eq	Equations
erase?	Erase screen, Y/N
err	Error status code
exp	Arithmetic expressions
expand?	Handle expanded syntax, Y/N
ext	File name extension
f	Format of report
field	Report field
fldtxt	Field text
fn	File name
formsid	Predefined or special forms identification
Fs	Number of blank fields between columns
func	DTM function: INFO or STAT
fxcz?	Fixed character size, Y/N
fxmt?	Forced transmit, Y/N
gran	Granularity
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
host	Host configured in DDP 1100
ic	Issuing cabinet
icc	Issuing report column-character positions
id	Issuing drawer
id col tbl	Cursor id, column names, table names
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
info	Specify additional information about the tables to be included
init	Initial file allocation size
interim?	Interim display, Y/N
ip	Issuing report parameters
iqty	Number of rows to insert
ir	Issuing report
istat	Alternate status code
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
lang	Alternate language identification code for RDMS only.
lb4	Line number before (start after this line)
lc	Local cabinet
lcpw	Local cabinet password
ldr	Local drawer
le	Local element
level	Previously saved level
-level	Level of encryption on encoded reports
lf	Local format
lfn	Local file name
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
lq	Local qualifier
lr	Local report
ls	Local site number
lsp	Line spacing
ltyp	Line type
ltyp1	Line type to compare in the first report
ltyp2	Line type to compare in the second report
lv	Local version
mapperf?	MAPPER format, Y/N
maxfs	Maximum file size
maxy	Maximum Y value
min-max	Range of numbers
ms	Milliseconds
msg	Message to display on the control line
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
n-char	Control treatment of null values
net-id	Network identifier of remote system
newcyc?	New file cycle, Y/N
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)
options	Optional information for creating tables
optmz?	Optimize result, Y/N
oq	Originating queue name
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 (number of lines)
plotter?	Add or delete plotter primitives, Y/N
pn	Form page number of text to display
pos	Position for copying a file using DDP 1100
predfl	Predefined line reference number
proglab	Program label
protect	Protected format
prtsite	Print site
pthfn	Full path and file name
pw	Password
q	Quantity
qual	Qualifier
r	Report number
r1	Report to compare
r2	Report to compare with first report
rb	Issue a ROLLBACK command, Y/N
rc	Receiving cabinet
rcc	Receiving report column-character positions
rd	Receiving drawer
rel?	Release control to run, Y/N
replstr	Replacement string
rhost	Receiving DDP 1100 host
ri	Report to send as input
rltyp	Receiving report line type
rmc	Remote cabinet
rmd	Remote department number

rmdr	Remote drawer
rme	Remote element
rmf	Remote format
rmfn	Remote file name
rmpw	Remote user password
rmq	Remote qualifier
rmr	Remote report number
rms	Remote site letter (or number)
rmt?	Program uses remote file, Y/N
rmu	Remote user-id
rmv	Remote version
rp	Receiving report parameters
rr	Receiving report
rspfunc	Response function
rspqu	Queue to receive message
rsrun	Name of run to call
rst	Run statement to queue
run	Run name
run-id	Run-id (batch)
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
skip	Number of data rows in the table to skip for the result
sl	Site letter
sn	Station number
sort	Specify whether primary key columns are to be displayed first
sp	Alternate character to designate the space character
spcc	Control character used to specify printer device controls
srpt	Starting report number
startv	Starting variable number
stmt	Another run statement
ststr	Starting string
substrv	Variable substring
syntax	SQL syntax
table	Name of table
tabs?	Tab characters, Y/N
tabp	Tab position
tach	Starting tachometer address
testv	Test variable
tgtstr	Target string
tics?	Enclose field name in apostrophes, Y/N

time	Time
title	Title (up to 12 characters)
tmo	Timeout (number of seconds to wait for response)
transl	Translate data (TRA, ASC, EBC)
transp?	Transparent format, Y/N
trfadr	Transfer address
trk	Track number
trnrpt	Report number of translation table
type	Type of file
ulvl?	Upper-level chart, Y/N
unit	Unit name/-id
unp?	Unpack result, Y/N
user	User-id
v	Variable name
val	Value
vaspect	Variable with aspect ratio of the specified station
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
vcolnxt	Variable with next column number
vcolor	Variable with graphics color flag or terminal
vcom	Variable with actual number of COMMIT commands issued
vcpl	Variable with number of characters per line
vcs	Variable with character set
vdata	Variable with data
vdbt	Variable with the database type
vdepn	Variable with department name
vdept	Variable with department number
vdpk	Variable with number of duplicate primary key columns
vdrawer	Variable with drawer letter
ver	OS 1100 element version name
vert	Vertical display of the result
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)
vgraph	Variable with graphics type
vins	Variable with number of rows inserted or updated
vld	Variables, literal data, constants, or a combination (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
vmfn	Variable with internal MAPER file name (for example, M00001)
vmfno	Variable with MAPER file number (for example, 1)
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 message
vnbq	Variable with number of columns inserted or updated (or the number of key columns for deleting rows)
vnxrd	Variable with next available report in the drawer
voccs	Variable with number of occurrences
vol	Volume-id for removable/system disk pack
vpos	Variable with character position
vqty	Variable with the number of columns selected
vretn	Variables with information to pass to the database manager (up to 18 variables)
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
vstat	Variable with error status
vtitle	Variable with title of the new report
vttyp	Variable with terminal type code
vupds	Variable with number of updates
vuser	Variable with user-id
vvsiz	Variable with vertical screen size (rows per screen)
vwpw	Variable with write password
WHERE clause	Clause specifying one or more conditions for data retrieval
wpcmd	Word processing command or " for interactive word processing
wpw	Write password
wrap	Control wrapping of columns and data lines
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 statements.

BFN A, B, C[2], E, F*n*, 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, *

1100: S(*x*) instead of S(s)

DAT A, T, W, *n*

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]

1100: Also X+ and X-

SRH A, B[(*n*)], C[2], D, F, H, L(*x*), N, P, Q[(*n*)], R*x*{-*y*|,*y*}, T[(*x*)], U[(*x*)], @, /

1100: Also E*n*, S

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.

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.

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]	Case sensitivity (character set control on 1100)	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 occurrence of item found. item appears more than once.	BFN
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
1100: E*n*	Estimate 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...*	Specify order of sorted fields	BFN

continued

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 Rx and R*x-y* are allowed.

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

Option	Purpose	Functions	
H	Display heading lines from only the first report in multiple report search.	SRH	
H	Cumulate horizontally.	TOT	
I[*n*]	Use index in report *n*. Default = report 2.	BFN	
I[1]	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	
N	Create separate line per item and item count in result.	BFN	
N	Omit grand total.	TOT	
N[*n*]	Substitute numeric value *n* for nonnumeric fields. Default = 0.	CAL	
N	Display lines not meeting match or search parameters.	MCH	SRH

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.

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

Option	Purpose	Functions		
O	Create result containing items found.	BFN	LCH	LOC
O	Omit data lines from result; include headings, value labels or totals.	CAL	TOT	
OU	Create update result.	LCH	LOC	
P	Include period lines in result (valid only with N option).	BFN		
P	Issuing and receiving reports are presorted.	MCH		
P	Process paragraphs.	SRH		
Q	Find a single item quickly.	BFN		
Q	Specifies reports are sorted and will not be verified.	MCH		
Q[(*n*)]	Stop scan after *n*th paragraph. Default = first paragraph; use with the P option.	SRH		
R*x*{-*y*|,*y*}[4]	Scan range of reports: reports *x* through *y*; scan reports *x*,*y*.	BFN	FND	SRH
R*n*	Round answers to nearest *n*.	CAL	TOT	
S	Scan each report separately.	BFN		
S*x*{-*y*|,*n*}	Start scan at line *x* through line *y*; scan *n* lines.	LCH	LOC	
S	Display matched lines in issuing report order.	MCH		
1100: S	Display found lines in search parameter order.	SRH		
S	Place subtotals in vertical operation fields.	TOT		
S(s)	Case sensitivity.	CAL		
1100: S(*x*)	Define character set interpretation. *x* = f, l, or s.	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.

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

Option	Purpose	Functions		
T	Include processed and unprocessed lines in result.	CAL		
T*x*	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	Converts dates into day of the week.	DAT		
X	Exclude invalid values in MIN, MAX, SUM, AVG, VMIN, VMAX, VSUM, and VAVG computations.	CAL		
1100: X+	Use OS 1100 Sort/Merge regardless of report length.	SOR		
1100: X-	Use normal MAPPER software sorting process regardless of report length.	SOR		
n	Specify *n* workdays in week. Default = 7.	DAT		
@	Find or search for blank characters (spaces).	BFN	FND	SRH
/	Find or 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.

Appendix B
Reserved Words

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

Use reserved words to retrieve information maintained by the system. For example, use the reserved word DATE1$ to retrieve the current date in the format YYMMDD. You can use reserved words to specify values for fields and subfields in run statements.

Table B-1 lists the available MAPPER reserved words. Some reserved words contain meaningful data only after execution of a specific run statement. For example, XERR$ contains meaningful data only after the run executes an RER statement. For these cases, the run statements that set the value of the reserved word are shown in parentheses after the description of the reserved word. (Note that some reserved words are included for compatibility with other kinds of MAPPER systems.)

For examples and information on using a reserved word that pertains to a specific run statement, refer to the run statement that sets its value. See also "Using Reserved Words in Run Statements" in Section 2.

Reserved Words in the Output Area

The system inserts the value of any reserved words placed in the output area, not the reserved word itself.

1100: On OS 1100 MAPPER Systems, the system always inserts the literal representation of reserved words (the reserved word itself) in the output area. To place the value of reserved words in the output area, load variables with the reserved words and place the variables in the output area.

Commonly Used Variable Types and Sizes of Reserved Words

The list of reserved words in table B-1 includes examples of commonly used variable types and sizes (enclosed in brackets after the description). Wherever necessary, the size indicates the most desirable variable size to use in runs that will be executed on more than one kind of MAPPER system, such as OS 1100 and U Series systems.

Example

```
DRW$  Drawer number of the current result (-0).  [I6]
```

[I6] indicates integer type, six characters long.

Following are the variable types used in the list of reserved words:

A	Alphanumeric
F	Fixed-point decimal
H	Hollerith
I	Integer
S	String

Use the examples not as hard and fast rules, but only as a quick guide. We do not indicate variable types and sizes for all of the reserved words.

Table B-1. Reserved Words

Word	Content
ACDRW$	Alphabetic drawer of the calling run. [H1]
ACTYPE$	Alphabetic drawer (form type) of the calling run. (Same as ACDRW$.) [H1]
ADRW$	Alphabetic drawer of the current result. [H1]
ADRW1$ through ADRW8$	Alphabetic drawer of the -1 through -8 results. [H1]
AEDRW$	Alphabetic drawer of the run control report. [H1]
AETYPE$	Alphabetic drawer (form type) of the run control report. (Same as AEDRW$.) [H1]
AKEY$	Key or key sequence used to perform the Abort function.
AREA$	Named area. [H12]
ASPECT$	Aspect ratio (used with graphics). [A12]
	PC MAPPER: The ASPECT$ reserved word provides a constant but is ignored.
ATYPE$	Alphabetic drawer (form type) of current result. (Same as ADRW$) [H1]
ATYPE1$ through ATYPE8$	Alphabetic drawer (form type) of the -1 through -8 results. (Same as ADRW1$ through ADRW8$.) [H1]
AXDRW$	Alphabetic drawer of the erring run. [H1]
AXTYPE$	Alphabetic drawer (form type) of the erring run. (Same as AXDRW$.) [H1]
BEL$	Code for the terminal bell.
	1100: The BEL$ reserved word is not available on OS 1100 MAPPER Systems.
1100: BPORT$	Run started in the batch port (if nonzero, yes; if 0, no). [I1]
CAB$	Cabinet number of the report or result last processed, or cabinet number of the report or result on display when the run started, or 0 if no report or result was processed or on display. [I4]
CAB1$	Currently active cabinet number. [I4]
CDRW$	Drawer number of the calling run control report from an external subroutine (if the report is from an internal subroutine, CDRW$ = 0). [I6]
CERR$	Message number of the error. Same as XERR$, but clears the error. [I6]
CHAR$	Number of characters per line in the drawer. [I3]

continued

Table B-1. Reserved Words (cont.)

Word	Content
COLOR$	Color terminal flag (if nonzero, terminal has color display). [I1] **PC MAPPER:** The COLOR$ reserved word always returns nonzero, even if the monitor is monochrome VGA graphics.
CPRIV$	User has coordinator privileges (if nonzero, yes; if 0, no). [I1]
CRID$	Report number of the calling report from an external subroutine (if the report is from an internal subroutine, CRID$ = 0). (Same as CRPT$.) [I4]
CRPT$	Report number of the calling report from an external subroutine (if the report is from an internal subroutine, CRPT$ = 0). [I4]
CTYPE$	Drawer (form type) number of the calling run control report from an external subroutine (if the report is from an internal subroutine, CTYPE$ = 0). (Same as CDRW$.) [I6]
CUR$	Cursor position (CUR$ *scl,col*). [I2] **1100:** The CUR$ reserved word is not available on OS 1100 MAPPER Systems.
CURH$	Horizontal character position of the cursor. [I3]
CURV$	Vertical character position of the cursor. [I2]
DATE0$	Current date in YMMDD format. [A5]
DATE1$	Current date in YYMMDD format. [A6]
DATE2$	Current date in DD MMM YY format. [A9]
DATE3$	Current date in YDAY format. [A4]
DATE4$	Current date in YYDAY format. [A5]
DATE5$	Current date in DDMMYY format. [A6]
DATE6$	Current date in MM/DD/YY format. [H8]
DATE7$	Current date in MONTH DD, YYYY format. [H18]
DATE8$	Current date in MMDDYY format. [A6]
DAY$	Current day in DAY format (DAY=MON, and so on). [H3]
DEPN$	User's department sign-on number. [I4]
DEPT$	User's department name. [H12]
DLINE$	Line number of the first nonheld line on display. [I6]
DLP$	Number of data lines processed during the run (includes LLPs and all lines processed; compare to IO$ and LLP$). [I12]
DRW$	Drawer number of the current result (-0). [I6]
DRW1$ through DRW8$	Drawer number of the -1 through -8 results. [I6]

Table B-1. Reserved Words (cont.)

Word	Content
1100: DTM$	Status of Data Transfer Module: [I1] 0 = DTM is not configured. 1 = DTM is configured, but the MAPPER system is not attached. 2 = DTM is configured, and the MAPPER system is attached. 3 = Requesting run is processing DTM input.
1100: DTNAM$	Name of this MAPPER system as known to DTM.
ECAB$	Cabinet number of the run control report. [I4]
EDRW$	Drawer number of the run control report. [I6]
ELINE$	Current execution line in the run control report. [I6]
EMODE$	Mode number of the run control report. (Same as ECAB$.) [I4]
ERID$	Report number of the run control report. (Same as ERPT$.) [I4]
ERL$	Erase to the end of the line or to the end of the next protected field. **1100:** The ERL$ reserved word is not available on OS 1100 MAPPER Systems.
ERPT$	Report number of the run control report. [I4]
ERS$	Erase to the end of the screen. **1100:** The ERS$ reserved word is not available on OS 1100 MAPPER Systems.
ESC$	Escape sequence. **1100:** The ESC$ reserved word is not available on OS 1100 MAPPER Systems.
ETYPE$	Drawer (form type) number of the run control report. (Same as EDRW$.) [I6]
F1$ through F10$	Key or key sequence used as the **F1** through **F10** function keys.
FCAB$	Cabinet number of the Screen Control form on display. [I4]
1100: FCC$	UTS 400 station has FCC (hardware protect capability) (if nonzero, yes; if 0, no). [I1]
FDRW$	Drawer number of the Screen Control form on display. [I6]
FFTYPE$	Drawer (form type) number of freeform drawer A. [I6]
FIELD$	Relative input field number in which the cursor is located. [I4]
FKEY$	Number of the function key that the run user pressed.
FMT$	Format of the report on display (the run must be registered as format sensitive). [I2]

Table B-1. Reserved Words (cont.)

Word	Content
FPAGE$	Last explicitly called page of a Screen Control form.
FRPT$	Report number of the Screen Control form on display. [I4]
GRAPH$	Graphics terminal flag (if 0, terminal cannot display graphics). [I1]
	PC MAPPER: The GRAPH$ reserved word always returns nonzero, signifying a UTS 60 graphics terminal.
HLINES$	Number of held lines.
ICVAR$	Input data entered on the control line (no leading tabs required; strings allowed) (CHD). [S80]
INMSV$	Input data from the function mask on the screen (OUM). [S80]
INPUT$	Input data from the screen or from an external source (up to 40 variables) (OUT, RRN, RUN, BR, SC).
INSTR$	Input from variables on screen lines (no leading tabs required; strings allowed) (OUT or SC). [S80]
INVAR$	Input from input fields on the screen, delimited by tabs (up to 40 variables and strings allowed) (OUT or SC).
INVR1$	Input from input fields on the screen (up to 40 variables and strings allowed) (OUT or SC).
IO$	Number of storage I/O requests processed during the run. [I12]
KKEY$	Key or key sequence used to obtain keyboard help.
LANG$	User's currently selected language number. [I1]
1100: LCAB$	Cabinet number of user's currently selected language. [I4]
LINE$	Line number of the next line to read (RDL, RLN). [I6]
LINK$	Run started from another another run via a LNK statement (if nonzero, yes; if 0, no). [I1]
LLP$	Number of logic lines processed during the run. [I12]
LNKDRW$	Drawer of the run that issued the LNK statement. [I6]
LNKRPT$	Report number that issued the LNK statement. [I4]
LRRSD$	Local site identifier. [I6]
MAPER$	Current MAPPER software level. [A12]
MAXCAB$	Maximum cabinet number available on your MAPPER system. [I4]

Table B-1. Reserved Words (cont.)

Word	Content
MAXDRW$	Maximum drawer number available on your MAPPER system. [I6]
1100: MAXLAB$	Maximum number of labels allowed in your run. [I3]
1100: MAXRPT$	Maximum report number available on your MAPPER system. [I4]
1100: MAXVAR$	Maximum number of variables allowed in your run. [I3]
MKEY$	Key or key sequence used to perform the Message Waiting function.
MODE$	Mode number of the report or result last processed, or mode number of the report or result on display when the run started, or 0 if no report or result was processed or on display. (Same as CAB$.) [I4]
MODE1$	Currently active mode number. (Same as CAB1$.) [I4]
MXRPT$	Maximum report number available on your MAPPER system. [I4]
1100: MSEC$	Current number of milliseconds since midnight. [I12]
OLINE$	Next line to write in the output area. [I6]
ORSTAN$	Originating station number of a background run. [I5]
RID$	The number of the report or result last processed, or of the report or result on display when the run started (if RID$ = -0, current result; if RID$ = 0, no report). After an ADR or DUP statement, RID$ contains the report number of the new report. (Same as RPT$.) [I4]
RPT$	The number of the report or result last processed, or of the report or result on display when the run started (if RPT$ = -0, current result; if RPT$ = 0, no report). After an ADR or DUP statement, RPT$ contains the report number of new report. [I4]
RRSID$	Remote run site-id. [I6]
RSLANT$	Reverse slant character (\). [H1]
RUN$	Name of the run executing. [H12]
SCNH$	Horizontal screen size of the user's terminal. [I3]
SCNV$	Vertical screen size of the user's terminal. [I2]
SOE$	Start-of-entry (SOE) character. [H1]
SOEH$	Horizontal cursor position of the SOE character. [I3]
SOEV$	Vertical cursor position of the SOE character. [I2]
STACK$	Current level number of saved variables. [I2]
STAT1$	Status word 1 (quantity or status) (BFN, CAL, CAU, IND, LLN, LOK, LZR, MCH, RRN).
STAT2$	Status word 2 (quantity or status) (BFN, CAL, CAU, IND, LLN, LZR, MCH).

Table B-1. Reserved Words (cont.)

Word	Content
STAT3$	Status word 3 to load save date on LZR.
STNUM$	Station number executing the run. [I5]
SYSNAM$	Port identification name (5 characters). [H5]
TIC$	Apostrophe character. [H1]
TIME$	Current time in HH:MM:SS format. [H8]
TTYPE$	Terminal type of station executing the run. **U Series and UNIX:** The terminal type is read from configuration report or MAPTERM environment variable. [H8] **A Series:** The terminal type is read from configuration report. **PC MAPPER:** The TTYPE$ reserved word always returns spaces.
TYPE$	Drawer (form type) number of current result (-0). (Same as DRW$.) [I6]
TYPE1$ through TYPE8$	Drawer (form type) number of the -1 through -8 results. (Same as DRW1$ through DRW8$.) [I6]
USER$	User-id of the user who started the run. [H11]
XDRW$	Drawer number of the run control report where the run aborted or erred (RAR, RER). [I6]
XERR$	Message number of error encountered by error routine (RER). [I6]
XFUN$	Last function call before run aborted or erred (RAR, RER). [H6]
XKEY$	Key or key sequence used to transmit.
XLINE$	Line number in the run control report where the run aborted or erred (RAR, RER). [I6]
XRID$	Report number of the run control report where the run aborted or erred (RAR, RER). (Same as XRPT$.) [I4]
XRPT$	Report number of the run control report where the run aborted or erred (RAR, RER). [I4]
XTYPE$	Numeric form type of the run control report where the run aborted or erred (RAR, RER). (Same as XDRW$.) [I6]

Appendix C
Control Commands for Transferring Data

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

This appendix lists the data control commands available that let you control the format of the data in the files being transferred. Use these commands with the Create File (FIL) and Start (STR) statements when you do not want the files in MAPPER format.

Data Control Commands

Enter these commands, beginning in column 1, on any line in the report. They take effect from that point on but the line containing the control command is removed from the output. Except for $INCL$, $TABA$ and $TRNA$, the format is just the command itself.

Table C-1. Data Control Commands

Command	Description
$CLRT$	Clears character and tab code translation initialized by the $TRNA$ and $TABA$ commands; translates tab characters to spaces (default).
$DATA$	Suspends any data translation by the $TRNA$ and $TABA$ commands; uses the original data (does not affect $TABA$ translations). See the following chart to determine how tab characters are affected:

Data control command:	Tabs after translation:	Tabs after $DATA$ command:
no command	space	tab
$TABA$ 'x'	x	x
$CLRT$	space	tab

(x = any character)

$DCML$	Deletes all asterisk type lines except heading lines.
$DFFL$	Deletes all period type lines except heading lines.
$ICML$	Includes all asterisk type lines.
$IFFL$	Includes all period type lines.

continued

Table C-1. Data Control Commands (cont.)

Command	Description
$INCL$	Includes reports in the runstream from numeric drawer ***nnnn***, reports ***n,n*** and ***n-n*** (any combination in any order). Format: **$INCL$ [H] D*nnnn* R{*n,n,n-n*}** or **$INCL$ [H] D*nnnn* A** where: **H** is a control character to pass report headings. ***nnnn*** is the numeric drawer designator. ***n*** is a report number. Separate individual report numbers with commas; specify a range of reports with a hyphen. **A** is a control character to include all reports in a drawer. For example, **$incl$ h d2 r3-6,23,40-57** includes reports, with headings, from numeric drawer 2, reports 3 through 6, 23, and 40 through 57.
$TABA$	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. (See Appendix D for ASCII codes.) ***y*** is the character (enclosed in apostrophes). For example, **$taba$ '&'** translates tab characters to ampersands.
1100: $TABC$ *nn*	Translates tab characters to the character represented by Fieldata code *nn*. Format: **$TABC$** *nn* where: *nn* is the Fieldata code of the character. (See Appendix D for Fieldata codes.) For example, **$tabc$ 50** translates tab characters to asterisks.

Table C-1. Data Control Commands (cont.)

Command	Description
1100: $TRAN$	Translates character *x* to the character represented by Fieldata code *nn*, or reestablishes the translation previously suspended by $DATA$. Format: **$TRAN$** *x,nn* or **$TRAN$.** where: *nn* is the Fieldata code of the character. (See Appendix D for Fieldata codes.) *x,nn* translates character *x* to the character *nn* in the Fieldata character set. (See Appendix D for Fieldata codes.) . is space-period-space (or may be blank to the end of the line) and reestablishes the translation previously suspended by the $DATA$ command. For example, **$tran$ &,05 $,03** translates ampersands (&) to spaces (Fieldata code 05) and dollar signs ($) to pound signs (#), (Fieldata code 03).
$TRNA$	Translates character *x* to the character represented by ASCII octal code *nnn* or to the character *y* (any combination in any order), or reestablishes the translation previously suspended by the $DATA$ command. Format: **$TRNA$** {*x,nnn*|*x,'y'...*} or **$TRNA$.** where: *x,nnn* translates character *x* to the character *nnn* in the ASCII character set. *x,'y'* translates character *x* to the literal **'y'**. . is space-period-space (or may be blank to the end of the line) and reestablishes the translation previously suspended by the $DATA$ command. For example, **$trna$ &,040 $,'?'** translates ampersands (&) to spaces (ASCII octal code 040) and dollar signs ($) to question marks (?).

Appendix D
Character Sets and Sorting Orders

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

This appendix contains the following topics:

- ASCII character set
- Fieldata (limited character set)
- Using the C(S) option
- Character set processing on the OS 1100 MAPPER System

ASCII Character Set

Table D-1. ASCII Character Set

Character	ASCII Internal Octal Code	ASCII Decimal Code	Hexadecimal Code
Tab	11	9	9
Space	40	32	20
!	41	33	21
"	42	34	22
#	43	35	23
$	44	36	24
%	45	37	25
&	46	38	26
'	47	39	27
(	50	40	28
)	51	41	29
*	52	42	2A
+	53	43	2B
,	54	44	2C
-	55	45	2D
.	56	46	2E
/	57	47	2F
0	60	48	30
1	61	49	31
2	62	50	32
3	63	51	33
4	64	52	34
5	65	53	35
6	66	54	36
7	67	55	37
8	70	56	38
9	71	57	39
:	72	58	3A
;	73	59	3B
<	74	60	3C
=	75	61	3D
>	76	62	3E
?	77	63	3F
@	100	64	40
A	101	65	41

continued

Table D-1. ASCII Character Set (cont.)

Character	ASCII Internal Octal Code	ASCII Decimal Code	Hexadecimal Code
B	102	66	42
C	103	67	43
D	104	68	44
E	105	69	45
F	106	70	46
G	107	71	47
H	110	72	48
I	111	73	49
J	112	74	4A
K	113	75	4B
L	114	76	4C
M	115	77	4D
N	116	78	4E
O	117	79	4F
P	120	80	50
Q	121	81	51
R	122	82	52
S	123	83	53
T	124	84	54
U	125	85	55
V	126	86	56
W	127	87	57
X	130	88	58
Y	131	89	59
Z	132	90	5A
[	133	91	5B
\	134	92	5C
]	135	93	5D
^	136	94	5E
_	137	95	5F
` (grave accent)	140	96	60
a	141	97	61
b	142	98	62
c	143	99	63

continued

Table D-1. ASCII Character Set (cont.)

Character	ASCII Internal Octal Code	ASCII Decimal Code	Hexadecimal Code
d	144	100	64
e	145	101	65
f	146	102	66
g	147	103	67
h	150	104	68
i	151	105	69
j	152	106	6A
k	153	107	6B
l	154	108	6C
m	155	109	6D
n	156	110	6E
o	157	111	6F
p	160	112	70
q	161	113	71
r	162	114	72
s	163	115	73
t	164	116	74
u	165	117	75
v	166	118	76
w	167	119	77
x	170	120	78
y	171	121	79
z	172	122	7A
{	173	123	7B
|	174	124	7C
}	175	125	7D
~	176	126	7E

Fieldata (Limited Character Set)

This section on the limited character set applies only to the OS 1100 MAPPER System.

Table D-2. Limited Character Set

Character	Fieldata Internal Octal Code
@	00
space	05
A	06
B	07
C	10
D	11
E	12
F	13
G	14
H	15
I	16
J	17
K	20
L	21
M	22
N	23
O	24
P	25
Q	26
R	27
S	30
T	31
U	32
V	33
W	34
X	35
Y	36
Z	37
)	40
-	41
+	42
<	43
=	44
>	45
&	46

continued

Table D-2. Limited Character Set (cont.)

Character	Fieldata Internal Octal Code
$	47
*	50
(	51
%	52
:	53
?	54
!	55
, (comma)	56
\	57
0	60
1	61
2	62
3	63
4	64
5	65
6	66
7	67
8	70
9	71
' (apostrophe)	72
;	73
/	74
. (period)	75

Using the C(S) Option

1100: For the OS 1100 MAPPER System, the following information applies only to FCS reports.

Some functions allow you to specify case sensitivity by using the C(S) option. When you use a function without the C(S) option, the system treats lowercase and uppercase characters the same. For example, when you sort a report, the letters **a** and **A** appear together in sort order.

When you use the C(S) option, the report is sorted in the strict order of the ASCII codes: uppercase with uppercase and lowercase with lowercase.

These differences also affect a range search. For example, a range search from **a** to **z** with no options produces **a** through **z** and **A** through **Z**. In the same range search with the C(S) option, the uppercase letters **A** through **Z** are not included in the result.

Table D-3 shows the sort order of the characters with the C(S) option. Table D-4 shows the order without the C(S) option. Both tables start at the top of the leftmost column and go down, then from the top of the second column and so on.

1100: Tables D-3 and D-4 do not apply to the OS 1100 MAPPER System; the sort order for LCS is shown in Table D-5 and the sort order for FCS is shown in Table D-6.

Table D-3. Sorting with the C(S) Option

Tab	-	;	I	W	d	r
Space	.	<	J	X	e	s
!	/	=	K	Y	f	t
"	0	>	L	Z	g	u
#	1	?	M	[	h	v
$	2	@	N	\	i	w
%	3	A	O	]	j	x
&	4	B	P	^	k	y
'	5	C	Q		l	z
(	6	D	R	` (grave	m	{
)	7	E	S	accent)	n	|
*	8	F	T	a	o	}
+	9	G	U	b	p	~
,	:	H	V	c	q	

Table D-4. Sorting without the C(S) Option

Tab	-	;	e	l	s	z
Space	.	<	E	L	S	Z
!	/	=	f	m	t	[
"	0	>	F	M	T	\
#	1	?	g	n	u	]
$	2	@	G	N	U	^
%	3	a	h	o	v	
&	4	A	H	O	V	` (grave
'	5	b	i	p	w	accent)
(	6	B	I	P	W	{
\	7	c	j	q	x	|
*	8	C	J	Q	X	}
+	9	d	k	r	y	~
,	:	D	K	R	Y	

Character Set Processing on the OS 1100 MAPPER System

This section applies only to the OS 1100 MAPPER System.

The OS 1100 MAPPER System processes Fieldata and ASCII character sets. You cannot use certain characters as data (namely, those characters used as control characters in report processing).

LCS, FCS, and FCSU

The Fieldata character set used in MAPPER software is called limited character set (LCS); the ASCII character set is called full character set (FCS). Also, you can store and process FCS report data in uppercase alphabetic characters only — called full character set upper (FCSU).

The character set of a drawer is determined when the drawer is generated. To see which character set a report is in, use the Line Zero (LZ) function.

Character Hierarchy

The hierarchy (that is, the relationship of characters to one another) is different between the LCS and FCS character sets. Since MAPPER software actually processes the data as represented by the Fieldata and ASCII codes, the order of the Fieldata or ASCII codes determines the character hierarchy.

For example, in LCS, numeric characters have higher values than alphabetic characters. Therefore, in sort or search-in-range processes, alphabetic characters come before numeric characters. On the other hand, in the FCS character set, alphabetic characters have higher values than numeric characters.

Consider these intercharacter relationships carefully, especially when processing drawers that use different character sets.

Using the C(x) Option

Under normal report or result processing, the functions assume that you want to process the characters according to their native code order.

With some functions, you can use the C(*x*) option to process data with intercharacter relationships different from their native code order. You can use one of the following:

C(F)	Process using full character set (ASCII) order
C(L)	Process using limited character set (Fieldata) order
C(S)	Process using the strict character order, that is, treat uppercase and lowercase characters differently

Using the C(L) and C(F) Options

When using a function that processes more than one report, and the reports are in different character sets, you may need to use the C(F) or C(L) option to indicate which character set order to use for the function. This ensures that the reports use the same hierarchy.

For example, suppose you want to match two reports in different character sets. You plan to sort the reports first and use the P option of the Match function, which specifies that the reports are presorted.

If the receiving report is in LCS and the issuing report is in FCS, use the C(L) option when sorting the issuing report.

Conversely, if the issuing report is in LCS and the receiving report is in FCS, use the C(F) option when sorting the issuing report.

Table D-5 shows the LCS order when using the C(*x*) option. Table D-6 shows the FCS order when using the C(*x*) option.

Table D-5. LCS Order Using the C(x) Option

Basic or Sorted C(L) or C(S)	Sorted FCS C(S)	Sorted FCS C(F)
@	@	Tab
Tab	Tab	Space
Space	Space	!
A	A	$
B	B	%
C	C	&
D	D	'
E	E	(
F	F	)
G	G	*
H	H	+
I	I	,
J	J	-
K	K	.
L	L	/
M	M	0
N	N	1
O	O	2
P	P	3
Q	Q	4
R	R	5
S	S	6
T	T	7
U	U	8
V	V	9
W	W	:
X	X	;
Y	Y	<
Z	Z	=
)	)	>
-	-	?
+	+	@
<	<	A
=	=	B
>	>	C

continued

Table D-5. LCS Order Using the C(x) Option (cont.)

Basic or Sorted C(L) or C(S)	Sorted FCS C(S)	Sorted FCS C(F)
&	&	D
$	$	E
*	*	F
(	(	G
%	%	H
:	:	I
?	?	J
!	!	K
,	,	L
\	\	M
0	0	N
1	1	O
2	2	P
3	3	Q
4	4	R
5	5	S
6	6	T
7	7	U
8	8	V
9	9	W
'	'	X
;	;	Y
/	/	Z
.	.	\

Table D-6. FCS Order Using the C(x) Option

Basic or Sorted C(F)	Sorted Sorted C(S)	C(L)
Tab	Tab	@
Space	Space	Tab
!	!	[
"	"	]
#	#	#
$	$	Space
%	%	A,a
&	&	B,b
'	'	C,c
(	(	D,d
)	)	E,e
*	*	F,f
+	+	G,g
,	,	H,h
-	-	I,i
.	.	J,j
/	/	K,k
0	0	L,l
1	1	M,m
2	2	N,n
3	3	O,o
4	4	P,p
5	5	Q,q
6	6	R,r
7	7	S,s
8	8	T,t
9	9	U,u
:	:	V,v
;	;	W,w
<	<	X,x
=	=	Y,y
>	>	Z,z
?	?	)
@	@	-
A,a	A	+

continued

Table D-6. FCS Order Using the C(x) Option (cont.)

Basic or Sorted C(F)	Sorted Sorted C(S)	C(L)
B,b	B	<
C,c	C	=
D,d	D	>
E,e	E	&
F,f	F	$
G,g	G	*
H,h	H	(
I,i	I	%
J,j	J	:
K,k	K	?
L,l	L	!
M,m	M	,
N,n	N	\
O,o	O	0
P,p	P	1
Q,q	Q	2
R,r	R	3
S,s	S	4
T,t	T	5
U,u	U	6
V,v	V	7
W,w	W	8
X,x	X	9
Y,y	Y	'
Z,z	Z	;
[	[	\
\	\	"
]	"	¨
—	-	—
	a	
{	b	{
|	c	|
}	d	}
~	e	~
	f	

continued

Table D-6. FCS Order Using the C(x) Option (cont.)

Basic or Sorted C(F)	Sorted Sorted C(S)	C(L)
	g	
	h	
	i	
	j	
	k	
	l	
	m	
	n	
	o	
	p	
	q	
	r	
	s	
	t	
	u	
	v	
	w	
	x	
	y	
	z	
	{	
	|	
	}	
	~	

Appendix E
Using Your Keyboard and Mouse

This appendix describes keys available on the function key bar and the use of the MAPPER software mouse.

Function Keys

The keys referred to in Table E-1 correspond to the keys available on the function key bar. Since your needs change depending on your task at the time, the keys available on the function key bar change to fit your task. For example, since you can edit a report, the **Edit** key appears in the function key bar when you have a report on display. Since you cannot edit a function form, the **Edit** key does not appear in the function key bar when a function form is on display.

Table E-1. Function Key Descriptions

Function Key	Description
AddLIn	Adds a line to the report where the cursor is positioned.
EdIt	Brings you into edit mode. From there you can update a report without using an SOE character. Also from there you have access to the Line Change menu from which you can choose to add, delete, or duplicate lines.
Exit	Takes you out of the MAPPER system and back into the local environment.
Format	Lists the formats and procedures to access a given function through the control line. This key appears only when a help screen for a specific topic is on display. For example, if help for the Totalize function is on display, press **Format** to display a list of formats available for use with the control line procedure.
Help	Displays context-sensitive help and general-system help. For example, if you need help while filling out a function form, press **Help** and specific instructions are displayed. Note that the **Help** key is always available.
LIneCh	Displays a menu from which you can choose how to change a certain line. For example, to delete a line, place the cursor on the line to be deleted and press **LineCh**. Then move the cursor to the **Delete line** selection and transmit. This function key appears on the function key bar only after the **Edit** key has been pressed.
KeyHIp	Lists the names of some of the keys used in the documentation and their corresponding keys on your keyboard. For example, the text in **KeyHIp** might explain that the **Abort** key in the documentation is equivalent to a specific key on your keyboard. If you need definitions for keys other than those listed, refer to the printed documentation for your terminal.

continued

Table E-1. Function Key Descriptions (cont.)

Function Key	Description
Menus	Describes how to access a certain function through the menus. This key appears only when a help screen for a specific topic is on display. For example, if help for the Totalize function is on display, press **Menus** to display the menu path and procedure to access the Totalize function.
Op&Prm	Lists options and parameters for a given function. This key appears only when a help screen for a specific topic is on display. For example, if help for Totalize is on display, press **Op&Prm** to get a list of Totalize options and parameters.
Paint	Repaints the screen. For example, with a report on display type some text but do not transmit. Now if you press **Paint**, the report is redisplayed without the text you just typed.
Quit	Displays the active screen. If a help screen is on display, this key redisplays the screen you were viewing before you requested help.
ReadMe	Describes information, if any, specific to your MAPPER system. This information may not be documented anywhere else.
Relatd	Lists the screen titles of the current help topic and of topics related to the current help topic. This key applies only when a help screen is on display. For example, if the "Computing Data Using Calculate" help screen is displayed, press **Relatd** to find other help topics related to CAL, such as "Computing Data Using Arithmetic" or "Computing Data Using Totalize."
Remote	Lists systems to which you have access. You can transfer your station to another MAPPER site by pressing the **Remote** key, moving the cursor to a system on the menu, and transmitting.
Report	Lists all the reports to which you have access. You can choose a report in which to work by pressing **Report**, moving the cursor to a report on the menu, and transmitting.
Resume	Continues a function from the point at which it pauses or stops. For example, since the Find function pauses after it finds the first occurrence of a character string, you must press **Resume** to find successive occurrences.
Return	Displays a previous screen. For example, if you display a report while the active screen is on display, pressing **Return** then redisplays the active screen. (The screen you are returned to varies depending upon the current screen display and the path you used to obtain it.)

Table E-1. Function Key Descriptions (cont.)

Function Key	Description
RollFw	Moves to the next screen in a sequence of screens.
RollBk	Moves to the previous screen in a sequence of screens.
Runs	Lists all the runs to which you have access. You can execute one of the runs by pressing **Runs**, moving the cursor to one of the run names, and transmitting.
SgnOff	Brings you back to the sign-on screen. From there, you can sign on to MAPPER again or exit the MAPPER system.
SignOn	Signs you on to the MAPPER system as a new user. This gives you access to the demonstration database and the basic manual functions. This should be used for training purposes only.
SOE	Places a start-of-entry (SOE) character at the cursor position for data updates. **1100:** The **SOE** function key is not available on the OS 1100 MAPPER System. Use the SOE key defined in the documentation for your terminal.
Tasks	Displays a selection of jobs you might wish to accomplish, such as printing, comparing reports, or finding data. For example, if you want to print a report, press **Tasks**, then select **Print** from that menu. Another menu then displays specific print functions from which you can choose.
Undo	Reverses the last update function performed. This key does not undo the action of a run, but it does reverse the action of some functions, for example, deleting a report. Note that you can only undo the last operation if the **Undo** key appears on the function key bar.
View	Displays a menu of selections that allow you to control the report display. For example, to shift the report 35 characters to the left, press **View**, then enter 35 in the **Shift** menu selection and transmit.

Table E-2 indicates which function keys appear on certain MAPPER screens. For example, note that the **Help** key is available on every screen. The **LineCh** key, however, is available only while the system is in the report editing mode.

1100: The **SOE** function key is not available on the OS 1100 MAPPER System. Use the SOE key defined in the documentation for your terminal.

Note: *The underlining in this table indicates that the key is not available unless the action is possible. For example, after you have performed a function that can be undone, the* ***Undo*** *key is displayed.*

Table E-2. Screen Location of Keys on the Function Key Bar

	...these keys are available									
When this screen is on display...	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Sign-on	SignOn	KeyHlp				ReadMe		Help		Exit
Active	Report	KeyHlp	Runs			Tasks	Remote	Help		SgnOff
Drawer Select menu		RollFw	RollBk	Return		Tasks		Help		Quit
Report Select menu		RollFw	RollBk	Return		Tasks		Help		Quit
Select Run menu		RollFw	RollBk	Return				Help		Quit
Select System menu (Remote Selection)		RollFw	RollBk	Return				Help		Quit
Report	Resume	Paint	SOE	Return		Tasks	View	Help	Undo	Edit
Report Edit	AddLin	Paint	SOE	Return		LineCh	View	Help	Undo	
Line Change menu	Resume	Paint		Return				Help		Quit
Select Task menu (Active screen) (Report screen)		 Paint		 Return Return				 Help Help		 Quit Quit
Change View menu	Resume	Paint		Return				Help		Quit
Function form	Resume	Paint		Return				Help		Quit
Function mask	Resume	Paint		Return			View	Help		Quit
Function mask (View menu)	Resume	Paint		Return				Help		
Help menu				Return				Help		Quit
Help text	Relatd	RollFw	RollBk	Return			Menu	Help		Quit

Online Help for Keys

To obtain a list of some of the most common generic key names used in the documentation, press **KeyHlp** from the first screen that appears when you enter the MAPPER system.

Some MAPPER systems also provide a list of actual key sequences to use on your keyboard for various key actions. If this feature is available, you will see an entry in the first **KeyHlp** screen called **KeyMap**. Press the keys shown to the right of the word **KeyMap** to display this list of actual key sequences.

If **KeyMap** does not appear on the first **KeyHlp** screen, specific information for your keyboard is not available online. Refer to the printed documentation for your terminal.

You can also use the following methods to obtain the list of generic key names:

Method 1

1. Press **Help** until you reach the main help menu.
2. Tab to "Key Assignments for Documentation Key Names."
3. Transmit.

Method 2

1. After signing on to the MAPPER system, move the cursor to the control line.
2. Type **help,key**.
3. Transmit.

Use of the MAPPER Mouse

This section applies only to the U Series MAPPER System and UNIX MAPPER Systems when using a personal computer as the MAPPER terminal.

PC MAPPER: *The Personal Computer MAPPER System supports a mouse; however, you configure your mouse with the native operating system on your personal computer. No other configuration is needed. See the* Planning and Installation Guide *for details.*

If your terminal is a PC running MS-DOS®, you can use a mouse to position the cursor on MAPPER screens and to select functions.

Though you can use either a two- or three-button mouse, the MAPPER mouse uses only two buttons:

- **Left button.** Moves the cursor to the place where you are pointing and transmits. This button is called **MouseMit** in the documentation.
- **Right button.** Moves the cursor to the place where you are pointing and does not transmit. This button is called **MousePos** in the documentation. You can begin typing at that point and press **Transmit** when you are ready to transmit.

Installing the Mouse on Your PC

Store the files you need for loading the MAPPER mouse software on your PC by following this procedure:

1. Obtain a diskette from your coordinator that contains these files: *mapmouse.exe* and *mapmouse.mnu*.
2. Create a directory on the C drive of your PC called *mapmouse*. At the C drive DOS prompt, enter the following:

```
mkdir \mapmouse
```

3. Insert the diskette in drive A. At the DOS prompt, enter the following:

```
copy a:mapmouse.* c:\mapmouse
```

MS-DOS is a registered trademark of Microsoft Corporation.

After storing the files on your PC, follow this procedure to load the MAPPER mouse software on your PC. Since MAPPER mouse software will be resident in memory, load it before loading other software to avoid memory fragmentation.

1. Install the mouse hardware and software provided by the vendor.
2. Change to the directory containing your MAPPER mouse software. At the C drive DOS prompt, enter the following:

   cd \mapmouse

3. At the DOS prompt, enter **mapmouse**.

The MAPPER mouse software loads and writes a file containing local information. Once you have MAPPER mouse software loaded in your PC, the function of the mouse buttons is unique to MAPPER software.

Guidelines

- If an error occurs when you are loading the MAPPER mouse software, a message appears. Contact your coordinator for assistance.
- If you run another application that uses your mouse, reload the MAPPER mouse before returning to the MAPPER system.
- Here are the sizes of the mouse-loading files:

  mapmouse.exe — approximately 28K bytes

  mapmouse.mnu — approximately 4K bytes

Appendix F
Developing Source-Protected Applications

This section does not apply to OS 1100 MAPPER Systems.

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

Developing an Application

You can source-protect any of your MAPPER runs by requesting that your coordinator create a run-timed version. The coordinator uses a run called RUNTIME to accomplish this task. A source-protected run can only be executed — its run control reports cannot be displayed or updated.

There are two major advantages to creating run-time applications:

- Because a source-protected run can only be executed, you can distribute your application to sites without revealing the design or coding.
- A site to which you distribute the application must consult you for any modifications they want to make.

How to Design a Run-Timed Application

To create a run-timed application, follow these procedures:

1. Design your application so it can be installed on other MAPPER systems.

   For example, use Define Constant (DEFINE) statements to assign logical names to cabinet, drawer, and report locations, and any other data that varies depending on the system on which the application is executed. These statements can then be modified to specify the actual report locations when the application is installed at another site.

Appendix G
Setting up Local Code in Your MAPPER System

This section does not apply to OS 1100 MAPPER Systems or Personal Computer MAPPER Systems.

You may find additional information specific to your MAPPER system by pressing **ReadMe** from the sign-on screen.

You can add your own MAPPER functions at your site and access them through the Local Code (**cmd*) statement.

Using a set of defined interfaces included in MAPPER software, you can read, write, create, and delete MAPPER reports.

Writing Your Functions

Follow the instructions and examples provided in the following files in the /mapper/obj subdirectory:

Files	Contents
local.h	C heading file containing defines and function prototyping.
local.c	C code file containing local code parsing table and examples of code.
term.h	C heading file containing terminal output buffer data formatting commands.
mtypes.h	C heading file containing MAPPER variable defines.

If you are writing the function in a language other than C, refer to the appropriate compiler documentation for more information.

Accessing Your Functions

Follow this procedure to access your functions:

1. Add the new run function calls to the *local.c* file.
2. Ask the site administrator to compile and link the *local.c* file and your code modules.
3. Use the ***cmd** run function call in a MAPPER run, where ***cmd*** is the run function call of a site-defined local run statement that you added to the *local.c* file.

See also the ***cmd** run statement in Section 7.

Help Us To Help You

Publication Title _____

Form Number _____ Date _____

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments _____

_____

_____

Name _____

Title _____ Company _____

Address (Street, City, State, Zip) _____

Telephone Number _____

Help Us To Help You

Publication Title _____

Form Number _____ Date _____

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments _____

_____

_____

Name _____

Title _____ Company _____

Address (Street, City, State, Zip) _____

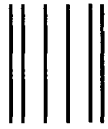
Telephone Number _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

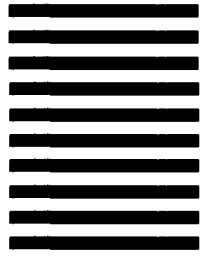
BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 1145 ST. PAUL MN 55164

POSTAGE WILL BE PAID BY ADDRESSEE

Unisys Corporation
Attn: MAPPER® Product Information
P.O. Box 64942 M.S.: 4792
St. Paul, MN 55164-0942 USA



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 1145 ST. PAUL MN 55164

POSTAGE WILL BE PAID BY ADDRESSEE

Unisys Corporation
Attn: MAPPER® Product Information
P.O. Box 64942 M.S.: 4792
St. Paul, MN 55164-0942 USA